Data Preparation and User Labelling for Time Series Classification

Dr. Albert Krohn, 21data.io
Adelina Barbur, Softing ROM s.r.l.

In this article we want to share our experience on applying classification algorithms for segmenting a large data set of industrial time series data. The classification is used as a data preparation step, to identify operation modes or system states of a time series recording of a large printing machine. We show important aspects of the practical application of classification algorithms. We take a special look at the influence on the classification results of supervised learning coming from imprecise user labelling.
We also motivate our work in the context of self-service analytics, which we understand as a crucial contribution for the successful implementation of data science solutions in industrial applications.

1 Introduction

The rising interest in the use of data derived from production and manufacturing system is a worldwide trend named under IIoT[1] or Industry 4.0. These movements proclaim the automatic and efficient use of all data with the aim to improve, automate or individualize the whole production chain and all involved machinery and processes. While the data acquisition, transformation and storage is covered by existing systems, most of the supporting systems for the data science lack automatic workflows and deployment. To overcome the necessity of a skilled team of data scientists and field engineers to address the data challenges in industrial manufacturing, we envision a more automatic way to work with the data through ready-to-use data pipelines. This approach has recently received attention under the terms *autonomous analytics*, *guided analytics* and *self-service analytics* with the idea to minimize the user interaction and maximize autonomy of the algorithms pipelines used.

[1] IIoT: Industrial Internet Of Things

1.1 Use case

In this article we look at the use case of rotational gravure printing machines typically used for printing high circulation products like newspapers. Those machines normally fill a whole production hall spanning 100m in length and several floors in height and run at a speed of up to 50 km/h. Paper is fed into the machine from large paper rolls (typically 1.5m in diameter and up to 4m in width, several tons in weight) and runs at full speed through the machine where printing, cutting and folding takes place. To achieve maximum uptime, several strategies have been developed to be able to keep the printing and production process running all the time - even during the change of the feeding paper roll. One idea is to glue a new paper roll at the end of a used-up paper roll and thus provide an endless flow of paper into the printing machine.
Several data science challenges have been identified to be or interest in this scenario:

- root cause analysis of paper rips during production
- key performance indicator derivation (KPIs)
- anomalies during production

For all these three groups, we need to prepare the data in a similar way in a workflow pipeline

1) data loading
2) data cleaning
3) feature engineering
4) system state identification / time segmentation
5) specific analytics

The identification of the current system state is crucial for many algorithms to work well. Anomaly detection works much better if we can tell what the "normal" operation state is supposed to be; so we need to separate the system state "production" from another state that could be "maintenance". The identification of the system state is realized through the *segmentation/classification of the time series data,* applying a system state to each point in time. After this, different approaches are taken in the

data science to cover for root cause analysis with the help of explanatory model, KPIs and anomalies. In the following chapters we will now take a closer look on time series segmentation/classification as it is of multiple use in the present applications and also has a general applicability for many other use cases.

1.2 Approach and problem statement

Time series segmentation (or classification) is a well studied discipline. We refer to one the largest meta-study on time series classification (Bagnall et al. 2016) for further reading. In the current literature, nearly all areas of machine learning have been applied for the purpose of time series classification: be it decision trees, unsupervised methods, clustering, use of artificial neural network and many proprietary algorithms. We decided to use a random forest classifier with multivariate input. To support our hypothesis that classification might help us to distinguish between system states, we show how the different states look on our data and discuss why we expect that it generally works.

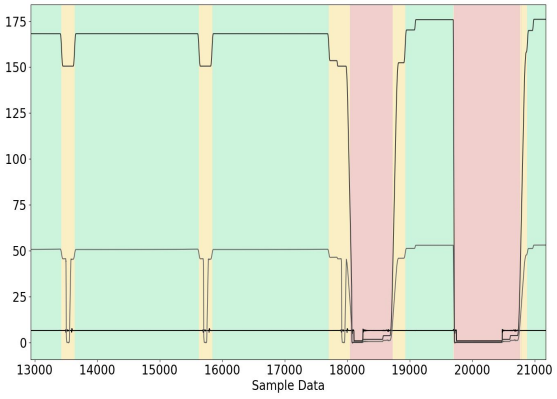


Figure 1 - Sample data

Figure 1 shows a sample of 3 sensors plotted over time. For ease of understanding we have marked the different systems states we want to classify with background colors. As we can see, each recorded sensor has a specific behavior dependent on the state in which the machine is running. The fluctuations in values are significant, which makes the different operation states clearly separable.

