NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Outlier Detection in Lithography

Benchmarking and the Creation of a New Algorithm

Monika Rose Brown

Internship Report presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Outlier Detection in Lithography

Benchmarking and the Creation of a New Algorithm

Monika Rose Brown

Internship Report presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

Advisor: Fernando José Ferreira Lucas Bação

External Supervisor*:* Alexei Nazarian, Senior Architect, ASML

February 2020

ACKNOWLEDGEMENTS

I want to thank Professor Fernando Bação for his guidance on this project - but most of all, for being the instructor that took an interest in my work and future and offered me encouragement throughout the Master's program and my internship.

A huge "thank you" to my mentor at ASML, Alexei Nazarian, for seeing me through the internship and helping me understand when I often did not. Your sense of humor and patience were greatly appreciated.

I'd like to also thank the whole ORC team at ASML. I've never met a group of such smart, humble people before. You guys made a big impression on me. Thank you all for being so welcoming.

Thank you to the staff at Nova IMS. I'm sure your team fields the same questions over and over again, but thank you for being there when I came with those same questions.

I'd like to give my thanks and gratitude to Carla. This adventure has been a long haul. Thank you for giving up the good life in California to endure some rough times with me to make it to this point. You put a lot of work into getting me here, so this is your achievement too. Thank you.

Radar, thank you for holding on so long and almost getting me to the finish line. I'm sorry I didn't get it done in time for us to go celebrate and have more time together. Thank you for being my study buddy all of those long nights and reminding me to go out for walks.

ABSTRACT

Overlay is the measure of a lithography machine's success at printing layers accurately atop one another. Poor overlay in semiconductor manufacturing can cause problems in a semiconductor chip's electrical current and, in the worst case, can render the chips unusable. A new algorithm is proposed for detecting outliers in lithography calibration wafers using an ensemble of well-known statistical methods. Detecting outliers on these calibration wafers and removing them from the data fed to the lithography machine contributes to improving overlay accuracy for semiconductor manufacturers as these wafers are used to set proper alignment on a lithography machine. Through simulation, the new ensemble algorithm along with other outlier algorithms are benchmarked against a lithography firm's current outlier detection algorithm. The new ensemble algorithm shows outperformance on every criterion tested.

Keywords: *outlier detection, anomaly detection, lithography, ensemble algorithm*

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

ABBREVIATIONS & ACRONYMS

ABOD	Angle-based Outlier Degree
DBSCAN	Density-based Spatial Clustering of Applications with Noise
FOUP	Front Opening Universal Pod
LOF	Local Outlier Factor
MAD	Median Absolute Deviation
ORC	Overlay Reference and Control
REC	Reticle Error Correction
RSS	Residual Sum of Squares
WEC	Wafer Error Correction

Chapter 1 | Introduction

This chapter examines the importance of overlay and its challenges for the semiconductor industry. The role of the lithography machine is described along with its connection to the semiconductor industry and an introduction to the leader in the lithography industry is given. Finally, the objective of the research is outlined.

1.1 Industry

Moore's law observes that the number of transistors on a microchip doubles every two years, leading to processing power doubling, and the cost of transistors halving. Unfortunately, this phenomenon does not happen on its own and that leads to Moore's second law: the capital cost of a semiconductor fabrication plant increases exponentially over time.

The most expensive tool in the semiconductor fab is the lithography machine with prices ranging from $10 million to over $100 million per machine, depending on the sophistication of the chips produced. These are the machines that the semiconductor industry relies on to enable chips to become smaller, faster and cheaper. Lithography machines transfer circuit patterns onto silicon wafers to make chips for everything from refrigerators and cash cards to cell phones and supercomputers. The integrated circuits are transferred onto a silicon wafer by building up the circuit through a series of layers using photoresist.

Close alignment between the layers is critical to the integrated circuits working properly. A measurement of how well each successive layer aligns with the previous is called "overlay" and good overlay allows companies to build smaller transistors with higher memory density and faster transistors. A visualization of overlay can be seen in Figure 1.1. Overlay error specification between wafer layers is now below 3 nanometers. A nanometer is one billionth of a meter. To put that in perspective, a page of a book is 100,000 nanometers thick, a red blood cell is 7,000 nanometers wide and human DNA is 2 nm in diameter. Therefore, deviation in alignment between layers is little more than the diameter of a DNA molecule.

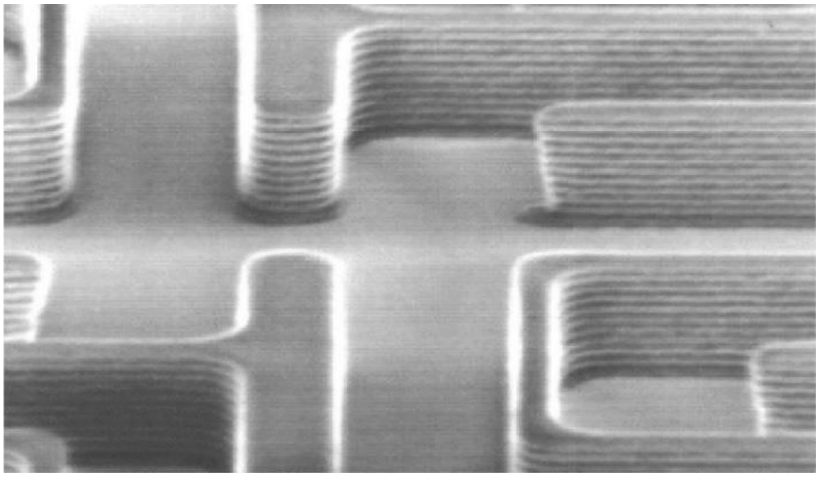


Figure 1.1 – Overlay of Layers

(Mack, C.M. 2006)

Lithography machines are complex and contain many moving parts and sensors to measure output. Vibrations, heat, and a buildup of contamination in the machines result in “drift.” Drift refers to the output of the machine that is further and further from the original pattern that is being replicated onto the silicon wafer and causes the quality of overlay to deteriorate. The factors that cause drift can result in the printing of a mark on a wafer that is significantly different from marks printed on other wafers in the corresponding position. Such a mark is considered an outlier. This report focuses on outlier detection algorithms at a unit within a company producing lithography machines and playing a critical role in making Moore’s Law a reality.

1.2 Company

ASML, headquartered in Veldhoven, Netherlands, is the world’s largest maker of lithography machines for the semiconductor industry with an 85.4% market share by revenue (Clarke, 2018). While ASML might not be a household name since it does not make a consumer product, the average European wears or carries more than 10 integrated circuits and the majority of these will have been produced on an ASML system.

In 2019, ASML had net sales of 11.8 billion Euros and an R&D spend of 2.0 billion Euros, which helps to support its more than 12,000 patents worldwide. It employs more than 24,000 people and has offices in 60 cities in 16 countries (ASML: ASML at a Glance, 2019). As one of Europe's largest blue-chip companies, ASML is a member of the Euro STOXX 50 stock market index, an index providing "a blue-chip representation of Supersector leaders in the Eurozone."

Within ASML's Development & Engineering unit (the company's equivalent to Research & Development) lies the Overlay division. The Overlay Reference and Control group (ORC) is a team of roughly 15 people within this division that run experiments and create tools to measure the overlay performance between wafer layers. Additionally, the ORC team manufactures wafers with a series of marks printed on them that enable customers; such as, Intel and TSMC, to recalibrate their lithography machines. The ORC team is further split into two parts: Reticle Error Correction (REC) and Wafer Error Correction (WEC) and each of these groups use separate outlier detection methods. The internship work was conducted with the Wafer Error Correction (WEC) group.

1.3 Problem and objective

An outlier is defined as an observation that appears to deviate markedly from other members of the sample in which it occurs. The internship project focused on finding outliers on the wafers created for lithography machine recalibration. In context of the project, an outlier is a mark etched onto a wafer that when compared to a reference wafer, known as a "golden wafer", deviates in position from the reference wafer greater than expected. That expectation is based on deviations seen on other wafers and on the wafer itself.

Occasionally, wafers showing very little difference with the golden wafer may be kept by the firm to use as a golden wafer in the future as the wafers wear out with use. Each calibration wafer must meet all of the company's specifications in order to qualify it to be sold to clients for lithography machine recalibration. Of particular interest to ASML was a comparison of how other algorithms performed in outlier detection on the firm's wafer data in comparison to the WEC team's existing outlier detection algorithm. Experiments running simulations on the data with various outlier detection algorithms were conducted and benchmarked to the performance of the WEC team's current outlier detection algorithm.

Any outliers appearing on golden wafers will be transferred to the new wafers being manufactured and reduce the precision of machine calibrations. This is another reason that outliers need to be identified and eliminated. Often in statistics there is a strong case for modifying or dampening the effect of an outlier rather than removing it, but in this case any outlier found is considered to be an error, so it is removed.

Outlier detection is important enough to ASML that although the ORC's WEC team already has an outlier detection algorithm with an average of 99.75% accuracy, 60.50% precision and 90.5% recall as measured in runs of 100 simulations at a time, they created a project for an intern to investigate new outlier detection algorithms and benchmark these algorithms against the firm's current outlier detection algorithm.

Over the course of the 6-month internship, a benchmarking platform was built that created simulations of outliers in wafers and measured the performance of the various outlier detection algorithms in finding those outliers. The performance of the algorithms was measured (benchmarked) against to the firm's current algorithm. In addition to the initial research question, "How do other outlier detection algorithms perform on the firm's wafer data in comparison to the currently used algorithm?," the project went on to also investigate whether a superior outlier detection algorithm could be custom designed for the firm using what was learned in the benchmarking process.

1.4 Roadmap

Chapter 2 will review existing research in outlier detection and then summarize several outlier detection algorithms. Chapter 3 explains the structure of the data and how the benchmarking was performed. The chapter then concludes with a detailed explanation of the customized algorithm for outlier detection that was built using what was learned in the benchmarking process. Chapter 4 presents the results of the simulations, while chapter 5 discusses the relevance and insights of the results and makes recommendations for future work. Finally, in Chapter 6, conclusions are given.

