Classifying and explaining defects with small data for the semiconductor industry

Jean-François Boulanger, Franck Corset, Franck Iutzeler, Jérôme Lelong

Classifying and explaining defects with small data for the semiconductor industry

Jean-François Boulanger* Franck Corset† Franck Iutzeler‡
Jérôme Lelong§

January 26, 2022

Abstract

In this work, we present an automatic classifier of wafer defects for the semiconductor industry. Hopefully defects are rare, but this puts the classifying problem in a small data context. We propose a fast and fully reproducible approach based on decision trees. The main interest of using decision trees lies in obtaining a highly explicable classifier, which makes the origin of the defect easy to identify.

1 Introduction

In this note, we provide an overview of our work on the classification of wafer defects as part of an industrial collaboration between Univ. Grenoble Alpes and Unity^{SC}.

Context Maimosine (`https://www.maimosine.fr/`) is a federative research structure hosted by Laboratoire Jean Kuntzmann, Université Grenoble Alpes and CNRS, that aims at promoting the development of cross-disciplinary projects based on mathematical modelling and simulation. One of Maimosine's main goals is to boost collaborations between academic research and innovative companies in the Grenoble area.

Unity^{SC} develops and manufactures high precision tools for the semiconductor industry worldwide. Research focused, the company's strategy is to design and integrate cutting-edge optical sensors and advanced algorithms to achieve the best measurement performance. The applications of the company automatically detect and classify small-dimension defects on customer products and measure feature sizes for process control.

The goal of this collaboration is to classify manufacturing defects on wafers from real data provided by the company. The data comes from physical measurements and various quantities automatically derived from proprietary image processing techniques. So far, this classification task was performed manually by highly qualified engineers which was very costly and time-consuming; in addition, the produced classification suffered from consistency issues depending on who was in charge of the classification. This work aims at replacing the manual work by a supervised learning approach of the defect classes based on data already labeled by experts. As an additional requirement, the approach should return a set of constraint boxes on the input features, each

*Unity^{SC}, 611 rue Aristide Bergès, Z.A. de Pré Millet, 38330, Montbonnot-Saint-Martin, France. *Email*: jf.boulanger@unity-sc.com

†Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France. *Email*: franck.corset@univ-grenoble-alpes.fr

‡Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France. *Email*: franck.iutzeler@univ-grenoble-alpes.fr

§Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France. *Email*: jerome.lelong@univ-grenoble-alpes.fr

constraint box corresponds to the prediction of a given defect. This additional requirement has emerged during preliminary discussions with the company, stemming from interpretability and real-time needs on the prediction process.

Learning a classifier A classification problem consists in learning a mapping h from some input space $\mathcal{X}$ to one of K possibles classes. To do so, from a data set consisting of n pairs of example/class $\{x_i, y_i\} \in \mathcal{X} \times \{1, .., K\}$, $i = 1, .., n$, the objective is to find a mapping that fits well the given data and also generalizes well to other pairs, provided that they are not too different from the original data.[1]

Lots of models and associated algorithms exist to find such predictors (Nearest Neighbors, Support Vector Machines, Logistic Regression, Random Forests, Neural Networks to name a few; see [2, 3] for a review and [4] for a Python implementation and comparisons). Beyond their performance, these models differ a lot in terms of prediction functions. Indeed, if a Neural Network is trained, the prediction consists in feeding it the new example to classify, which can be costly in terms of storage and prediction time on small processing units. Furthermore, these kind of methods are sometimes decried in industrial applications for their lack of interpretation. Linear methods such as SVM or logistic regression predict a class from a linear combination of the input coordinates/features, which is easier to interpret in particular when $K = 2$, but may suffer from performance limitations. Decision trees offer a good compromise between performance and interpretability. These methods consist in sequentially partitioning the input space by finding the feature and value that separate best the different classes.

Specificity of the task Interpretability was indeed a key concern in our work. The classifier produced by the supervised learning approach had to be understandable by the senior engineers at Unity who used to carry out this task manually. Thus, our goal was to output simple rules in order to enable them to quickly identify the production step responsible for the defect in the manufacturing process. Moreover, the classifier had to be encoded in XML using only boundary constraints on the input features. Then, it was obvious that the classifier had to expressed in terms of "boxes" written as a product of one dimension intervals. Decision tree classifiers are precisely designed to produce such classification rules and were thus our workhorse in this collaboration.

2 Modelling and Methodology

2.1 Data set

The data provided by the company is a set of features obtained from a multi-stage image processing on 1190 defective wafers, representing 13 common types of defects. Each wafer is photographed using three acquisition modes leading to as many data layers. Then, these layers are analyzed to detect potential defects. They are presented as small thumbnails from which a predefined set of features are measured. Our features correspond to these 195 extracted values and the defect class is annotated manually by a senior engineer.