To segment the time series in a supervised way, we need labelled data based on expert knowledge through manual annotation. During that process of labelling we encountered several obstacles:

- even for an expert it was not always easy to label correctly the events by just looking at plotted curves of measurements
- the pure amount of data we covered in our study (much more than the test data) would take days to be completely manually annotated with the danger of human mistakes
- the time-precision of manual annotation is limited but will severely influence the overall performance

This first manual annotation is a crucial step in the classification process, as it will heavily influence our later results. We will detail on this in the following chapter and then conclude how it affects an autonomous way of processing and propose some pragmatic solution.

2 Time series segmentation with a tree classifier

2.1 The test data

The test data is an excerpt of approx. 8 hours of machine data with 96 sensors sampled parallel at 1 seconds rate (~30 k data vectors).

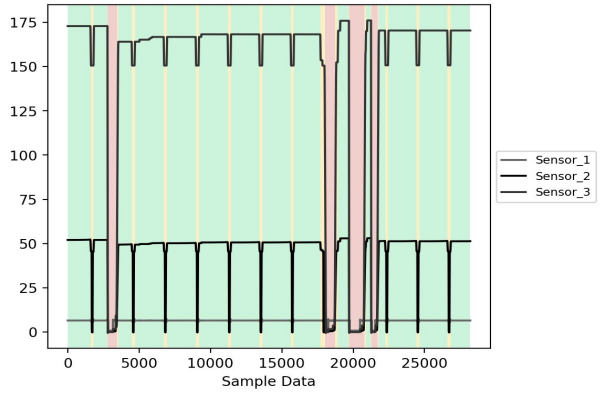


Figure 2 - The test data

Our test data is basically divided into three areas (i.e. three system states): a normal operation (green), a machine failure (red), and changing of paper rolls in the printing process (yellow). For training our random forest classifier we have selected and labelled a few time areas representative for each system state.

2.2 Data preprocessing

One of the many advantages of decision trees is that they don't require significant pre-processing of the data. Because decision trees split nodes by thresholding, any monotonic transformations applied to the data -like centering, scaling - will not

influence the results, as they cannot change the way the data is divided.

In the Random Forests algorithm, trees are built using a randomized subset of features, selected in a way which minimizes the correlation between trees. This tackles two major drawbacks of more traditional algorithms, namely multicollinearity bias and the so called "curse of dimensionality". (Janecek et al. 2008; Pappu and Pardalos, n.d.)

Although ensemble classifiers such as Random Forests can manage high dimensional data without requiring feature engineering, it has been shown that in some cases, data transformations such a principal components analysis improved results' accuracy. However, whether such data wrangling techniques will prove effective, heavily depends on the data. (Janecek et al. 2008)

We wanted to see if data engineering would improve our classifier's performance, so we applied different transformations. First, we classified on normalized and unnormalized data. We also tested if dimensionality reduction would help us classify systems states more accurately. So for this, we applied principal component analysis on the data and used the resulting components as input for our classifier. The results are presented below.