Chapter 2 | Background

An overview of the development of research in outlier detection and ensemble models for outlier detection is provided in this chapter. It is followed by a description of the algorithms that were either tested or used in the benchmarking process. Please note that the Appendix supplies numerical examples of the calculations for a few of the algorithms.

2.1 Literature Review

Research on outlier detection methods is an active field that has been propelled by the growth of data as computational power increases. Therefore, the advancement in outlier techniques can be indirectly tied to advancements in lithography and the semiconductor field.

Outlier detection methods are developing in complexity as data itself becomes more complex and higher dimensional. Early detection measures for outliers were limited to small, univariate datasets. Examples of such univariate methods include Peirce's criterion (Peirce, 1852) and Chauvenet's criterion (Chauvenet, 1863). A century later, Dixon's Q Test (Dixon, 1951) and the Grubbs Test (Grubbs, 1969) were still focused on small, univariate datasets.

1970 was the year that Intel unveiled the first DRAM chip. As personal computers hit the market in the mid-1970's, it became easier to work with large multivariate datasets. The trend for outlier research also moved to multivariate datasets and using algorithmic data mining approaches to identify outliers. Cook's Distance, a method to show the influence of data points in a least-squares regression, was developed during this time (Cook, 1977).

In their survey of outlier detection algorithms, Hodge and Austin break outlier detection algorithms into the following categories: statistical models, neural networks, machine learning, and ensemble systems (Hodge, Austin, 2004). Ensemble systems contain at least two algorithms from the previous categories and have received more attention in the last twenty years. Walczak and Massart studied combinations of outlier diagnostics and found that the combinations yielded better identification of outliers (Walczak, Massart, 1998). Aggarwal more formally focused on ensemble analysis and sought to categorize the types of ensembles given the outlier problems they solve (Aggarwal, 2013). Aggarwal's research also examines the impact of the types of combinations of models and concludes that the emerging area of ensemble analysis for outlier detection holds opportunities for future research in improving the quality of outlier detection algorithms. A research

paper following up on Aggarwal's findings states, "Surprisingly, for outlier detection there have not been many attempts to use ensemble techniques in a principled way, let alone investigations of the theoretical basis of doing so." (Zimek, Campello, Sander, 2014, p. 12)

In support of the comment above by Zimek, Campello, and Sander; while a number of research papers can be found using machine learning and neural network techniques for outlier detection in the semiconductor industry, only one paper could be found citing an ensemble model for outlier detection (Susto, Beghi, McLoone, 2017). No research could be identified using an ensemble model for outlier detection in the photolithography industry, but one such paper for the print lithography industry was uncovered (Englund, Verikas, 2005).

2.2 An Overview of Outlier Detection Algorithms

Below are summaries of the outlier detection methods tested during the benchmarking process, including those incorporated in the ensemble outlier detection algorithm that was eventually developed. Some methods, like Cook's Distance and Order Statistics, are not typically referred to as outlier detection algorithms but the term "outlier detection algorithm" will be used throughout this report for convenience to refer to each of the methods used.

2.2.1 Statistical Model-Based Methods

Statistical model-based methods suffer from two issues in particular: many are unsuitable for multidimensional datasets and most are distribution-based. However, for datasets with characteristics suited to statistical model-based methods these algorithms are easy to implement into existing code, transparent and tend to be computationally fast.

2.2.1.1 The Three-Sigma Rule

The three-sigma rule is based on a normal distribution and states that the probability of an observation falling within 3 standard deviations of the mean is about 99.73%.

$$Pr(\mu - 3\sigma \leq observation \leq \mu + 3\sigma) \approx 99.73\% \quad (1)$$

Outliers are identified as observations that fall beyond 3 standard deviations in either direction from the mean of the distribution. As can be seen in the equation above, the three-sigma rule is calculated using the mean and standard deviation and both of these measures are sensitive to outliers. This sensitivity to outliers creates the potential for two problems, masking and swamping.

Masking occurs when an outlier is not detected due to a larger outlier skewing the mean and standard deviation so much toward it that other outliers closer to the mean end up lying within the prescribed standard deviation when the larger outlier is present. Only when the larger outlier is removed do the outliers closer to the mean appear as outliers. Swamping occurs when an observation that is not an outlier becomes classified as an outlier due to a large outlier skewing the mean and standard deviation toward itself and away from the non-outlier so that the non-outlier falls outside of the standard deviation and is considered an outlier. Figure 2.1 illustrates both masking and swamping.

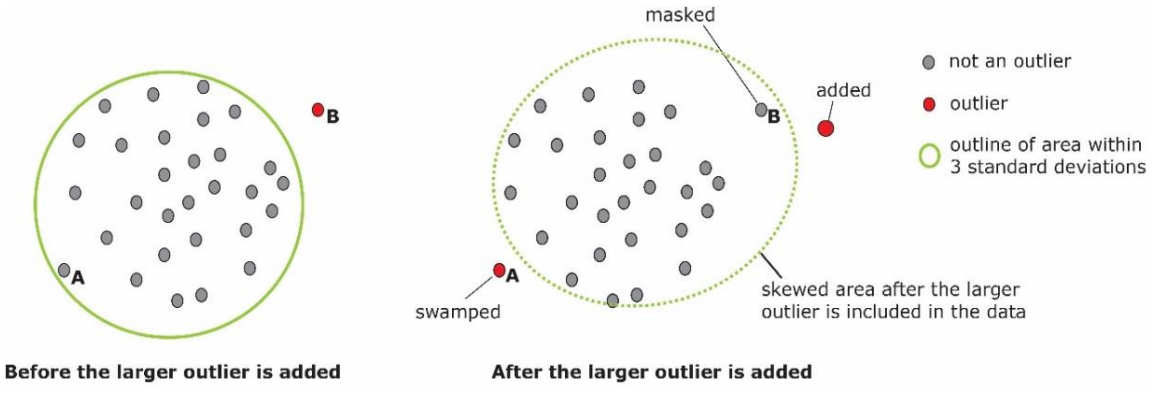


Figure 2.1 – Masking and Swamping

2.2.1.2 Boxplots

Boxplots were first introduced by Mary Eleanor Spear (Spear, 1952) and further enhanced by John W. Tukey (Tukey, 1977) and can be used for analyzing outliers in univariate and multivariate datasets. A data point is considered to be an outlier if it exceeds the points specified in the equation below. As seen in Figure 2.2, Q_1 represents the first quartile and Q_3 represents the third quartile in the equation, while *IQR* represents the interquartile range. Notably, the values $Q_1 - (1.5 * IQR)$ and $Q_3 + (1.5 * IQR)$ fall close to 3 standard deviations on a normally distributed dataset.

$$x_i \text{ is an outlier if: } \quad x_i > Q_3 + 1.5 * IQR \quad \text{or} \quad x_i < Q_1 - 1.5 * IQR \qquad (2)$$

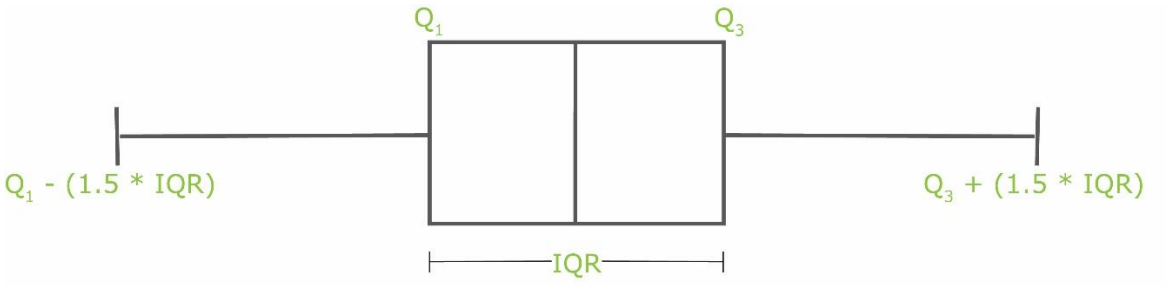


Figure 2.2 – Boxplot

2.2.1.3 Cerioli outlier detection

This outlier detection method uses the Minimum Covariance Determinant (MCD) estimator (Rousseeuw, 1985). The MCD is "a highly robust estimator of multivariate location and scatter" that can resist the masking effect (Hubert, Debruyne, Rousseeuw, 2018, Abstract). In other words, it can transform the data while keeping the same ratio of distances between datapoints and the same symmetry without being influenced by outliers. The Cerioli outlier detection algorithm uses a reweighted version of the MCD to build a distribution of the squared Mahalanobis distance for each observation and iterates through sequences of the weighted distances to calculate an approximate distribution (Cerioli, 2010). Cut-off values are then determined for the distribution and outliers are points beyond the cutoff.

2.2.1.4 Cook's Distance

Cook's distance is a method used with Ordinary Least Squares (OLS) regression to give a measure of an observation's influence on the computation of the linear regression (Cook, 1977). As illustrated on the y-axis of the right-hand side of Figure 2.3, Cook's distance shows the scaled change in the fitted values for each observation ("index") in the x variable. It is computed by deleting one observation at a time and refitting the regression on the remaining (n-1) observations to determine how much the fitted values change when the *i*th observation is deleted.

$$D_i = \frac{\sum_{j=1}^{n}\left(\hat{y}_j - \hat{y}_{j(i)}\right)^2}{pMSE} \quad (3)$$

where

- $\hat{y}_j$ is the jth fitted response value
- $\hat{y}_{j(i)}$ is the jth fitted response value, where the fit does not include observation i
- *MSE* is the mean squared error
- *p* is the number of coefficients in the regression model

While no strict rule exists for a cut-off threshold in Cook's distance, a rule of thumb uses 4/*number of observations* to determine the threshold. An observation with a Cook's distance above the cut-off threshold is considered an influential point, and therefore, an outlier.

