Yield Modeling with Rule Ensembles

Giovanni Seni, Edward Yang, Said Akar
PDF Solutions
San Jose, CA 95110 USA
E-mail: giovanni.seni@pdf.com, edward.yang@pdf.com, said.akar@pdf.com

ABSTRACT
In this paper we introduce the application of a new statistical modeling algorithm called *Rule Ensembles* to the problem of yield-loss characterization. Yield loss modeling is viewed as a regression or classification problem, and a model is constructed as a linear combination of simple rules derived from the data. These rule ensembles have been shown to produce predictive models competitive with the best methods. In addition to their high accuracy, however, these rules are easy to understand. Similarly, the degree of relevance of each rule, and its respective variables, can be assessed. The algorithm also provides methodology for automatically identifying those variables that are involved in interactions with other variables, and the strength and degrees of those interactions. To illustrate the interpretation advantages of the method, an analysis on semiconductor manufacturing data is provided.

KEYWORDS: Yield-loss characterization, decision trees, regression, classification, predictive learning, ensembles.

1. INTRODUCTION

The predictive learning problem is stated as follows [1]: one is given a "training" data base of N "observations"

$$D = \{y_i, x_{i1}, x_{i2}, \cdots, x_{in}\}_1^N = \{y_i, \mathbf{x}_i\}$$

where y_i, x_{ij} are measured values of attributes (properties, characteristics) on an object, and D is considered to be a random sample from some joint (population) distribution. The quantity y is called the "output" (or "response") variable, and $\mathbf{x}$ are referred to as the "input" (or "predictor") variables. The goal is to build a functional model

$$\hat{y} = F(x_1, x_2, \cdots, x_n) = F(\mathbf{x})$$

that offers an adequate and interpretable description of how the inputs affect the output.

Interpretation involves gaining an understanding of those particular input variables that are most influential on the response, and the nature of the dependence of the model on those influential inputs.

In the semiconductor domain, the response variable is yield and the predictors correspond to yield management variables of lots or wafers (e.g., tool used at each processing step). To the extent that the model F qualitatively reflects the true relationship between input and output variables, the information provided by the interpretation process can shed light into the factors contributing to yield loss and help prioritize yield improvement efforts.

The remainder of this paper is organized as follows: Section 2 gives a short overview of decision trees, their strengths and limitations, and their variable importance scoring scheme. Section 3 provides a brief description of boosting and tree ensembles. Section 4 introduces the new approach of rule ensembles. Section 5 presents an analysis on real data, and is followed by concluding remarks.

2. DECISION TREE METHODS

Decision Trees [2,3] are a popular non-linear data mining method that has proven useful in the problem of yield-loss characterization, and nowadays are available within yield management software tools (e.g., PDF's *data*Power™). Tree-based methods partition the space of all joint values of the input variables into a set of (hyper) rectangles, and then fit a simple model (like a constant) in each one. Partitions are constructed with a series of straight-line boundaries, as in Figure 1, perpendicular to the axis of the input variable being used.

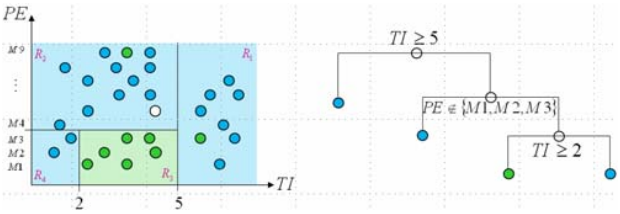


Figure 1: Artificial 2-class data – i.e., $y \in$ {*high_yield*, *low_yield*}, and a partition induced in the 2-dimensional input space by the tree on the right.

If we have a partition into M regions, $R_1, \ldots, R_M$, and the response within each region is a constant c_m, our predictive model has the form $F(\mathbf{x}) = \sum_{m=1}^{M} c_m I(\mathbf{x} \in R_m)$, where $I(\cdot)$ is an indicator of the truth of its argument. Variable interactions are hinted by the presence of multiple variables in the top- (root node) to-bottom (terminal nodes) paths in the tree. This is, however, not a sufficient condition, as different such paths can combine to substantially reduce, or eliminate, the interaction effect as reflected in the overall model.