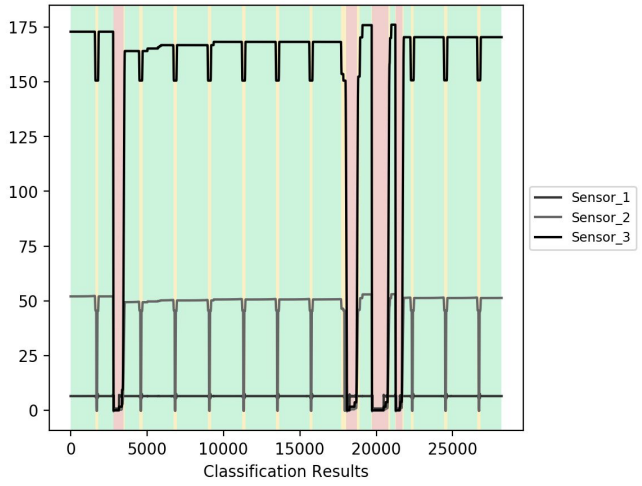


Figure 3 -Classification on unnormalized data

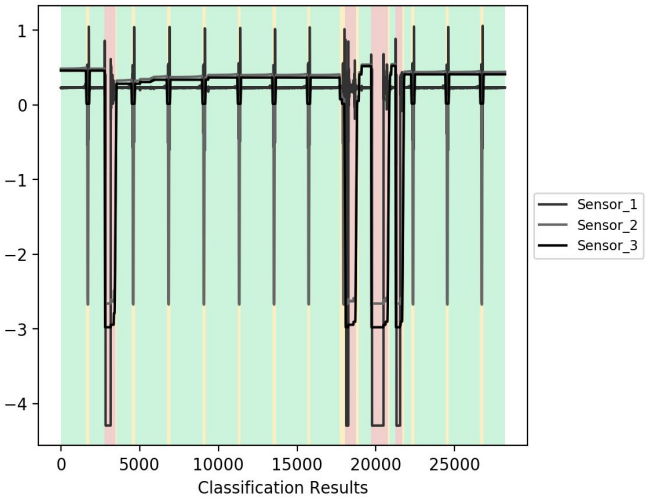


Figure 4 -Classification on normalized data

As expected, normalizing the data before classification has not changed the results.

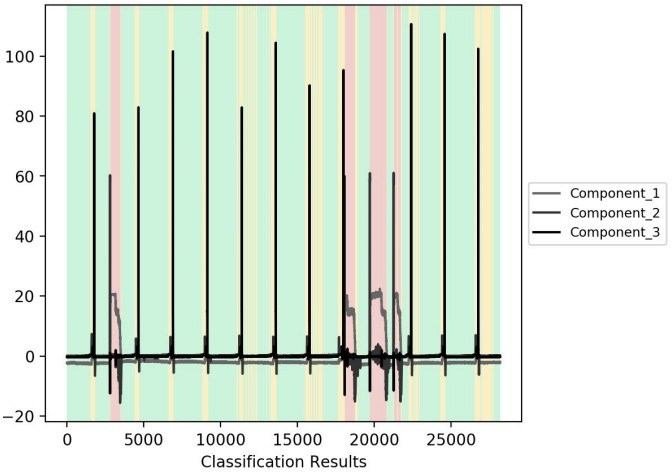


Figure 5 -Classification on PCA data

For the PCA, we reduced dimensionality from 96 features to 20 features (equivalent to a proportion of 90% of explained variance). When taking a closer look at the results we can observe that the paper roll changes (yellow) are not so well separated anymore - in the transition phase between normal operation (green) and paper roll change (yellow), classes alternate- and there is also some misclassification of normal operation (green). When reducing the dimensionality of the data by almost 30%, vital information can get lost, so in order to validate whether this is the main reason behind the decrease in accuracy we repeated the experiment, only that this time we kept all 96 dimensions. Basically, we classified on a linear and uncorrelated transformation of the original data.

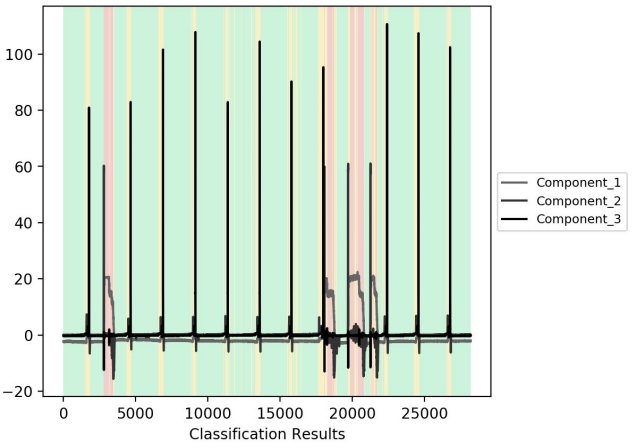
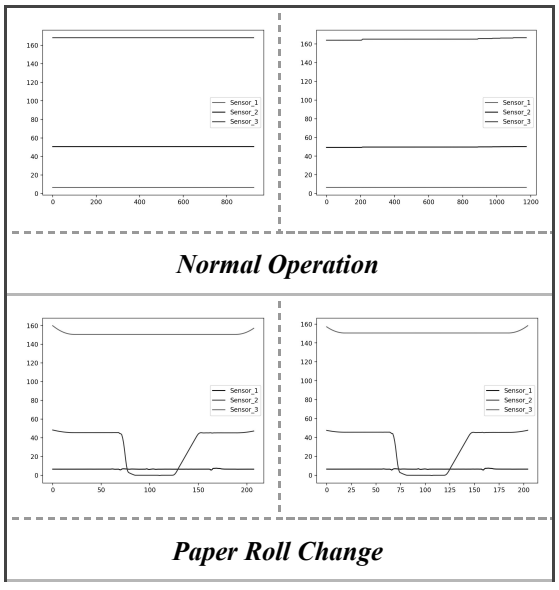


Figure 6 -Classification on PCA data

As we can see, the problems encountered previously persist. There is some misclassification of the downtime (red), and some of the paper roll changes(yellow) are not clearly separated. Therefore, in our case, PCA has not improved the classification accuracy, on the contrary. We decided to use all 96 unnormalized features for our further analyses.