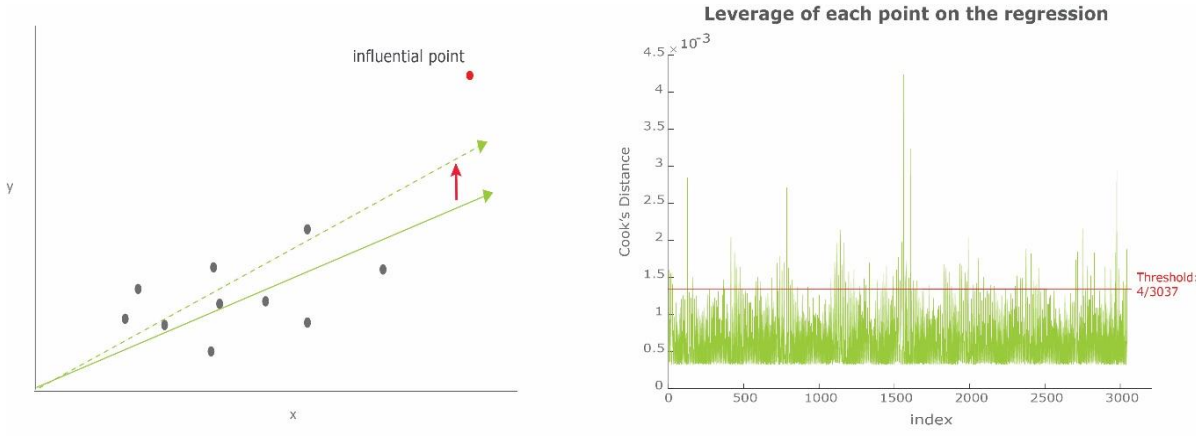


Figure 2.3 – Cook's Distance

The figure on the left shows the shift in the regression line caused by the influential point. The figure on the right shows the scaled change in the fitted values for each observation and the threshold for outliers.

2.2.1.5 Grubbs' Test

The Grubbs' test (Grubbs, 1969) calculates a value similar to the Z-score

$$G = \frac{\max_{n=1,\ldots,N} |x_n - \bar{x}|}{s} \quad (4)$$

where

- $\bar{x}$ is the sample mean, and s is the sample standard deviation

for each observation in a dataset. Then, one-at-a-time starting with the largest calculated value, the calculated values are compared with a critical value. The critical value will depend on the number of observations in the dataset and the significance level chosen, generally 5% or 1%. If a calculated value exceeds the critical value, that observation is considered an outlier. When an outlier is found in the dataset, it is deleted and the Grubbs' test is recalculated for the remaining dataset. This process continues until no outliers are found. The dataset should include more than six observations and a normal distribution is assumed.

2.2.1.6 Mahalanobis Distance

Mahalanobis distance was created by P.C. Mahalanobis in 1936 (Mahalanobis, 1936). The method calculates a matrix of distances from their average, transposes that matrix, multiplies it times the inverse of the covariance matrix and multiplies that result by the distances from their average again. Finally, the square root is taken. A calculated example of the Mahalanobis distance is provided in the Appendix.

$$\text{Mahalanobis Distance} = D = \sqrt{(x_i - \bar{a})^T * S^{-1} * (x_i - \bar{a})} \quad (5)$$

It differs from Euclidean distance in that it takes correlations into account. Once the Mahalanobis distance is calculated, the result is compared to the chi-squared table. A value greater than the corresponding number in the table indicates an outlier. Weaknesses of the Mahalanobis rule include: it assumes a multivariate normal distribution; it assumes the outliers will deviate strongly from the distribution and it is sensitive to the outliers.

2.2.1.7 Median Absolute Deviation (MAD)

The Median Absolute Deviation (MAD) Rule has the advantage of not being sensitive to outliers since it uses the median – unlike some statistical methods that use mean and/or the standard deviation. However, it can be aggressive and find 'too many' outliers (Gauss, 1816).

To calculate the MAD, first the median value of a dataset is computed, then each observation's absolute difference from the median is calculated and the median of these resulting values is chosen. This results in the MAD value.

$$MAD = median(\,|\, x_i - median(x) \,|\,) \quad (6)$$

Then the computed deviation from the median is divided by the MAD value.

$$MAD_{adj} = \frac{(\,|\, x_i - median(x) \,|\,)}{MAD} \quad (7)$$

Next, a designated cutoff value, such as 3*MAD, is chosen. If the resulting MAD_{adj} value exceeds the designated cutoff value, then the observation is considered an outlier.

2.2.1.8 Order Statistics

Order statistics first sorts observations in ascending order of value. Once ordered, the difference between the values of each sequential observations is taken. These steps can be seen in Figure 2.4. The distribution of the differences is then found. The resulting distribution of the differences will only contain positive numbers as subtracting values in ascending order will always yield a value of zero or greater. With the distribution for the differenced data determined, a cutoff threshold can then be established. This threshold identifies a sequential step that is too large – i.e. subtracting the difference in values of those steps would yield a difference beyond the cutoff threshold. Any difference between sequential observations that is above the cutoff value for the distribution is considered an outlier (Arnold, Balakrishnan, Nagaraja, 1992).

Figure 2.4 – Order Statistics

On the left are the steps in order statistics while the right shows the resulting distribution of the differences. If the cutoff threshold was at 13, then the move from the first step of 11 to 25 would indicate an outlier.

2.2.1.9 Weighted Mean

As shown in equation 8, the weighted mean is derived by assigning different weights to observations and taking the summation of each observation's value multiplied by its assigned weight and then dividing by the sum of the weights.

$$\bar{X} = \frac{\sum_{i=1}^{n}(x_i * w_i)}{\sum_{i=1}^{n} w_i} \quad (8)$$

The WEC team's current outlier detection method is named, "weighted mean," but uses a modified version of the weighted mean, as shown in equation 9. This version of the weighted mean sorts the values of the observations in ascending order and takes the average of each value and the value that follows it. This average is then divided by the difference of the same two values. Finally, a summation of these ratios is divided by a summation of the inverse of the differences of the values. An example of the calculation is in the Appendix.

Dividing the averages by the distance between them dampens the impact of any outliers and yields a value closer to the median than the average. Such methods of reducing the weightings of outliers are commonly used in robust statistics, methods that seek to be resistant to outliers. Median and the previously discussed Median Absolute Deviation (MAD) are both robust estimators (Rousseeuw, Croux, 1993).

$$WM(m) = \frac{\sum_{i=1}^{n-1}\left(\frac{(m_i+m_{i+1})/2}{|m_i-m_{i+1}|}\right)}{\sum_{i=1}^{n-1}\left(\frac{1}{|m_i-m_{i+1}|}\right)} \quad (9)$$

At ASML, the WEC team's weighted mean outlier detection algorithm also builds in a tolerance around its cutoff value before declaring an outlier. That tolerance is based on the firm's findings using its historical data and will not be shown here to protect the firm's proprietary methods.

2.2.2 Clustering Methods

The advantage of clustering algorithms is that they are relatively fast compared to distance-based models. Another advantage is that via clustering, insights about local distributions can be provided and they are efficient when there is a great deal of information to summarize, but the granularity of a dataset is lost. Another downside is that due to the randomization, there can be high variability in the results between executions of the algorithms.

2.2.2.1 K-means Clustering

K-means clustering "has been proposed by several scientists in different forms and under different assumptions." (Bock, 2008, pg. 2). The number of clusters, *k*, must be predetermined and the goal of the algorithm is to partition the data points into *k* groups with no data point being a member of more than one group. For the first step, *k* random points in the dataset are chosen to be the centroids, or geometric centers, of the clusters. Next, each data point is assigned to the centroid nearest to it based on its Euclidean distance to the centroid. After each data point is assigned to a centroid, for each of the *k* clusters the centroid is recalculated for its given data points. This will likely result in a new position for the centroid and the previous step of assigning each data point to the nearest centroid is repeated. The assigning of data points to the nearest centroid and recalculation of the centroid is repeated as many times as preset for the algorithm or, if there is no preset number of iterations, then it is repeated until the centroid no longer changes position or the data points no longer move from one cluster to another. The result of this process is to minimize the residual sum of squares (RSS), this is the sum of the squared distances for each data point from its respective centroid.

The random choice of the centroids causes variability in the results. For this reason, the algorithm is generally run repeatedly and the best results are chosen. The best results will be the run with the minimum RSS. If the clustering is done with a training set known to not contain outliers, then detecting outliers in a test set is simply a matter of comparing the maximum distance to the centroid in the training set for the relevant cluster with the distance of the test set's points to the centroid. If a distance exceeds the maximum distance in the training set, then that point can be considered an outlier. If there is no training set without outliers to work with, then outliers can be determined by setting a threshold for the data points within the cluster, such as the 95th percentile. In such a case, all data points with distances to the centroid that fall within the 95th percentile of distances are considered normal and the remaining data points with the longest distances are considered outliers.

2.2.2.2 DBSCAN

The density-based spatial clustering of applications with noise (DBSCAN) method is a popular method in this category (Ester, Kriegel, Sander, Xu, 1996). The user must choose the parameters k (MinPts) and Eps (the distance measurement that will form a radius about the point to determine the neighborhood). The Eps-neighborhood of a point is denoted by $N_{Eps}(x)$ and defined by:

$$N_{Eps}(x) = \{\mathrm{q}\epsilon D | dis(x,q) \leq Eps\} \quad (10)$$

The algorithm will randomly initialize a point *x*, and *x* will be considered a core point if at least k (MinPts) points are directly reachable from *x*. This will be repeated for the next data point until all points have been evaluated and core points identified. If a non-core border point (a point that cannot reach k other points) is still within the Eps of a cluster, it is assigned to that cluster. The remaining non-core points that are not within a neighborhood are assigned as noise (i.e. outliers). Figure 2.5 illustrates the algorithm. The advantages of DBSCAN are: its speed, its robustness to outliers, and its ability to find clusters of various shapes and sizes (for this reason, it can find some results that cannot be found with K-means). However, despite its speed, it has trouble when the clusters have widely varying densities and is less effective in high dimensional spaces where density is harder to define.