It is easy to see that any decision tree has an equivalent representation as a set of conjunctive "rules", one for each terminal node. Each rule is given by the product of the indicator functions associated with all the edges on the path from the root to the terminal node (see Table 1).

$$r_1(\mathbf{x}) = I(TI \geq 5)$$
$$r_2(\mathbf{x}) = I(TI \geq 5) \cdot I(PE \notin \{M1, M2, M3\})$$
$$r_3(\mathbf{x}) = I(TI \geq 5) \cdot I(PE \notin \{M1, M2, M3\}) \cdot I(TI \geq 2)$$
$$r_4(\mathbf{x}) = I(TI \geq 5) \cdot I(PE \notin \{M1, M2, M3\}) \cdot I(TI < 2)$$

Table 1: the tree of Figure 1 represented by rules.

Despite their conceptual simplicity, decision trees are very powerful and remain a popular off-the-shelf data mining method. Their appeal can be explained in terms of a number of desirable characteristics. Trees can naturally handle mixed variable types (categorical and continuous) and missing values. Thus, one can easily combine parametric and process equipment data in the yield analysis. They are invariant under monotone transformations of the individual input variables, and thus largely immune to bad input distributions (e.g., outliers). This avoids the need for re-scaling and other data preprocessing operations. They are relatively fast to construct, which allows working with voluminous wafer-, or die-, level data. They produce interpretable – i.e., easy to read, models (see Figure 1). Finally, they perform internal variable subset selection as an integral part of the tree construction procedure. They are thereby resistant to the inclusion of many irrelevant predictor variables [1]. Thus, trees can be effectively used to identify key yield factors from among hundreds, or thousands, of parameters.

Relative Importance of Input Variables

Which input variables are the most important? Obviously, the variables selected in the final tree are deemed the most relevant – i.e., they have a substantial influence on the response. It is often useful, however, to learn the relative importance of those variables that, while not giving the best split of a node, may give the second or third best. Consider the simple data of Table 2: we observe that a single node tree, with the split test $PE - CC = C1$ (see Figure 2), can perfectly partition the data into the *GOOD* and *BAD* groups.

Yield	*PE - BB*	*PE - CC*	*PE - DD*	*PE - EE*
BAD	B2	C1	D1	E1
BAD	B1	C1	D2	E1
BAD	B2	C1	D1	E1
BAD	B1	C1	D2	E1
BAD	B2	C1	D1	E1
GOOD	B1	C2	D2	E1
GOOD	B2	C2	D1	E2
GOOD	B1	C2	D2	E2
GOOD	B2	C2	D1	E2
GOOD	B1	C2	D2	E2

Table 2: Sample yield data with 10 observations and 4 input (categorical) variables.

Variable $PE - EE$ doesn't occur in this tree; yet, if "masking" variable $PE - CC$ is removed and another tree grown, $PE - EE$ would be selected, and the resulting tree would be almost as accurate as the original. In such a situation, we would like the "importance" of $PE - EE$ to be close to that of $PE - CC$.

The theory of decision trees provides a methodology for deriving a variable ranking that allows for the detection of masking variables [2]. The relevance measure is based on the contribution of each variable to error reduction during the process of building the tree. The relevance scores are relative to a given tree – i.e., they might change if the data is modified and a different tree is built, and thus it is customary to assign the largest a value of 100 and then scale the others accordingly (see Figure 2).

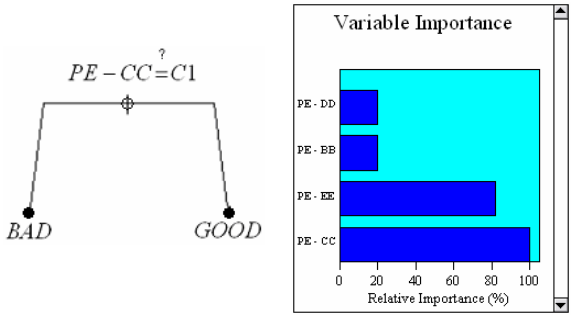


Figure 2: Simple decision tree for the data of Table 2, and corresponding (relative) variable importance scores.