2.3 Influence of user annotations

Like with any algorithm, having a good training data set is crucial. In order for random forests to provide highly accurate results and great generalization power, we need to have a very good and comprehensive representation of the data. In our case, it means to have a good collection of samples of the behavior of the machines in different system states, represented by annotations.



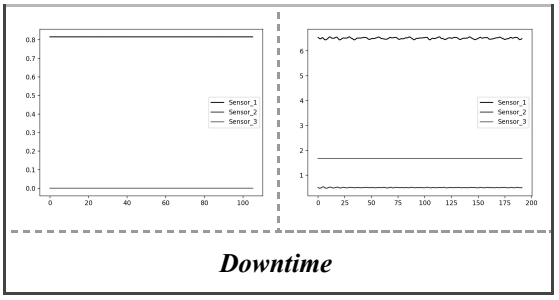


Table 1 -Annotations used for learning

Table 1 shows the collection of the system states used for training. We have used 2 different areas to describe normal operation, 2 for paper roll changes and 2 for downtime. As during downtime, there are a lots of changes in the behavior of the the machines, we tried to capture them as good as possible with our annotations.
Initial classification results on the dataset (Figure 7) show a clear classification of the 3 system states, namely normal operation (green), paper roll change (yellow) and downtime(red).

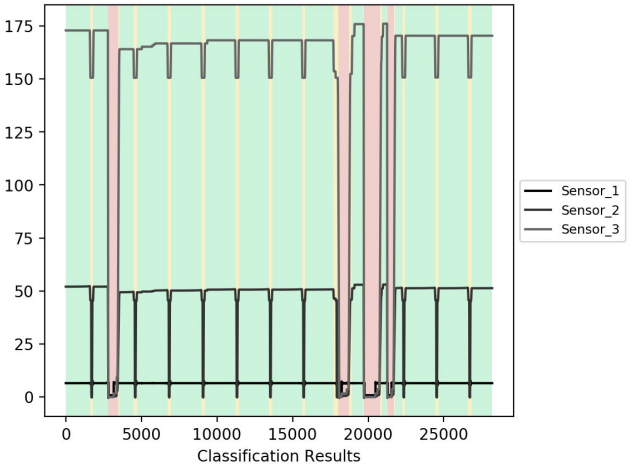


Figure 7 -Initial classification results

2.3.1 Overlapping classes

One important aspect which can seriously jeopardise the accuracy of a classifier is the problem of overlapping classes. Having data samples appear as valid examples of more than one class will prevent the algorithm from building clear discrimination criteria.(Das, Krishnan, and Cook 2013; Xiong, Wu, and Liu 2010)
In our case, the problem of overlapping classes arises when the preciseness of annotations is not good. To illustrate this, we have expanded the expert's paper roll change annotations by 20% in time which would result in an overlap between normal operation and paper roll change, and re-classified.