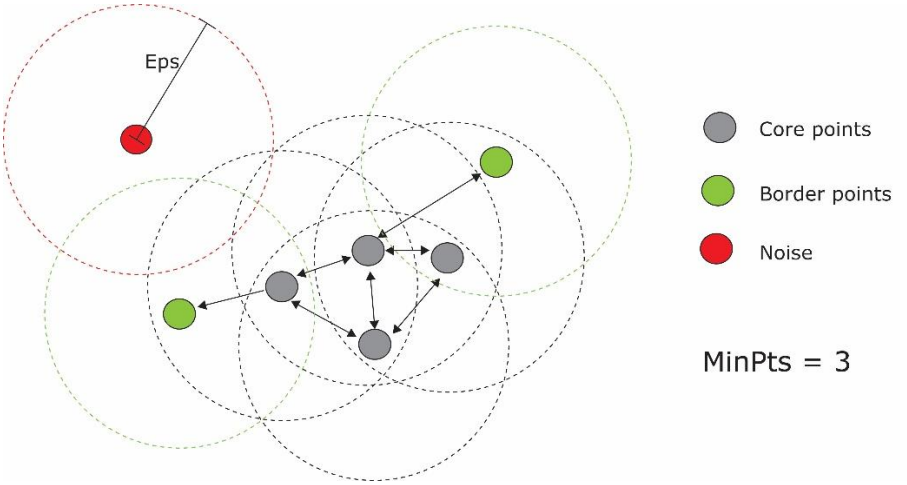


Figure 2.5 – DBSCAN

The circles represent the neighborhood formed around each point by the radius, Eps. The grey core points each have neighborhoods that reach at least the MinPts. The green border points are within the neighborhood of a core point but border point's own neighborhood includes fewer than the MinPts. The red noise point, an outlier, lies outside of the core points' neighborhoods.

2.2.3 A Density Method

Density-based outlier detection methods refer to the proximity of neighbors around a data point. Outliers will display lower densities of neighbors than other data points in its neighborhood.

2.2.3.1 Local Outlier Factor (LOF)

Local outlier factor (LOF) is a popular data mining approach to outlier detection and is a density-based approach. An advantage of LOF is that it accounts for local levels of skews and abnormalities. That said, with the data sparsity in high dimensional spaces, the idea of locality becomes hard to define (Breunig, Kriegel, Ng, Sander, 2000).

The method compares the density around a point with the density of its local neighbors. Points that have lower densities than their neighbors are determined to be outliers. In order to calculate the density, first a *k*th-distance is found by calculating the distance between a point and its *k*th nearest neighbor. The local reachability density of an object is the inverse of the average reachability distance based on the MinPts nearest neighbors of *x*. Figure 2.6 shows the k-distance. A numerical example is provided in the Appendix.

MinPts = k (example below) = number of nearest neighbors used to define the local neighborhood.

$$\textbf{k\_distance}(x) = \{\, q \in D \setminus \{x\} \mid d(x,q) \leq k_{distance(x)}\} \tag{11}$$

$$\textbf{Reachabilty Distance} = reach\_dist_k(x,o) = \max\{k_{distance(o)}, d(x,o)\} \tag{12}$$

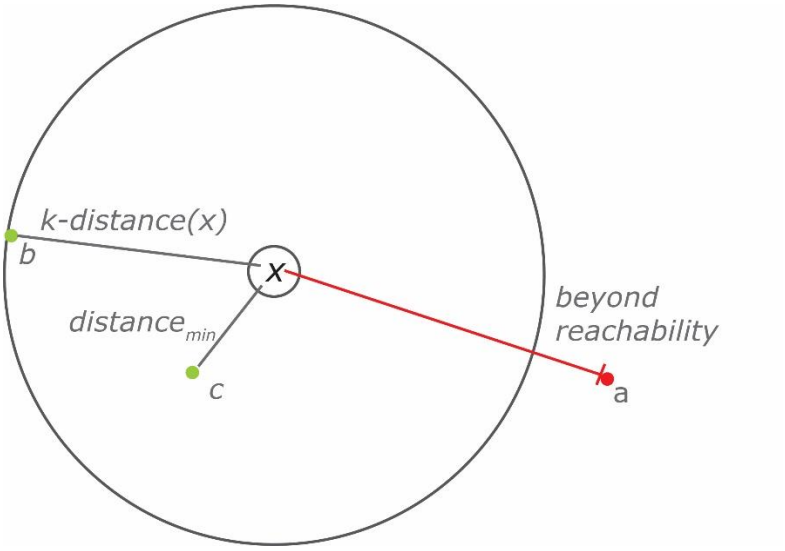


Figure 2.6 – LOF

An illustration of k_distance(x) and reachability distance. Objects b and c have the same reachability distance when k =2. a is not a nearest neighbor.

Local Reachability Density of object $x =$

$$lrd_{MinPts^{(x)}} = 1 /[\frac{o \in N \sum_{MinPts(x)} reach-dist_{MinPts(x,o)}}{\left|N_{MinPts^{(x)}}\right|}] \quad (13)$$

$$= \quad [\frac{\left|N_{MinPts^{(x)}}\right|}{o \in N \sum_{MinPts(x)} reach-dist_{MinPts(x,o)}}]$$

$$\textbf{Local Outlier Factor} = LOF_{MinPts^{(x)}} = \frac{\sum_{o \in N\ MinPts(x)} \frac{lrd_{MinPts(o)}}{lrd_{MinPts(x)}}}{\left|N_{MinPts^{(x)}}\right|} \quad (14)$$

LOF ≈ 1 is considered to be in a cluster, while LOF > 1 is considered to be an outlier.

2.2.4 An Isolation-Based Method

2.2.4.1 Isolation Forest

The Isolation Forest algorithm works by creating partitions that separate, or isolate, outliers (Liu, Ting, Zhou, 2012). Examples of the partitioning can be seen in Figure 2.7. Observations are randomly selected and a split value between the minimum and maximum of that feature (dimension) is randomly selected. Decision trees (isolation trees) are created to determine the partitions and an anomaly score is calculated based on how many partitions are required to separate the observation. The length of the path of the tree from the root to the termination node is equivalent to the number of partitions. Random subsampling in the creation of the decision trees is used and the partitioning process is repeated over and over to calculate an average anomaly score. The Isolation Forest algorithm is based on the concept that outliers should require fewer partitions to isolation them than normal observations require for isolation. An advantage of the Isolation Forest algorithm is that it does not rely on a distance or a density calculation.

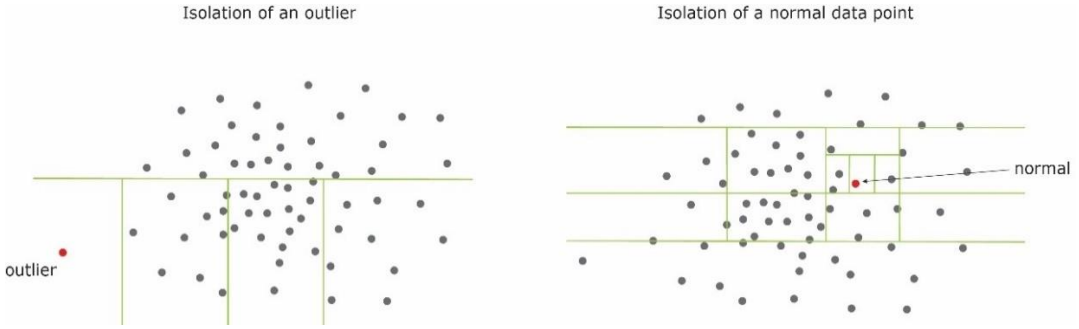


Figure 2.7 – Isolation Forest

2.2.5 An Angle-Based Method

2.2.5.1 Angle-based outlier degree (ABOD)

This method examines the angle from a data point to every pair of points in the dataset. The more isolated points will require a smaller overall angle to encompass the entire dataset while data points in the interior of the data will need a wider set of angles to form angles between every pair of points in the data (Kriegel, Schubert, Zimek, 2008).

Outliers are identified by their substantially smaller angles compared to other points in the dataset, as shown in Figure 2.8. The advantages of this method are that it doesn't rely on any particular parameter and it isn't based on any assessment of distance. However, the time-complexity of the algorithm makes it computationally inefficient and there is no way to define a precise threshold for outliers.

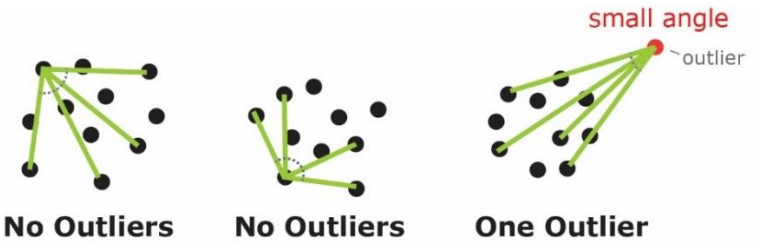


Figure 2.8 – ABOD

A summary of the outlier detection algorithms can be found in Table 2.1.