Obviously, some defects may not be visible on all layers, which implies that the data set obtained by aggregating the data from all the layers has a lot of missing data. Note that in this context a missing data is actually very meaningful since a value is missing when the associated feature could not be measured because it was not present in the thumbnail. As almost all the features were actually representing lengths, it was crystal clear that missing data had to be set to 0. Another possibility would be to add for every layer a Boolean variable stating the presence of absence of the defect in the given layer in order to obtain a more easily readable classifier, but this solution was found to be less robust while offering virtually the same performance.

[1] From a mathematical point of view, we can model the data set as a sampling from some unknown distribution. Then, we expect our predictor to perform well on other pairs sampled from the same distribution.

Classification results highly depend on the training data. In particular, anyone who intends to use a classification algorithm must be fully aware that any bias in the training data will automatically be passed to the classifier. This phenomenon is well-known by researchers working on automatic classification and naturally tends to expand when using reinforcement learning as the algorithm learns from its own decisions. As a guarantee against these bias issues or outliers, we advise to remove from the input dataset any defect class with too few samples, which would produce classification rules without any real foundation, especially since some of these defects were flagged as borderline by the experts. In practice, we remove any defect class with less than 10 samples in our tests.

2.2 Supervised learning and decision trees

Consider a training set built of n labeled samples $\mathcal{D}_n = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{K},\, i = 1, \dots, n\}$, where $\mathcal{X}$ is the feature space and $\mathcal{K}$ the types or classes of defects. A classification can be modeled as a function $h : \mathcal{X} \longrightarrow \mathcal{K}$, which associates a defect class to an element of $\mathcal{X}$ (typically a subset of $\mathbb{R}^p$ where p is the number of features). To measure the fitness of the classifier, we usually define a merit function. For instance, it can be its *accuracy* defined as the fraction of correctly classified data

$$M(\mathcal{D}_n; h) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i; h) \qquad \text{where } \ell(x, y; h) = \begin{cases} 1 & \text{if} \quad h(x) = y \\ 0 & \text{otherwise} \end{cases}$$

In this work, a classifier can be identified to a partition of $\mathcal{X}$ in which every element of the partition (referred to as a box) can be written as a product of intervals, *ie.* $\mathcal{X}$ is split into disjoint sets $\mathcal{B}_1, .., \mathcal{B}_b$ taking the form

$$\mathcal{B}_j = \{x \in \mathcal{X},\, \underline{j_1} \leq x[1] \leq \overline{j_1}, \dots, \underline{j_p} \leq x[p] \leq \overline{j_p}\}$$

where $x[k]$ represents the k-th element of x (hence the k-th feature among the p features). Then, for every j, $\mathcal{B}_j$ is associated to a given defect class. Let us denote this mapping by $a : \{\mathcal{B}_1, .., \mathcal{B}_b\} \to \mathcal{K}$. In this context, the classifier h can be written

$$h(x) = \sum_{j=1}^{b} a(\mathcal{B}_j) 1_{x \in \mathcal{B}_j}.$$

Decision Trees [1] are known to produce a partition of the feature space of the form $\{\mathcal{B}_1, .., \mathcal{B}_b\}$ along with the mapping a. Indeed, they consist in iteratively partitioning the input space along one feature at a time, resulting in a tree-like structure whose leaves represent the boxes of our classifier. Their main parameters are how to find the best possible splitting thresholds, which is done by looking at the split that increases most the purity of the leaves measured in terms of Gini coefficient or entropy; and how many leaves to output, which can be set by either constraining the depth of the tree or its width. Although, decision tree classifiers are sub-optimal, their accuracy is already very good and they are obtained using cross-validation.

The number of boxes b (or equivalently of leaves in our case) plays a major role in the balance between goodness of fit and generalization. Indeed, with b sufficiently large, one can form a box around every input point to achieve a perfect accuracy; however, the generalization to unseen data can be arbitrarily bad. Thus, we fix a maximal number of boxes B by cross-validation before finding the optimal classifier with at most B boxes on the full dataset. As an illustration, we display in Figure 1 a decision tree for our problem with 6 boxes (the number of boxes is actually much greater in practice).

2.3 Performance & Validation

Decision trees are prone to over fitting, that is why we cross validate the parameters of the tree (maximal width, depth, minimal number of samples in a leaf, impurity measure) by performing 10-fold cross validation.