Tree Limitations

Despite their long list of desirable characteristics, trees are not necessarily universally better, and themselves suffer from some limitations. Tree structure can be *unstable*: resort the data a little bit, re-grow the tree, and one might get a second tree very different from the first (although in practice tree instability is often limited to a few of the lower nodes). Trees produce a coarse *piecewise-constant approximation* to the true, but unknown, target function $F^*(\mathbf{x})$ (think of a linear function approximated by a staircase-like function). Trees suffer from *data fragmentation*: as each node splits, the children nodes have less data to work with. This is particularly problematic if our data matrix is very "wide" – i.e., it has many more columns than rows. Finally, trees are biased towards *high interaction order approximations* (because interactions enter the model through root-to-terminal node paths involving more than one input variable). Recently, a new technique has emerged that fixes these limitations at the expense of some interpretability loss: boosted decision trees [4, 5, 6]. Interpretability is then enhanced by replacing the trees with rules.

3. BOOSTED TREES

The basic idea is to combine the outputs from many trees to produce a more robust and accurate "committee," or

"ensemble," decision. More formally, a linear combination of (many) small trees is used instead of just one single tree

$$F(\mathbf{x}) = \sum_{m=1}^{M} a_m T_m(\mathbf{x})$$

where each $T_m(\mathbf{x})$ is a decision tree. Thus, boosting can be understood as an algorithm for fitting an additive expansion in a set of "basis" functions [5, 6]. Additive expansions like this are at the core of other learning methods – e.g., sigmoids in neural networks, wavelets in signal processing, etc.

How do boosted trees fix single tree limitations? The function approximation is still piecewise constant but by combining many more pieces, the approximation is much finer; thus, better approximations to smooth target functions can be produced. By averaging over a large set of small trees, the variance of the estimate produced by a single large tree is reduced. Data fragmentation is no longer an issue as the trees in the model are small (e.g., 6-8 terminal nodes), and every new tree added to the model uses all the data.

Finally, in boosting decision trees the interaction order of the approximation can be controlled by restricting the average size J (number of terminal nodes) of the individual trees included in the model. Setting $J = 2$ results in single split trees ("stumps"), and produces models with only main effects – i.e., no interactions. Setting $J > 2$ permits interactions to order $J - 1$.

Interpretation

Although the simple two-dimensional representation offered by a single tree is no longer available, a new interpretation methodology has been developed for ensemble models [5]. A (relative) *importance score* ranks the input variables according to their relevance (see Figure 3). An *interaction statistic* allows identification of those variables that are involved in interactions with other variables, and the strength and degrees of those interactions (see Figure 4). And *partial dependence plots* allow the study of how the response variable (i.e., yield) changes as a function of the most important variables (see Figure 6).

The ensemble *importance* scoring of input variables is a simple generalization of the measure initially proposed for single trees (the original measure is averaged over the trees in the ensemble). Due to the stabilizing effect of averaging, the generalized relevance measure turns out to be more reliable than its counterpart for a single tree [1].

Once the most important variables have been identified, we can gain insight into the input-output relationship represented by the training data by visualizing the dependence of the approximation $F(\mathbf{x})$ on their joint values. For this purpose, *partial dependence plots* have been proposed. Just like in a trellis plot, where one can picture a two-variable function $F(x_1, x_2)$ by showing the dependence of the function on x_1 conditioned on the respective values of x_2, partial dependence plots are intended to visualize the effect on $F(\mathbf{x})$ of a small subset of the important input variables $\mathbf{x}_S$ after accounting for the (average) effects of the other ("complement") variables $\mathbf{x}_C$ – i.e., $\mathbf{x} = \mathbf{x}_S \cup \mathbf{x}_C$. More formally, the partial dependence on $\mathbf{x}_S$ is defined as $F_S(\mathbf{x}_S) = \mathrm{E}_{\mathbf{x}_C} F(\mathbf{x}_S, \mathbf{x}_C)$ and is estimated by

$$\hat{F}_S(\mathbf{x}_S) = \frac{1}{N} \sum_{i=1}^{N} F(\mathbf{x}_S, x_{iC})$$

where $\{x_{1C}, \ldots, x_{NC}\}$ are the values of $\mathbf{x}_C$ occurring in the training data [1].