Methods	*Pros*	*Cons*	*Outlier determination*
ANGLE-BASED			
Example			
Angle-based outlier degree	· No parameters needed · No distance assessment needed	· Computationally inefficient · No precise threshold	Smaller relative angles
CLUSTERING			
Examples			
K-means Clustering DBSCAN	· Relatively fast · Insights about local distributions	· Loses granularity · Variability due to randomization	Distance
DENSITY-BASED			
Example			
Local Outlier Factor	· Accounts for local skew	· Complicated · Computationally inefficient	LOF value > 1
ISOLATION-BASED			
Example			
Isolation Forest	· No distance assessment needed · Handles high dimensional data	· Suffers from "blind spots"	Anomaly score
STATISTICAL MODEL-BASED			
Examples			
The Three-Sigma Rule Boxplots Cerioli outlier detection Cook's Distance Grubbs' Test Mahalanobis Distance Median Absolute Deviation Order Statistics Weighted Mean	· Easy to apply · Transparent · Computationally fast	· Mostly for univariate data · Distribution-based	Confidence tests Thresholds

Table 2.1 – Summary of Outlier Detection Methods

Chapter 3 | Methodology

This chapter begins with a description of the data and how outliers were artificially introduced into the simulations. Next, the creation of the benchmarking process, the most time-consuming part of the research, is outlined and the metrics used to qualify the results are given. The chapter culminates with an explanation of how the ensemble algorithm was created with the insights gained during benchmarking.

3.1 Data

The following sections discuss the source, structure and processing of ASML's proprietary data used to conduct the research. The structure of the data is slightly convoluted, so care is taken to explain it and also provide the reader with some background understanding of how the data is created.

3.1.1 Source of the data

The data used for outlier detection comes from the ORC team's past production of calibration wafers. As stated in the introduction, lithography machines are expensive and making them available for the company's internal experimentation rather than sales means gathering data comes at quite an expense to ASML. As a result, a limited number of datasets were available to use for outlier detection analysis.

While overlay data can be produced by lithography machines, ASML has a faster method of producing the data by scanning the wafers in Yieldstar. Yieldstar is ASML's metrology system for measuring overlay with scatterometry, a method that bounces a beam of light off the wafer and measures structures on the wafer using the diffused light it returns. To read the data, printed wafers are loaded into a FOUP (Front Opening Universal Pod) which performs a role similar to the carousel in a multi-disc CD player. Newly printed wafers are stacked in between golden wafers. As a reminder, "golden wafers" are the reference wafers that the marks printed on new wafers will be compared to. The numbers and configurations in which wafers are loaded can vary, but Figure 3.1 below shows an example configuration of a FOUP loaded with 25 wafers. A dummy wafer is in the first position, then a golden wafer appears in the second position followed by three newly printed wafers. The configuration of a golden wafer followed by three new wafers is repeated five more times to add up to a total of 25 wafers. The golden wafer in position two serves as the reference measurement for the wafers in positions three, four and five. (For this reason, a golden wafer will also be referred to as a

"reference wafer" in this report.) The additional golden wafers are interspersed in the FOUP as a new reference for the subsequent wafers. This creates an updated reference adjusted for machine drift caused by the machine heating up during use.

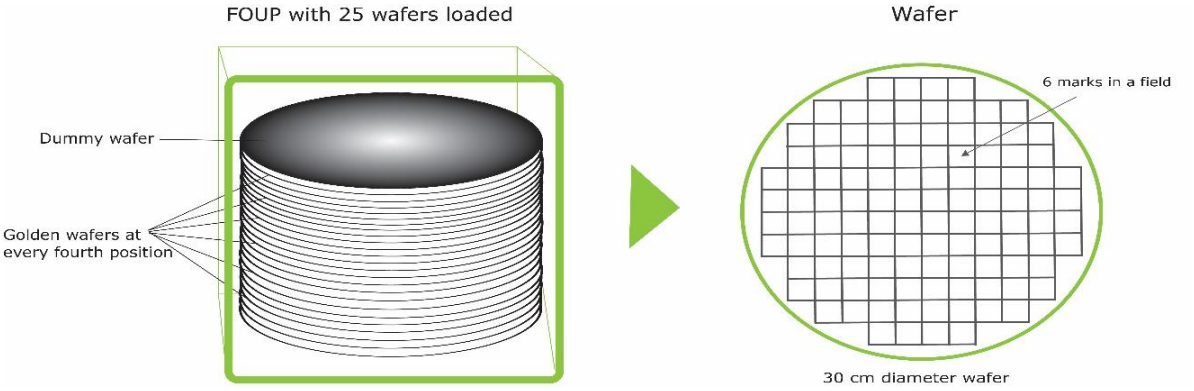


Figure 3.1 – Wafer Profile

The FOUP for loading wafers into the machine and a view of the wafer. Wafers used in the benchmarking process had many more fields than depicted on the wafer on the right.

3.1.2 Structure of the data

The dataset used for benchmarking contains data from 24 wafers (25 wafers – 1 dummy wafer) and each wafer contains thousands of fields. Each of those fields has 6 marks within it that are used for calibration purposes. ASML refers to the marks as "shifts," but these will be referred to with the more descriptive name of "marks" here. (ASML requested that I do not include an illustration of the marks in this report. The marks are etchings on the wafer that the lithography or Yieldstar machine uses to identify a position on the wafer as it scans the wafer.) It is the position of those marks relative to marks on the reference wafer ("golden wafer") that create the data used for the benchmarking of outlier detection algorithms. For consistency, data used in the benchmarking process was limited to four wafers positioned sequentially in the FOUP and only mark #1 in each field was used.

Lithography machines print and scan wafers in horizontal and vertical directions, translating data as a coordinate plane with *x* and *y* values. The dataset used in the outlier detection process was filtered for known errors; such as misalignment on the wafer table, and was also translated from the raw position of the mark to instead represent the delta between the wafer mark position and that of the corresponding mark on the reference wafer. Therefore, the *x* and *y* data represent the error from what was expected for the position of the mark. These *x* and *y* error distances from the reference wafer can be illustrated as a series of vectors as seen in **Error! Reference source not found.** below. T he vectors in each field are formed by a line from the mid-point, or origin, of the field to the point

formed by the *x* and *y* values. These vectors are of varying length and direction. The longer vectors show a greater delta for the positions of the mark compared to corresponding mark positions on the reference wafer. Going forward, the *x* and *y* values in a field will be referred to as "the data" or collectively as the "data point" but will always refer to the delta, or difference, between the position of mark #1 in the field and the position of mark #1 in the same field of the reference wafer.

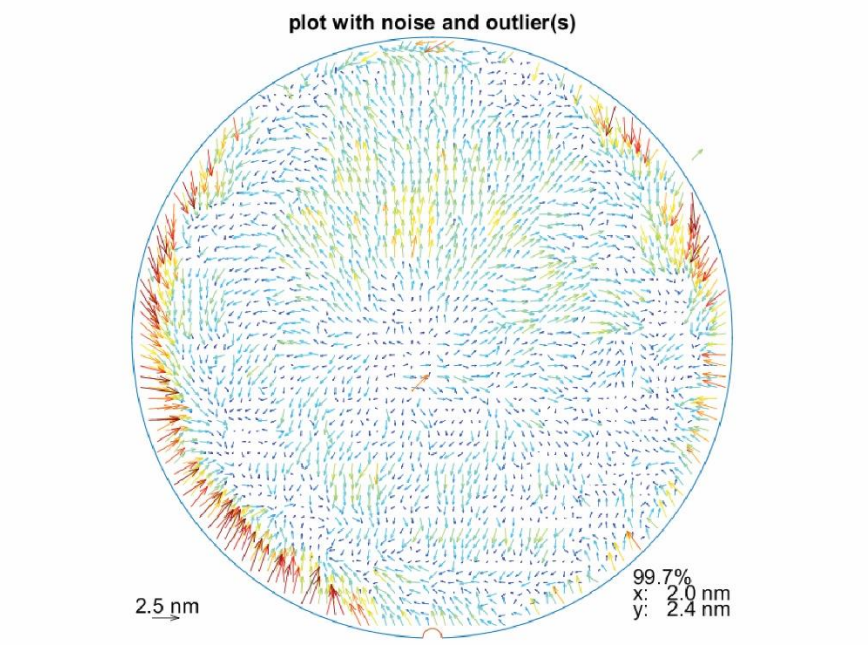


Figure 3.2 - Vector Plot

The image shows the vectors formed by the corresponding x and y values. This is the view at 2.5 nm. At 3-sigma (99.7%) with the other 0.3% of values removed, the absolute value of the largest x value difference from its reference is 2.0 nm and the largest absolute value of the y value difference is 2.4 nm. The color scale goes from blue (shortest vectors) to green to yellow to red (longest vectors).

Distinct patterns on the wafer in **Error! Reference source not found.** can be seen. The longer v ectors around the edges may be caused by the residual effect of the clamping of the wafer to the wafer table. Many patterns are distinct to individual lithography machines and are known as a "machine fingerprint."

Each of the six different marks on a wafer returns 22,464 potential values from across the scanned coordinate plane of the wafer. After deleting NaNs returned by scanning an area of the coordinate plane without a mark, the total number of numerical values for mark #1 was 3,037 per wafer. Since these values represent points on the coordinate plane, there are 3,037 *x* and *y* values.

The outlier algorithms found to have the best performance were later tested on wafers from two separate datasets that had been reserved for this purpose. For the dataset used in the

benchmarking process, random noise was added each time the algorithm was run and artificial data to create outliers was randomly placed in different locations each time the algorithm was run. (The use of artificial data for outliers is explained in greater detail in section 3.2.) Therefore, the benchmarking dataset was constantly changing for each simulation.

3.1.3 Processing the data

ASML's WEC team accesses the data using a Matlab script that processes the data and filters out known distortions caused by heat, lens focusing issues, the misalignment of the wafer on the wafer table and the clamping of the wafer to the table. The Matlab script performs a series of calculations and adjustments to the data and the results can be accessed on various levels of filtration. The data used for outlier detection has been fully processed and filtered resulting in a distribution of *x* and *y* data points with a bell-shaped curve but with a right skew preventing it from being considered normally distributed. The median of the *x* values on mark #1 on wafer #2 is -1.21766e-10.