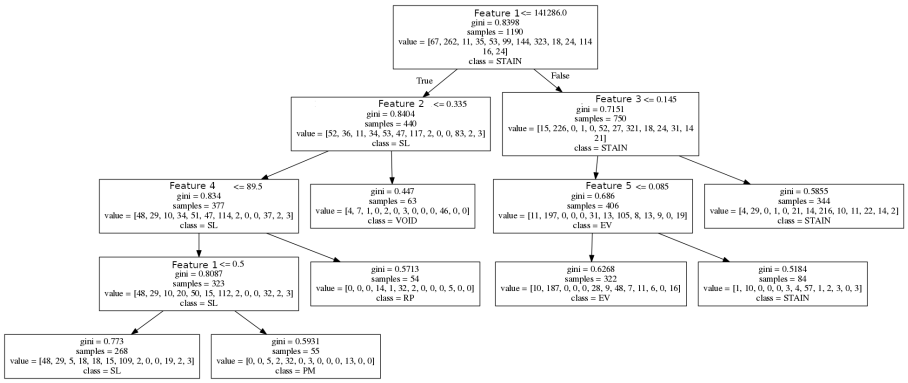


Figure 1: Example of a Decision Trees with 6 leaves/boxes

We compared the obtained accuracy to other classifiers in Table 1. We observe that the Decision Tree offers a much better accuracy than the Logistic regression, this is mostly due to the highly non-linear aspect of the problem.

Method	Accuracy		Comments
	ave.	sdev.	
Decision Tree	0.605	(0.083)	method used
Logistic regression	0.420	(0.076)	linear classifier minimizing the cross-entropy loss
Random Forest	0.635	(0.072)	ensemble of 100 random decision trees
Gradient Boosting	0.720	(0.086)	gradient boosted ensemble of 100 decision trees

Table 1: 10-fold cross validation performance of several methods. We display the average accuracy on the 10 folds as well as the standard deviation.

We also compare decision trees to ensemble methods. These methods are based on the aggregation of several trees and are known to overcome the tendency to over fitting of decision trees; they also usually offer good performances. Thus, they can be seen as a performance target in our case. Compared to decision trees, theses approaches perform respectively 3% and 11% better. However, these methods aggregate several trees (usually 100) and thus produce decision frontiers that are much less interpretable and implementable, making them unfit for our target problem.

3 Impact for the company

The method described above has been integrated in the processing tools of Unity^{SC}. A learner module is first presented to the operator to label imagelets of defects detected on a wafer like represented on Figure 2. This process is done offline during the recipe creation. Then, from the learned classification tree, an automatic classification module for production has been integrated. It allows process engineers to see in a glance the results of defect classification as represented on Figure 4 where the location and labels of defects are displayed on the wafer map. The graylevel and binary mask imagelets of the defects are also accessible through that interface.

The example presented across the different figures corresponds to an automatic defect classification applied to data acquired by the edge inspection module developed at Unity^{SC}. This module composed of several cameras acquires graylevel images of the edge of wafers. Then, the defects are detected through an automatic image processing framework.

For this example more than 80000 defects are detected on the edge of the test wafer (which is a highly contaminated one compared to what is usually inspected). Depending on the nature of defects, several classes are defined as represented on Figure 3. Defects of different nature are