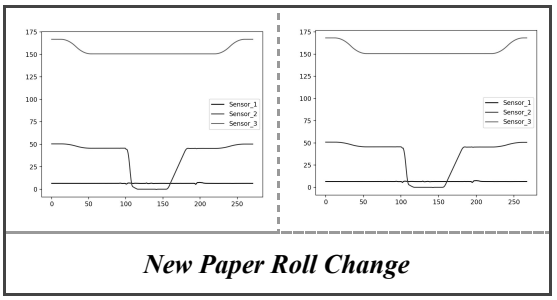


Table 2 - Annotations with increased borders

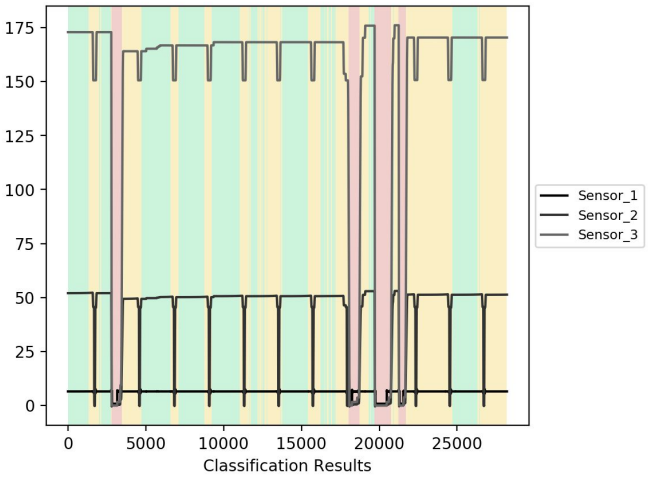


Figure 8 -Classification with overlapping classes

As we can see, our model has serious trouble in discriminating between normal operation (green) and paper roll changes (yellow).

2.3.2 Data drifts

So far, we have seen that training on overlapping classes can seriously compromise our results. These can arise from imprecise annotations of the data, or from changes in the data, which result in capturing the same behavior in two different classes.

In our dataset, we can see that the normal operating state is not consistently represented, the machines functioning at lower level after the 1st downtime, and and higher levels between the 2nd, 3rd and 4th downtime. This could be a normal part of the ramp up process after downtime.

In order to test the robustness of our classifier against these drifts, we first ran it using precisely annotated paper roll changes and have not captured these drifts in the training set. Results in Figure 9 show that, with precisely annotated paper roll changes, the classifier can manage these data drifts.

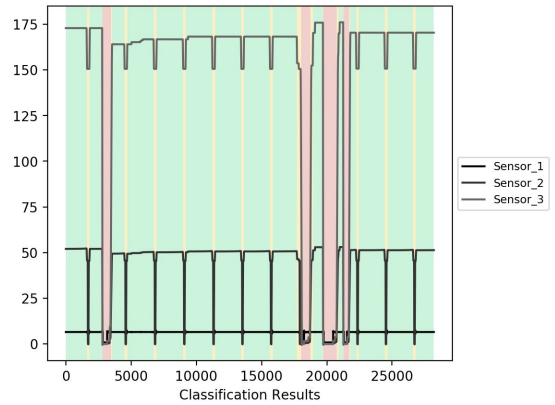


Figure 9 -Classification without training on data drifts

Next, we increased the borders of our paper roll changes by 20%, introducing an overlap between this system state and the normal operating state, and kept the same annotations for the latter.

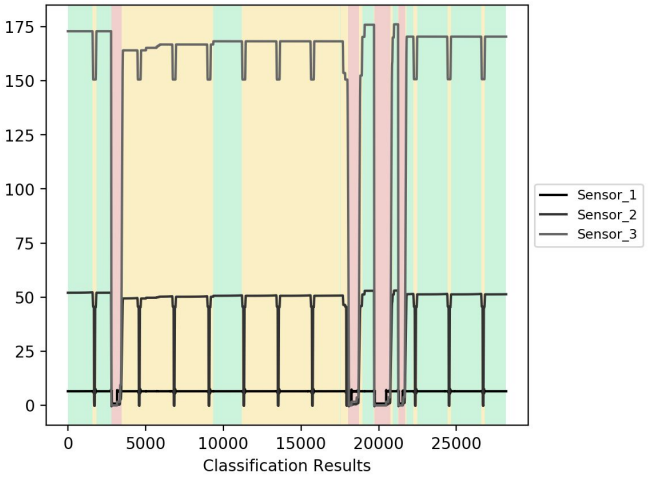


Figure 10 -Classification without training on data drifts and with overlapping classes

From Figure 10 we can observe that before and after the data drifts, the classifier cannot distinguish correctly between the classes.

We wanted to see if including the data drifts in the training data set would help improve our results, and reduce the effect of overlapping classes, therefore we changed the annotations representing the normal system state to include the data drifts.

The results in Figure 11 show an improvement in classification especially in the area of the data drift (after the first downtime), however, the classifier still has difficulties in determining the correct classes, especially after the 2nd and 3rd downtime areas.