Partial dependence plots can be used to help interpret models produced by any "black box" method. In the case of decision trees, however, $\hat{F}_S(\mathbf{x}_S)$ can be rapidly computed from the tree itself without requiring additional passes over the data. For the complete ensemble, the results are averaged over all the trees.

Partial dependence plots can also be used to uncover and study *interaction effects*. An interaction between x_i and x_k is present if the difference in the value of $F(\mathbf{x})$ for different values of x_i depends on the value of x_k. Conversely, if the shape of the dependence function on either variable is unaffected by the value of the other variable, then no interaction is assumed. This notion is formalized in [8] where a statistic for testing whether a specified variable interacts with any other variable is defined

4. RULE ENSEMBLES

The boosting methodology is not restricted to decision tree ensembles. Many types of functions ("base learners") can be combined – e.g., sigmoids, RBFs, etc. Furthermore, base learners from different families could be combined, which can result in increased accuracy when the different families are chosen to complement each other –e.g., trees and linear functions.

For the purpose of interpretation, it is desirable that the ensemble be comprised of "simple" rules such as the ones defining a decision tree – i.e., each base learner is a 0/1-valued function of the form

$$r_m(x) = \prod_j I(x_j \in s_{jm})$$

where each s_j identifies a region of the input space. These rules are often defined by a small number of variables and thus are easy to understand.

Combining the non-linear rules, which allow for the modeling of interaction effects, with purely linear terms, results in a "hybrid" ensemble model

$$F(\mathbf{x}) = a_0 + \sum_{k=1}^{K} a_k r_k(\mathbf{x}) + \sum_{j=1}^{n} b_j x_j \ .$$

The rules can come from many different starting points – e.g., using genetic algorithms or other approximate optimization approaches, but a natural way would be to take advantage of a decision tree ensemble constructed via boosting. Once the set of trees $\{T_m(\mathbf{x})\}_1^M$ has been built, the rule set $\{r_k(\mathbf{x})\}_1^K$ is derived by "decomposing" all trees into their constituent rules [8]. Given such a set of rules, the coefficients for the linear combination are obtained by a regularized linear regression:

$$(\{\hat{a}_k\},\{\hat{b}_j\}) = \underset{\{a_k\},\{b_j\}}{\arg\min} \sum_{i=1}^{N} L\left(y_i,\ a_0 + \sum_{k=1}^{K} a_k r_k(\mathbf{x}_i) + \sum_{j=1}^{n} b_j x_{ij}\right) + \lambda\left(\sum_{k=1}^{K}|a_k| + \sum_{j=1}^{n}|b_j|\right)$$

where the second term is the l_1 (lasso) penalty that "shrinks" the coefficients towards zero [7]. This step helps produce *sparse* models, with less correlated components, that often lead to substantial gains in prediction accuracy.

5. ANALYSIS OF SEMICONDUCTOR DATA

In this section we present the application of rule ensembles, and the associated interpretational tools developed for them in [8], to a yield data set. All experiments were conducted using the "RuleFit" implementation for R and internally developed software.

The data consists of 84 lots (2045 wafers) of limited yield of one memory failing BIST bin, ranging from 60% to 96%, each described by 680 input categorical (process equipment) and continuous (time) variables (missing values are present). The goal is to build a model that can help establish if certain machines (or combination of them) at certain time at certain process steps cause low yield.

Table 3 summarizes the estimated test error results for the lot- and wafer-level data using three methods: ensembles, main effects only model (i.e., using single-variable rules in the ensemble), and a single tree build with CART [2]. For the ensemble model, the number of terms (rules and linear) with nonzero coefficients is also shown. For the single tree, the tree size is given as well.