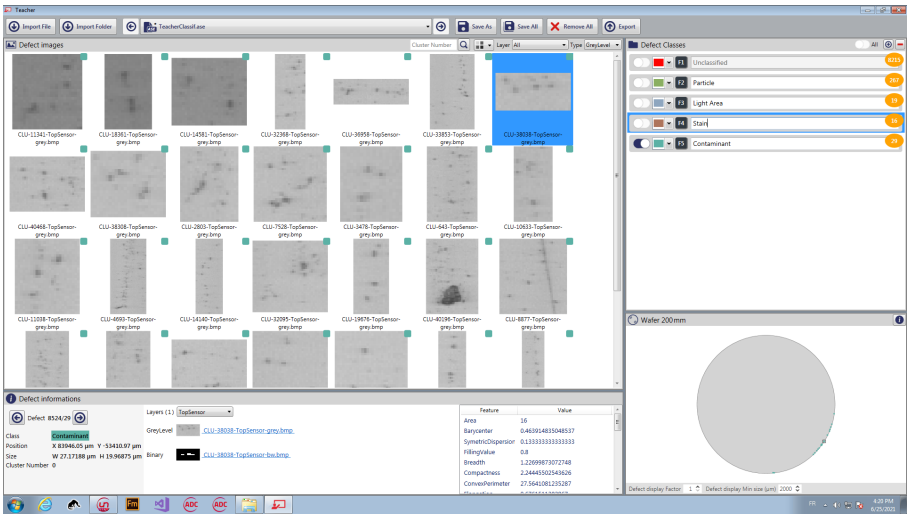
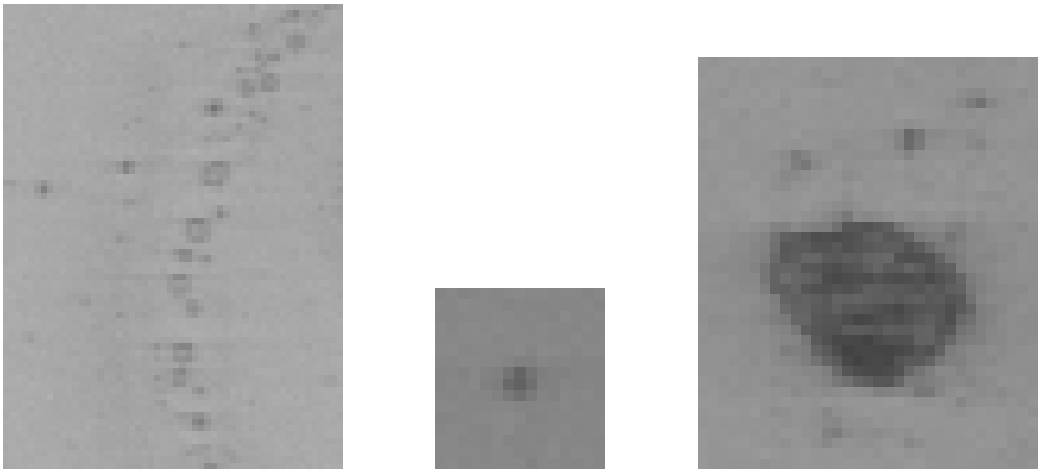


Figure 2: Interface of the learner module interface. Example of learned defects: contaminant



(a) Contaminant (b) Particle (c) Stain

Figure 3: Image examples of 3 different defect classes.

present and correspond to process residue or contamination by the environment. Automatically making the difference between the cases is often a key for the final customers.

When tackling application cases where many singularities (*i.e.* area on wafers declared as defects on the detection phase) are present on the surface of wafers, the software module developed during this project presents several advantages in terms of reliability and flexibility.

4 Conclusion

In this work, we have designed a reliable classification tree which reached the specified performance in term of precision. The method allows us to save time and increases the reliability of the classification process. The produced classification tree eases the work of application engineers and avoids them to waste time on complex data visualization and manual classification tree set up. The effort is actually focused on acquisition and labelling, which are straightforward tasks. The result is also reproducible and maintainable since it does not depend on the subjective choice of an operator on what feature to select and what threshold to set.

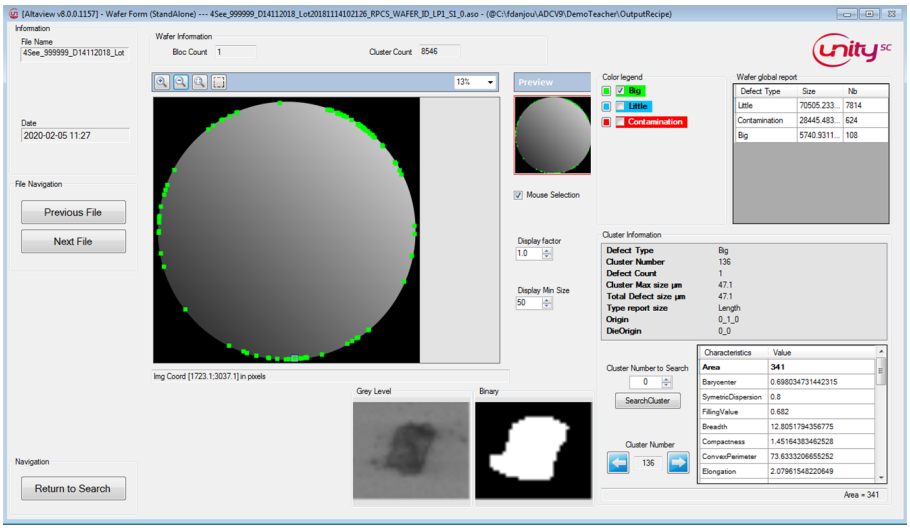


Figure 4: Interface of the classification presented to the final user. Locations and labels are displayed on the wafer maps. Examples of defect grey image and binary mask are visible.

References

[1] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[2] László Györfi, Michael Kohler, Adam Krzyżak, and Harro Walk. *A distribution-free theory of nonparametric regression*, volume 1. Springer, 2002.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[4] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.