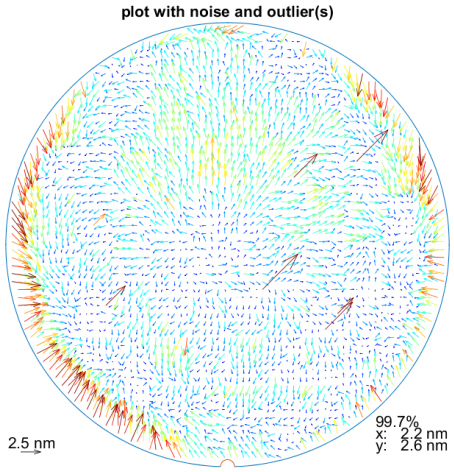
Outlier detection runs as a module within the WEC team's Matlab script. Therefore, it was necessary to build the benchmarking platform in the Matlab programming language in order to integrate the WEC team's current outlier detection algorithm into the simulations and to be able to use the existing functions for visualization of the wafer with vectors. Any new algorithms created also needed to be able to integrate back into the existing production code.

3.2 Artificial Noise and Outliers

To create realistic simulations, a varying set of random noise was added to the data values on each of the four wafers for each run of the simulation. The purpose of this was to imitate the noise developed by the machine during the processing of the wafers. The added noise always had a standard deviation of 1×10^{-10}; a value derived from the results of the WEC team's past work to measure the average standard deviation of machine noise.

Once the wafers were prepared with the noise an initial problem in setting up the simulation remained. This was determining how to validate and measure outlier detection on the dataset when there was no way to know for certain how many outliers existed in the data. (Given the heavy pre-processing performed on the data, the majority, if not all, of the outliers that existed in the raw data should have been filtered out.) The solution was to make an assumption that there were no outliers in the dataset and to introduce outliers artificially.

In the benchmarking process, ten outliers are added to a wafer in random positions for each run; each with ten different vector lengths. The lengths always varied in equal increments from 1 picometer (1×10^{-12}) to 4 nanometers (4×10^{-9}). This is illustrated in Figure 3.3 with the larger outliers appearing as dark red vectors. The shortest outlier at 1 picometer was expected to be nearly impossible to detect and the longest outlier with a vector of 4 nanometers was expected to be very easy to detect. The placement of the outliers for each run of the simulation was recorded and performance was determined on the success of finding the artificially placed outliers.



Outlier	Size of vector
1	1.00e-12
2	4.45e-10
3	8.90e-10
4	1.33e-09
5	1.78e-09
6	2.22e-09
7	2.67e-09
8	3.11e-09
9	3.55e-09
10	4.00e-09

Figure 3.3 – Outliers on the Vector Plot

Outliers ranged from a size of 1 picometer to 4 nanometers. While the artificially added outliers in the figure above are all pointing in the same direction, tests were also run with the vectors pointing in varying directions.

3.3 Approaches in Data Comparison

Two different approaches were tried in the experiments.

3.3.1 The "Across Wafers" Approach

The WEC team's current outlier detection algorithm compares data across the wafers. It compares a data point to the same data point across all wafers being viewed together. In the benchmarking process four wafers were used, so the outlier detection algorithm's calculations were made with just four data points to evaluate to make an outlier determination at each data point. See Figure 3.4 for an illustration of these four points. The *x* and *y* points are looked at separately with this approach. The use of four wafers is representative of a typical calculation due to the batching of wafers in the FOUP with a reference wafer, but the algorithm can also be used on a larger number of wafers.

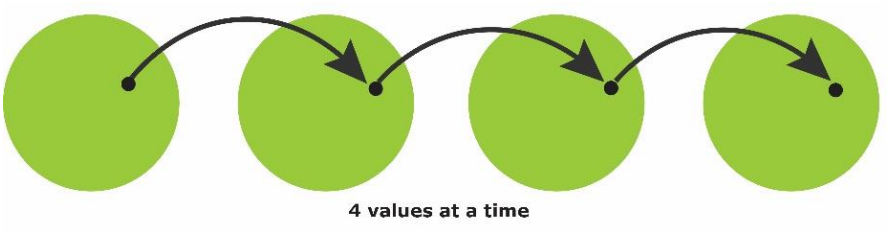


Figure 3.4 – Across Wafers

3.3.2 The “By Wafer” Approach

Initial experiments were run comparing the data in all fields of one wafer instead of one field across the four wafers. |As Figure 3.5 shows, this approach supplies 3,037 data points to evaluate for the outlier determination, compared to just four data points in the “Across Wafer” method. The reason for trying this change in approach was to determine if using a larger dataset would lead to improved outlier detection performance.

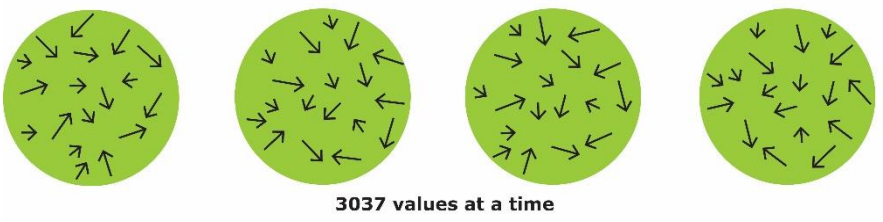


Figure 3.5 – By Wafer

3.4 Challenges in Outlier Detection

Changing to the “By Wafer” approach created the following set of challenges for outlier detection:

- Defining a *normal delta* when vector lengths vary greatly by region
- Defining a *normal delta* when there are imprecise boundaries between regions
- Defining a *normal delta* when there is noise in the data
- Defining a *normal delta* when a *normal delta* keeps evolving as drift occurs

For example, in Figure 3.6 there are two vectors in red with similar lengths pointed out. Both look like possible outliers, but only the upper vector is an outlier. The upper vector was artificially placed in the data. It has a longer vector than the surrounding vectors and an angle that differs from the surrounding vectors. The lower vector, a part of the dataset, is only slightly longer than the surrounding vectors and has a similar angle to the surrounding vectors. (Increasing the zoom of this document is helpful here.)

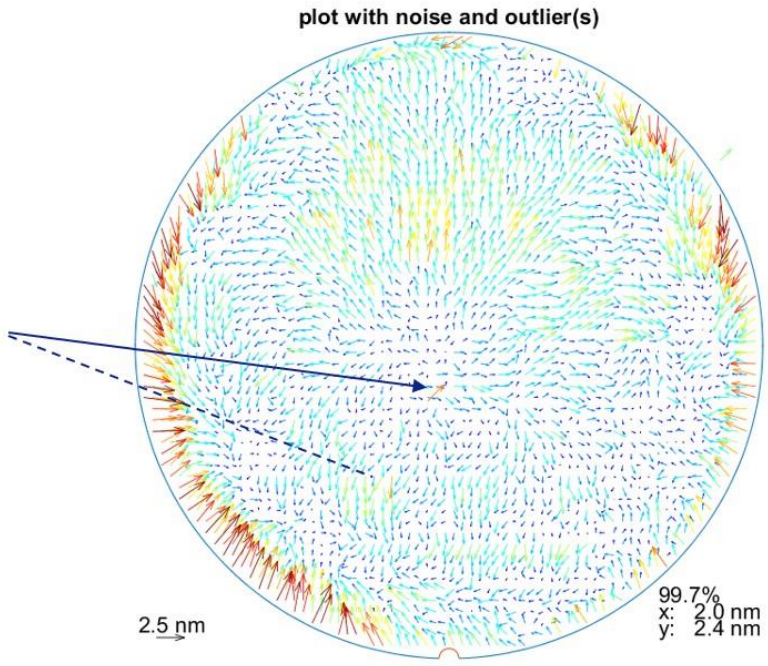


Figure 3.6 – What Makes an Outlier?

Two vectors of similar length are highlighted with the upper one an outlier and the lower vector a normal data point. This illustrates that the vector's length is only one criterion in determining an outlier. The vector's angle and length relative to the surrounding vectors are important determinants too.

Detecting outliers with the "Across Wafers" approach has its issues too. Outlier detection would be simple if it was just a matter of comparing a data point on a wafer to the average of the same data point on each of the wafers. Once the values are averaged, though, a fraction of any outlier transfers to the average. It is for this reason, that the median is the preferred measurement of central tendency in outlier detection. In Figure 3.7, the problem with averaging is illustrated. While the fourth wafer is labeled as an outlier with its longer vector and different angle, there is no obvious threshold as to what distance of vector or change in angle makes an outlier.

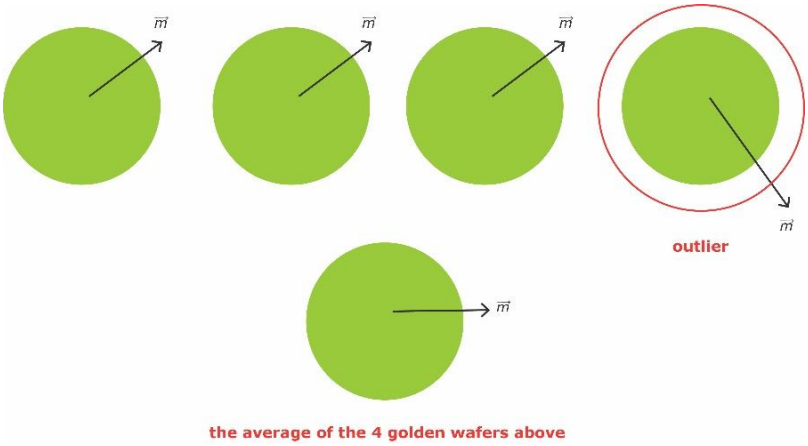


Figure 3.7 – The Problem with Averaging

With an even number of wafers, another difficulty in using the "Across Wafers" approach occurs. As illustrated in Figure 3.8, when half of the wafers have similar vectors and the other half has similar but different vectors it is difficult to determine which half is within an expected range for the dataset or whether both halves are outliers. Another question to ask is whether the change in position between the second and third wafers is extreme enough to even suggest there are outliers.