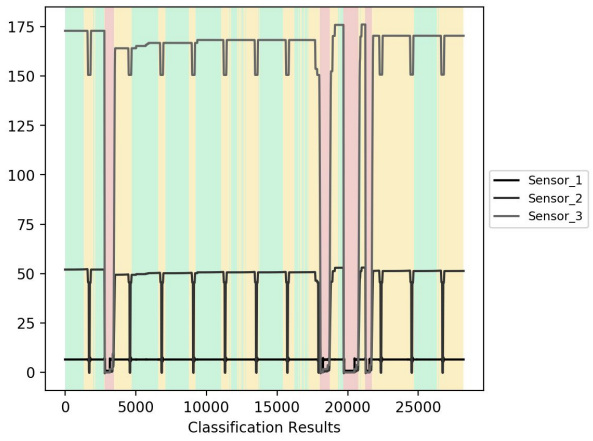


Figure 11 -Classification with training on data drifts and overlapping classes

In our example, random forest classifiers can be robust against data drifts, without requiring training on those areas, but only in the absence of the overlapping classes. Including data drifts in the training data can, to some extent, improve performance in such cases, however, not to a satisfactory level.

2.4 Unbalanced classes

Another well recognized challenge is the class imbalance problem. The class imbalance problem is concerned with the performance of classifiers when one or more classes are underrepresented in comparison with the other classes. Classification algorithms such as random forests minimize the overall error rate, which means that a misclassification of the underrepresented class (the "rare" class) will not be costly for the algorithm's accuracy rate.

There are several balancing methods proposed, which can be non-heuristic such as random undersampling or oversampling, or heuristic like NCL (Neighbourhood Cleaning Rule), Smote (Synthetic Minority Over-sampling Technique) or a combination of the two. (Prati and Monard, n.d.)

For random forest specifically, balancing methods include using stratified bootstrap when building the trees, or assigning weights to each class, and penalize more the misclassification of minority classes. (Chen, Liaw, and Breiman 2004).

Although the classification obtained was satisfactory, we wanted to see how would balancing our data set influence the results. In our data set, the minority classes are represented by the paper roll change, and the downtime system state; the data is split as follows: 10.6% downtime, 6.9% paper roll changes and 82.5% normal operation. For balancing, we decided to use the weighting method.

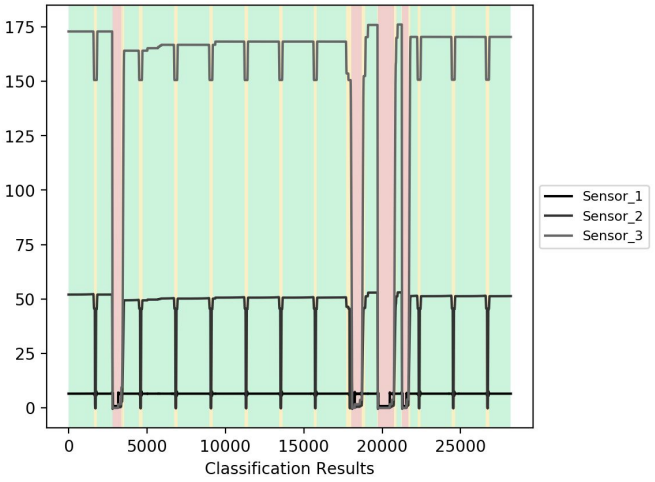


Figure 12 -Classification with balanced classes

Results show neither an improvement, nor a deterioration of accuracy, an indication that our data set is not affected by the class imbalance problem even though the distribution of system states over time is not equal.

2.5 Supporting user annotations

So far, overlapping classes turned out to be the most problematic aspect of our classification job.

In order to mitigate this risk, we need to make sure that the labelling, done through manual annotations in our case, is as precise as possible. Furthermore, in order to ensure a high generalization power, a comprehensive collection of annotations is needed. This can make the process of annotating a lengthy and exhaustive one.

To support the user and automate the process as much as possible, we propose the use of additional steps like :

- using clustering to determine the optimum borders of classes
- using anomaly detection to find "rare" classes
- using pattern mining to extend the training data set

2.5.1 Clustering

To illustrate how clustering can be used to support annotations, we used an annotation representing a paper roll change, whose borders have been increased by roughly 50% (Figure 13) and ran a k means (k=2) on it. Our intuition is that, by using k

means with 2 clusters, we will be able to distinguish between our area of interest and surplus.