	Lot-level	*Wafer-level*
Ensemble model (# terms)	3.7 (213)	0.35 (220)
Main effects only model	5.1	0.91
Single tree (# terminal nodes)	5.5 (1)	0.44 (32)

Table 3: Average absolute prediction error (estimated via cross-validation) for different modeling methods applied to the yield data.

The average absolute prediction error for the ensemble model is significantly lower than the corresponding error for an additive model restricted to main effects only. Thus, there is good evidence for interaction effects being present. At the lot-level, where the data matrix is "wide," the best single tree was a stump (one terminal node only). Thus, 5.5 is the error associated with the constant model $\hat{y} = mean(y)$, and the ensemble model's error represents a ~33% improvement. At the wafer-level, the single tree model had 32 terminal nodes and a depth of 7 – i.e., a not so easy to read tree. In this case, the ensemble model improves the single-tree's error by ~ 20%.

Table 4 shows the three globally most important terms in the lot- and wafer-level models. For rules, the coefficient (*Coef*) values represent the change in predicted value if the rule is satisfied (or "fires"). Support (*Supp*) refers to the fraction of the data to which the rules applies. Thus, rule 1 of the lot-level model applies to ~27% of the data, and can be interpreted as saying that lots for which PE.2100.1000.LTH201 is not in {LEQUIP753, LEQUIP755} and PE.3730.1000.LTH205 ≠ LEQUIP702 tend to have higher yield.

Imp	*Coef*	*Supp*	*Lot-level Model Rule*
100	1.88	0.27	PE.2100.1000.LTH201 ∉ {LEQUIP753, LEQUIP755} & PE.3730.1000.LTH205 ∉ {LEQUIP702}
88	-1.49	0.52	PE.2975.1610.WET505 ∉ {CEQUIP702} & PE.3940.1500.RI441 ∈ {EEQUIP704, EEQUIP706}
87	-1.82	0.20	PE.3760.4000.FRN333 ∈ {DEQUIP701, DEQUIP703} & PE.3880.4350.ILTM444 ∈ {YEQUIP702, YEQUIP706, YEQUIP707, YEQUIP709} & PE.4450.4200.CMP561 ∉ { PEQUIP703}
Imp	*Coef*	*Supp*	*Wafer-level Model Rule*
100	1.58	0.25	PE.1550.1000.LTH203 ∈ {LEQUIP754} & PE.3560.4720.ILT112 ∉ {YEQUIP704, YEQUIP706, YEQUIP710, YEQUIP711} & PE.3880.4350.ILTM444 ∉ {YEQUIP702, YEQUIP706, YEQUIP707, YEQUIP709} & TI.1850.1805.WETETCH13 ≤ 38620
89	-1.37	0.27	PE.3300.0400.WETCFF21 ∈ {SEQUIP702} & PE.3670.4200.CMP552 ∉ {PEQUIP702} & TI.3230.2115.INS711 ≥ 38620
71	-1.09	0.29	PE.2100.1175.ION621 ∉ {IEQUIP703} & PE.2450.1040.WETS23 ∉ {CEQUIP704} & PE.3970.4200.CMP554 ∉ {PEQUIP706, PEQUIP707}

Table 4: Top three rules based on global importance for the lot- and wafer-level ensemble models for the yield data.

Similarly, rule 2 of the lot-level model applies to ~52% of the data, and can be interpreted as saying that lots for which PE.2975.1610.WET505 ≠ CEQUIP702 and PE.3940.1500.RI441 is in {EEQUIP704, EEQUIP706} tend to have lower yield.

Figure 3 shows the relative importance of the ten most important input variables (out of the 680 given ones) of the wafer-level model, as averaged over all predictions. Variable PE.3880.4350.ILTM444, which is also present in the top rules, figures prominently. An analysis (not shown) of the partial dependence of yield on this variable, reveals that wafers using YEQUIP707, or YEQUIP709, at step 3880.4350 tend to have noticeably lower-yield.