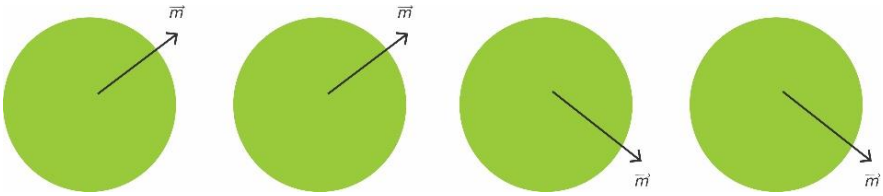


Figure 3.8 – The 2-2 Problem

3.5 Metrics to Be Used

The ranking of the performance of an outlier detection algorithm depended first and foremost on its ability to find outliers. To measure an algorithm's ability to detect outliers, the average number of artificially placed outliers detected across simulations was captured. For example, in a run of one hundred simulations if the algorithm found eight of the ten outliers in fifty simulations and nine of the ten outliers in the other fifty simulations then it would return an average of 8.5 for that run. The averages returned were compared to the averages returned by the WEC team's Weighted Mean algorithm run simultaneously on the same data. Dividing the average number of outliers found by the ten artificial outliers is equivalent to the statistical measure known as recall (the proportion of artificial outliers identified correctly).

Additionally, the delta (estimated machine fingerprint – reference machine fingerprint) averaged across the runs for the algorithm needed to be lower than Weighted Mean's average delta. For the third metric, the maximum of the absolute values of the deltas for the *x*-values and the *y*-values also needed to be lower than the Weighted Mean algorithm's maximum absolute value of deltas. Similarly, the mean 3-sigma for *x*-values and *y*-values needed to be lower than that of the Weighted Mean algorithm. The mean 3-sigma represents the average of the deltas after the largest .003% of the data was removed. The fourth metric was generally the easiest for the algorithms to beat while the third metric, maximum of the absolute values of the deltas, proved the most troublesome metric for many of the algorithms. This suggests that it was just a few fields with large deltas that would take the algorithm out of contention; even if on average it returned overall lower deltas.

For the outlier detection algorithms most competitive with the Weighted Mean algorithm, additional hurdles were added. To be declared a better algorithm than the Weighted Mean, it also needed to show a higher accuracy and precision (proportion of outliers detected that were correct) than the Weighted Mean. Only when an outlier detection algorithm could match or improve upon the Weighted Mean's performance for all of these metrics could it be declared a better outlier detection algorithm.

Table 3.1 below summarizes the metrics that the final performance was evaluated upon.

Metric	*Description*
# of Successes	· The number of times the artificial outlier was found.
Delta	· Estimated machine fingerprint - Reference machine fingerprint
Max x & Max y	· The maximum absolute value of the delta after the top .3% of data is removed.
Mean 3-sigma x & Mean 3-sigma y	· The average 3-sigma of delta after .3% of data is removed.
Accuracy	$\frac{True\ Positives + True\ Negatives}{All\ Values}$
Precision	$\frac{True\ Positives}{True\ Positives + False\ Positives}$
Recall	$\frac{True\ Positives}{True\ Positives + False\ Negatives}$

Table 3.1 – Summary of the Metrics Used

3.6 Creating the Benchmarking Platform

Before the challenges in outlier detection could even be addressed, a benchmarking platform needed to be built. This was the most time-consuming part of the research project. The benchmarking platform automates simulations of outlier detection and measures the performance of each different outlier detection algorithm relative to the WEC team's current Weighted Mean algorithm. The goal of benchmarking was to assess the performance of the Weighted Mean algorithm relative to the other algorithms. It was necessary to integrate the benchmarking platform with the WEC team's existing functions before any outlier detection algorithm could be tried. The benchmarking platform automated and performed the following tasks:

- Adding noise to the data to imitate machine processing
- Randomly placing outliers in the data
- Tracking the positions of the outliers
- Preparing and integrating the data with other functions
- Calling and calculating the outlier detection algorithm
- Removing the outliers detected
- Collecting the results of the successes in outlier detection
- Calculating the delta between the pre and post detection datasets
- Calculating standard deviations
- Visually representing results through reports and graphs

Separate benchmarking platforms for the "By Wafer" approach and the "Across Wafers" approach were built.

3.6.1 Benchmarking the "By Wafer" Approach

The WEC team's current Weighted Mean outlier detection algorithm attempts to determine outliers based on just four data points, the same data point across four different wafers. Hypothesizing that more reliable results could be found with more data, a different approach was taken to outlier detection. This approach looked at all data points on one wafer at a time with the goal of detecting every outlier on a single wafer. In the benchmarking process this meant finding all ten artificial outliers, from one picometer to 4 nanometers.

3.6.1.1 Determine the “True Machine Fingerprint”

Figure 3.9 shows the first steps in the benchmarking process. Each of the four wafers of the dataset averaged together is referred to as the “true machine fingerprint.” It is depicted as a blank wafer in the following graphics, but the average of the four wafers would actually form a wafer with various vector lengths; much like the wafer shown in **Error! Reference source not found.**.

3.6.1.2 Add the random noise and calculate the “Reference Machine Fingerprint”

Next, the random noise is added individually to each of the four wafers. After the noise is added, each point across the four wafers is averaged to derive the “Reference Machine Fingerprint.” This is equivalent to the True Machine Fingerprint plus the average noise added.

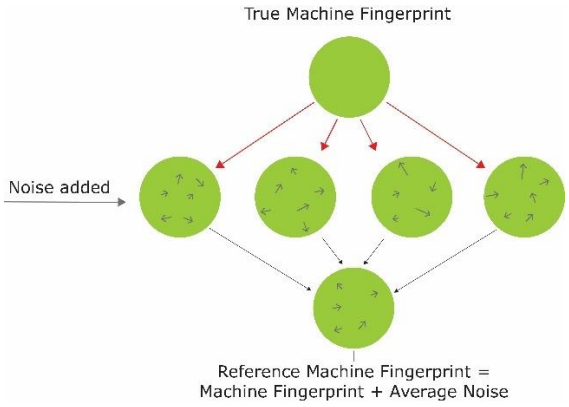


Figure 3.9 – The Benchmarking Platform by Wafer (Part 1)

3.6.1.3 Add the random artificial outliers and subtract the average of the datapoints per field to remove the machine fingerprint

The ten artificial outliers are then added to just the first wafer. They are placed in random locations. (See the upper half of Figure 3.10.) In addition to the random locations, experiments were made with the outliers placed in random directions and also uniformly pointing in the same direction. It was found that the two different types of placements had no impact on the results of the detection algorithms.

The values of the corresponding data points on each of the four wafers are averaged (4 values averaged across 3,037 fields) and then the average for the field is subtracted from the field’s data point on each of the four wafers. The result of subtracting the average for each point is that the machine fingerprint is removed. Removing the machine fingerprint makes outlier detection easier and the outliers become more exposed.

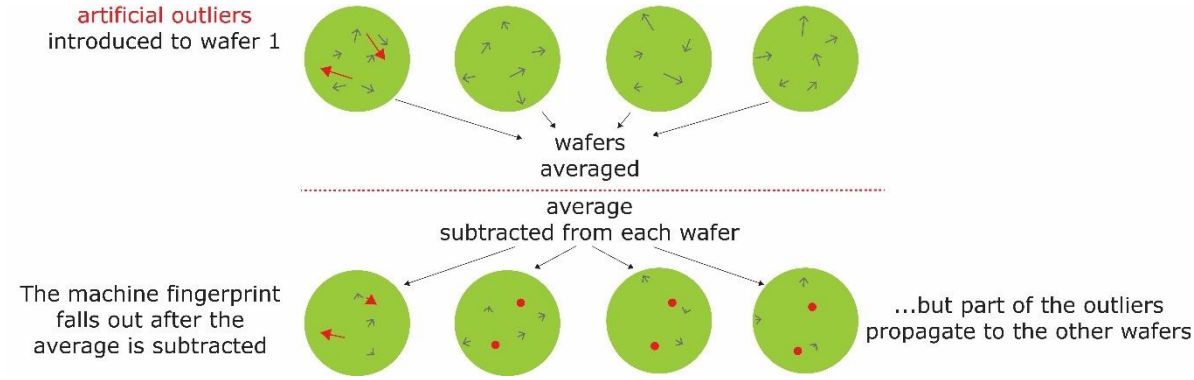


Figure 3.10 – Benchmarking Platform by Wafer (Part 2)

A challenge with this approach, however, is that a small part of the outliers on the first wafer will propagate to the other three wafers due to the averaging. Subtracting the median instead of the average is a workaround for this issue. Eventually, though, the even more effective approach of dividing the wafer into neighborhoods was discovered and the removal of the machine fingerprint became no longer necessary.

3.6.1.4 The Neighborhood Approach

The large variation in the "topography" across the wafer, due to the changes in vector lengths, made outlier detection more difficult for the "By Wafer" approach. A solution was created to look at distinct areas, or neighborhoods, of the wafer and run the chosen outlier detection repeatedly within each neighborhood. Figure 3.11 illustrates this approach. The size of the neighborhood was set by choosing a diameter centered on the datapoint at each point of the iteration. (3,037 different neighborhoods were formed.) All datapoints lying within the circumference formed around the diameter were used in the algorithm's calculation. Since a linear regression will be run on the datapoints for the calculation of Cook's Distance, the size of the circumference needed to yield enough datapoints to run the regression. A minimum yield of 30 datapoints was set. Many simulations of circumference lengths were tried, with this minimum yield of 30 points as a constraint, before setting an optimal diameter size of .04 cm (400,000 nm).