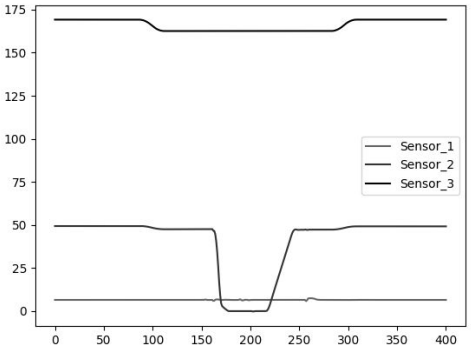


Figure 13 -Paper roll change with shifted borders

As it can be seen in Figure 14, our clustering algorithm separates the paper roll change (yellow) and surplus (green). The surplus actually represents and overlap area with the normal operation system state.

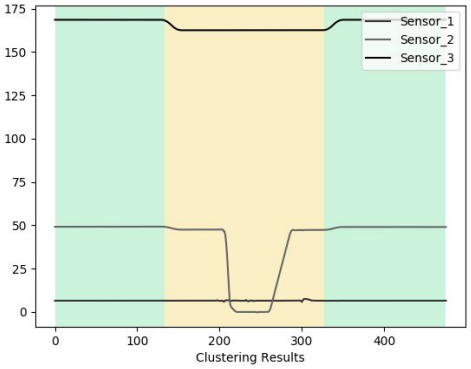


Figure 14 -Clustering results k=2

We repeated this step for the other paper roll change annotation, and classified again using random forest.

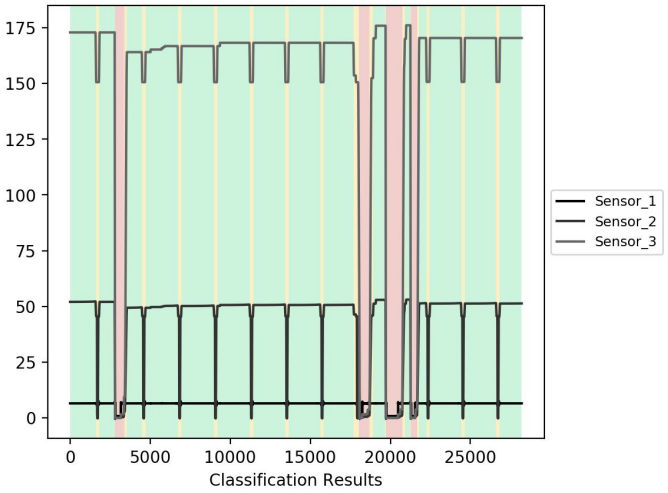


Figure 15 -Classification results with clustered annotations

The results obtained (Figure 15) are identical with the ones from Figure 7, which leads us to conclude that using a clustering technique to make annotations more precise is a feasible to stabilize imprecise user labelling.

2.5.2 Pattern mining

To take a step further towards making the process as autonomous as possible, we try to extend our training data with the use of machine learning algorithms. For us, this translates into starting with a user defined annotation to represent each system state, and use pattern mining to find and mark additional similar areas; the results of this process will constitute the new training data set for our classifier. The patterns used are the 1st annotations for each system state from Table 1.

For finding these areas we used SAX (Symbolic Aggregate Approximation), a pattern mining algorithm which transforms and reduces the time series to a string of arbitrary length, and then determines similarity based on a non-linear distance. A more detailed overview of the approach can be found in the following papers (Lin et al. 2007; Megalooikonomou et al., n.d.)

We've decided to use half of the dataset to mine for manually annotated patterns, and kept the other half as novel data to test our classifier's predictive power.

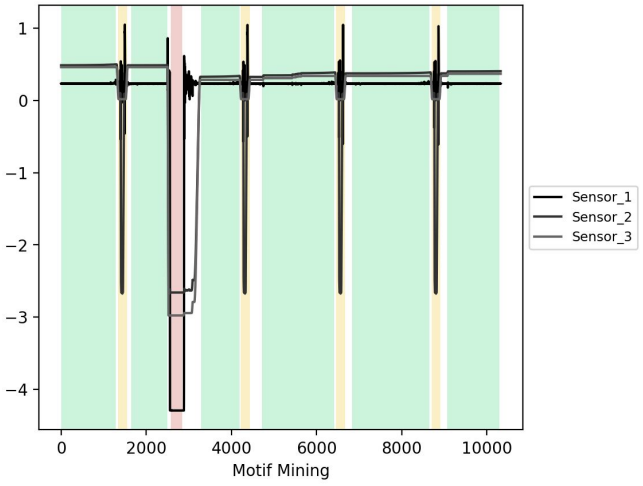


Figure 16 -Pattern mining results

The setup of our pattern mining algorithm was: alphabet size =5, number of PAA segments = 10, distance threshold = 0.35.