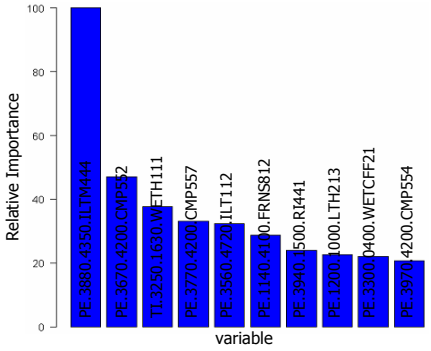


Figure 3: Input variable relative importance (global) for a rule ensemble model built on the yield data.

The analysis of interaction effects can now be focused on the smaller set of variables deemed most relevant. The yellow bars in Figure 4 show the values of the statistic used to test whether a specified variable interacts with any other variable. The red bars correspond to the reference (null) distribution values. Thus, the height of each yellow bar reflects the value of the interaction statistic in excess of its expected value under the null hypothesis of no interaction effects.

The null (reference) distributions are computed using a bootstrap method. The idea is to generate "random" data sets that are similar to the original training data, repeatedly compute the interaction statistic on these artificial data, and compare the resulting values with those obtained from the original data.

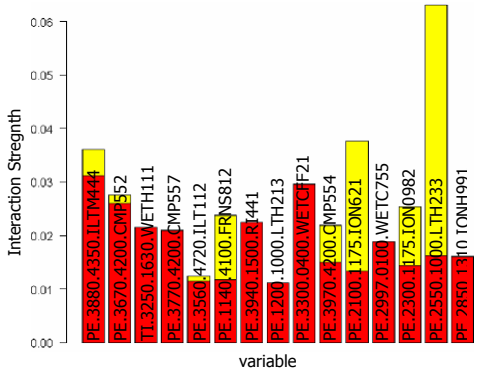


Figure 4: Total interaction strengths in excess of expected null value for top-15 input variables.

Although the strengths of the interaction effects shown in Figure 4 are not large, at least two of the fifteen most influential variables appear to be involved in interactions with other variables. After identifying those variables that interact with others – e.g., PE.2100.1175.ION621 and PE.2550.1000.LTH233 above, we need to determine the particular other variables being interacted with.

The values of the two-variable interaction strength statistic for PE.2550.1000.LTH233 are shown in Figure 5. Here one sees that PE.2550.1000.LTH233 dominantly interacts with PE.2997.0100.WETC755.

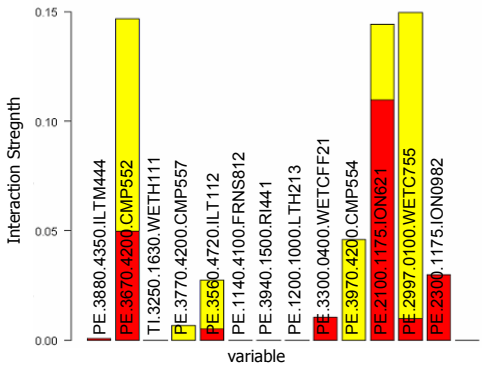


Figure 5: Two-variable interaction strengths of variables interacting with PE.2550.1000.LTH233.

The detailed nature of the PE.2550.1000.LTH233 interaction with PE.2997.0100.WETC755 can be further explored with the corresponding partial dependence plot (see Figure 6). In the absence of an interaction between these variables, all PE.2550.1000.LTH233 partial dependence distributions, conditioned on different values of PE.2997.0100.WETC755, would be the same.

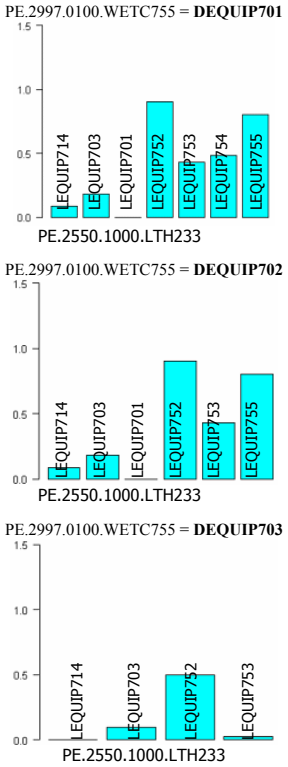
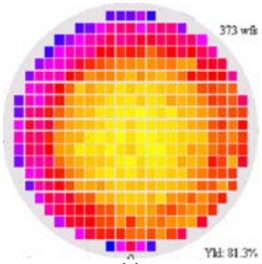


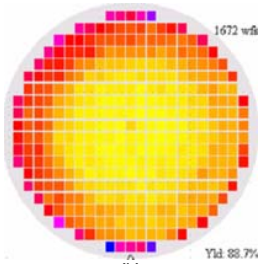
Figure 6: Joint partial dependence on variables PE.2550.1000.LTH233 (found earlier to be a relevant input) and PE.2997.0100.WETC755.