The results of the pattern mining process are shown in Figure 16. Yellow represents paper roll change, green represents normal operation, while red represents the downtime area. As we can see, the pattern miner did a fairly good jobs at identifying areas of interest. There are some white backgrounded areas, for which no match has been found, that, due to the strict restrictions regarding both shape and size imposed by SAX, will fall in between patterns.
However, with the help of pattern mining we managed to enlarge our training data set by a factor of 2, with minimum effort.
We run our classifier again on the entire dataset, and obtained the following results:

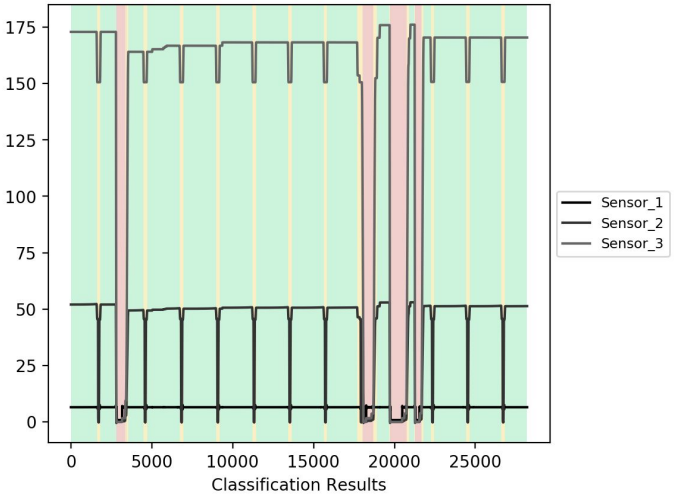


Figure 17 -Classification results with mining results as training data

As expected, the results are identical to the ones from figure 7. Furthermore, by increasing the size and representativity of our training dataset, we have neutralized problems such as overlapping classes and sensitivity to data drifts.

3 Conclusion

In this article, we have discussed the experience we made during the processing of a large industrial time series data set. As a data preparation step, we needed to segment our time series in areas of interest for further execution of algorithms like anomaly detection or predictors. For this common preparation step we discussed the use of normalization or PCA for dimensionality reduction which showed negative influence on our results. We also looked into data drift and the influence of imprecise user annotations leading to overlapping and unbalanced classes which were identified as a major source of classification weakness of our random forest classifier.
We also showed that simple supporting steps like auto-cleaning the borders of annotated motifs with the help of a clusterer can significantly improve the classification result.

References

Bagnall, Anthony, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2016. “The Great Time Series Classification Bake off: A Review and Experimental Evaluation of Recent Algorithmic Advances.” *Data Mining and Knowledge Discovery* 31 (3): 606–60.

Chen, C., A. Liaw, and L. Breiman. 2004. “Using Random Forest to Learn Imbalanced Data.” *University of California, Berkeley*. http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

Das, B., N. C. Krishnan, and D. J. Cook. 2013. “Handling Class Overlap and Imbalance to Detect Prompt Situations in Smart Homes.” In *2013 IEEE 13th International Conference on Data Mining Workshops*, 266–73.

Janecek, Andreas, Wilfried Gansterer, Michael Demel, and Gerhard Ecker. 2008. “On the Relationship Between Feature Selection and Classification Accuracy.” In *New Challenges for Feature Selection in Data*

Mining and Knowledge Discovery, 90–105.

Lin, Jessica, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. “Experiencing SAX: A Novel Symbolic Representation of Time Series.” *Data Mining and Knowledge Discovery* 15 (2): 107–44.

Megalooikonomou, V., Qiang Wang, Guo Li, and C. Faloutsos. n.d. “A Multiresolution Symbolic Representation of Time Series.” In *21st International Conference on Data Engineering (ICDE’05)*. https://doi.org/10.1109/icde.2005.10.

Pappu, Vijay, and Panos M. Pardalos. n.d. “High Dimensional Data Classi Cation.” http://plaza.ufl.edu/psnvijay/Site/Publications_files/Classification_HDD.pdf.

Prati, Ronaldo C., and Maria C. Monard. n.d. “Balancing Strategies and Class Overlapping?” http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.829&rep=rep1&type=pdf.

Xiong, H., J. Wu, and L. Liu. 2010. “Classification with Class Overlapping: A Systematic Study.” *The 2010 International Conference on E-*. http://www.atlantis-press.com/php/download_paper.php?id=2053.