Here one sees similar distributions when PE.2997.0100.WETC755 takes the values DEQUIP701 and DEQUIP702 (with one PE.2550.1000.LTH233 value not represented in one case). The distribution for PE.2997.0100.WETC755 = DEQUIP703 is fairly different from the others, suggesting it captures the essence of the interaction effect between these two variables: yield is lower throughout in this case.

We can further visualize the above interaction using *wafer*MAP plots (see Figure 7). Figure 7(a) shows average yield for the wafers using LEQUIP701 at step 2550.1000 or using DEQUIP703 at step 2997.0100. Figure 7(b) shows average yield for the "complement" set of wafers. A ~7.5% yield loss is observed.



(a)



(b)

Figure 7: Wafer map representation of yield. Minimum values are represented by darker colored die, graduating to lighter die for higher values. In (a), average yield for the 373 wafers using LEQUIP701 at step 2550.1000 or using DEQUIP703 at step 2997.0100. In (b), average yield for the remaining 1672 wafers.

6. SUMMARY

Ensemble methods in general, and boosted decision trees in particular, constitute one of the most important advances in machine learning in recent years. In the absence of detailed a priori knowledge of the problem at hand, they provide superior performance. A number of interpretation tools have been developed that, although applicable to other algorithms, often are easier to compute for trees. With the introduction of rule ensembles, the interpretability of the ensemble model has been further significantly improved. Together they provide a solid methodology that is applicable to yield loss characterization.

REFERENCES

1. Hastie, T., Tibshirani, R. and Friedman, J. *The Elements of Statistical Learning – Data Mining, Inference, and Prediction*. Springer, 2001.

2. Breiman, L., Friedman, J. H. and Olshen, R. A., Stone, C. J. *Classification and Regression Trees*. CRC Press, 1984.

3. Quinlan, J. R. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1987.

4. Freund, Y. and Schapire, R. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference* (San Francisco), Morgan Kauffman, 1996.

5. Friedman, J. Greedy function approximation: the gradient boosting machine, *Annals of Statistics,* 29 (2001)*,* 1189–1232.

6. Friedman, J. *Stochastic gradient boosting*. Technical Report, Stanford University, 1999.

7. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. Royal Statistics Society B.,* 58 (1996), 267-288.

8. Friedman, J. and Popescu, B. E. *Predictive learning via rule ensembles*. Technical Report, Stanford University, 2005.

BIOGRAPHY

Giovanni Seni is currently a Data Mining Engineer for PDF Solutions, and Adjunct Faculty at the Computer Engineering Department of Santa Clara University, where he teaches a Pattern Recognition and Data Mining class. Giovanni obtained an MS and PhD in Computer Science from State University of New York (SUNY) at Buffalo. Prior to joining PDF, Dr. Seni was a Member of the Technical Staff at Motorola Labs

Said Akar has more than twelve years experience in software design, programming and project management. He is one of the key designers/developers of PDF's dataPOWER Yield Management System. He has also performed extensive research in neural networks and image processing. Dr. Akar earned his B.E. in Electrical Engineering from the American University of Beirut, and his M.S. in Applied Mathematics and Ph.D. in Electrical Engineering from Florida Institute of Technology.

Edward Yang is a member of the YMS Applications Engineering group at PDF Solutions. Before joining PDF Solutions, he had experience with Philips Semiconductors as CAD engineer, Product engineer and Yield engineer. Mr. Yang received an M.S. degree from San Jose State University and a B.S. degree from National Chiao Tung University in Taiwan