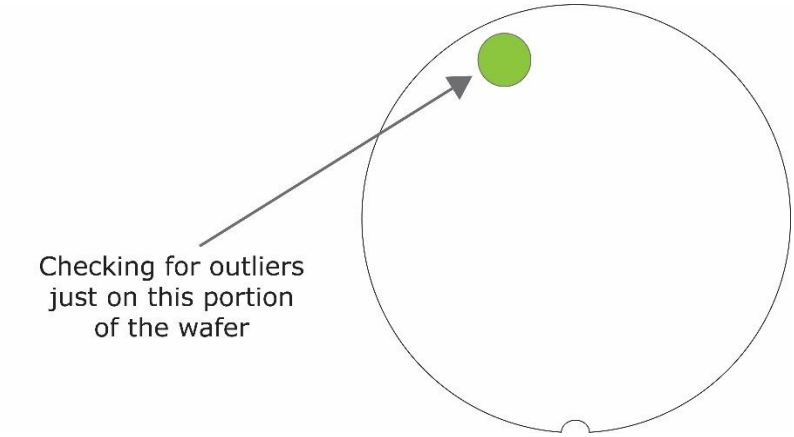


Figure 3.11 – A Neighborhood

3.6.2 Benchmarking the "Across Wafers" Approach

3.6.2.1 Add the random noise and calculate the "Reference Machine Fingerprint"

In outlier detection conducted point-by-point across the four wafers, the steps of adding noise to the wafers and averaging them to get the Reference Machine Fingerprint are the same as in the "By Wafer" approach.

3.6.2.2 Add random artificial outliers

Then outliers are added to the first wafer, as in the previous process, but averages are not subtracted. For each run, the positions of the artificial outliers are stored.

3.6.3 Running the Simulation

With the data prepared, various outlier detection algorithms are tried out. Early simulations tried as many as eight different algorithms on the data at a time. As it became clearer which algorithms were the most effective, this was reduced to four algorithms at a time and then two algorithms at a time. The WEC team's Weighted Mean algorithm was always one of the algorithms, since the goal was to measure its performance relative to other algorithms.

Any outliers found were removed from the data and replaced with a NaN value. Comparisons were made to evaluate the difference in results with the removal of outliers and with the replacement of outliers by the median of the relevant points (either the four points across the wafers, the points within a neighborhood, or all points on a wafer). However, with the ten outliers making up only 0.3% of the data, the difference was minimal. Also, the outliers were considered to actually be errors

created in the machine process so the decision was made to remove outliers rather than replace them with another value. Univariate detection algorithms were run on both the *x* and *y* data. If an outlier was found in an *x* datapoint, the corresponding *y* datapoint was also removed, and vice versa.

3.6.4 Collecting and Measuring the Results of the Simulation

The number of successes was calculated by comparing the the positions of the outliers detected by each algorithm and the positions where the artificial outliers were placed. The sum of matching positions determined the number of successes and the non-matching positions indicated a false positive or false negative.

With the detected outliers removed, an estimated machine fingerprint is calculated for the remaining data for each outlier detection algorithm. The estimated machine fingerprint for an outlier detection algorithm is the calcuation of the average datapoints (vectors) for each field for the four wafers after the outliers have been deleted. A perfect outlier detection algorithm that finds all outliers and no false positives would have an estimated machine fingerprint very close to the reference machine fingerprint. Only the differences in noise would cause variations between the values. Differences in outliers found will cause variations in the estimated machine fingerprint calculation for each outlier detection algorithm. This is not due to just the result of differences in successes, but also due to the number of false positives. A false positive will cause the deletion of a valid datapoint and this will change the estimated machine fingerprint for that particular field and create a delta with the reference machine fingerprint.

In addition to the calculation of the difference between the “estimated machine fingerprint” and “the reference machine fingerprint,” the mean 3-sigma and maximum differences were also recorded.

3.6.5 Visualizing the Results

The benchmarking platform created multiple vector plots illustrating the measurements of the results. Figure 3.12 below shows an example of the delta between the estimated machine fingerprint and the reference machine fingprint, the maximum standard deviation of the x values at each amplitude of outlier, and the mean 3-sigma of the x values at each amplitude of outlier.

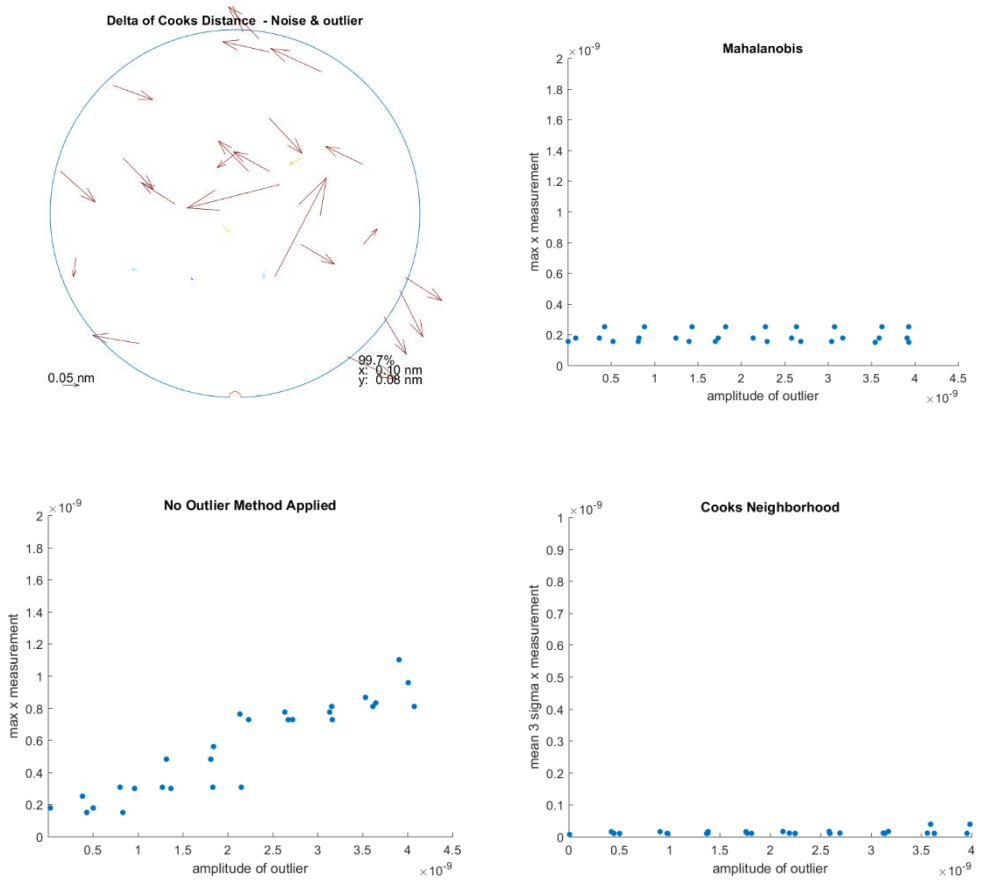


Figure 3.12 – Measuring the Results

The first plot on the left shows vectors representing the deltas after the reference machine fingerprint is subtracted from the estimate machine fingerprint for the Cook's Distance method. The vectors represent missed outliers and false positives. If zoomed, some differences in noise would also show up. The upper right plot represents the maximum standard deviation of the x values at each amplitude of the outlier, while the lower left shows the maximum standard deviation of the x values at each amplitude when no outliers are detected. The value increases as the length of the outlier increases. The final plot in the lower right shows the mean 3-sigma pf the x values for each amplitude of the outlier.

3.7 Tuning Parameters

In between simulations, parameters for the various algorithms where changed and the simulations were re-run in an effort to find the optimal parameter settings. For speed, only twenty simulations were run at a time. One hundred simulations were run after the parameters to use were determined. Table 3.2 shows the parameters associated with some of the more successful outlier detection algorithms for this project. Table 3.3 illustrates how the parameters were tuned.

Algorithm	*Name of Parameter*	*Description*
Mahalanobis	· Probability	Probability to use in chi-square equation | controls sensitivity
	· Degrees of freedom	Degrees of freedom - (# of x factors) | needed for chi-square
MAD	· Cutoff	Threshold value
Cook's Distance	· Numerator	Numerator to be divided by n
Rank	· Threshold	Threshold value
Neighborhood	· Neighborhood size	Diameter

Table 3.2 – Parameters for the Algorithms

PARAMETERS				RESULTS				METRICS		
Percentile	*Cutoff*	*Neighbor-hood size*	*Numerator*	*TP*	*FP*	*FN*	*TN*	*Accuracy*	*Precision*	*Recall*
0.989	0.3	0.01	9.5	9.35	6.3	0.65	3020.7	0.9977	0.5974	0.9350
0.989	0.3	0.01	12	9.45	5.65	0.55	3021.4	0.9980	0.6258	0.9450
0.989	0.3	0.01	15	9.7	6.7	0.3	3020.3	0.9977	0.5915	0.9700
0.989	0.3	0.015	9.5	9.45	6.3	0.55	3020.7	0.9977	0.6000	0.9450
0.989	0.3	0.015	12	9.4	6.9	0.6	3020.1	0.9975	0.5767	0.9400
…	…	…	…	…	…	…	…	…	…	…
0.993	0.4	0.02	9.5	Best Parameters:			percentile:	0.989		
0.993	0.4	0.02	12				cutoff:	0.3		
0.993	0.4	0.02	15				neighborhood size:	0.01		
							numerator:	12		

Table 3.3 – Tuning Parameters

Twenty simulations were run for each combination during tuning. Best results are highlighted in green.

3.8 Building a Customized Algorithm

In the course of benchmarking the outlier detection algorithms against the Weighted Mean algorithm, it was noted that two particularly well-performing algorithms were often unsuccessful at identifying different outliers. These were median absolute deviation (MAD), and Cook's Distance. Both algorithms were always successful in finding the seven largest outliers, but among the shortest three outliers it was often the case that where one would be successful at identifying the outlier the other might be unsuccessful. The idea was conceived to combine the two algorithms into a new ensemble outlier detection algorithm that uses elements of both the Across Wafer and By Wafer approaches.

In the process of combining the two algorithms, it was determined that this was an opportunity to address other weaknesses of the detection algorithms found during benchmarking. These other weaknesses include a lack of facility to directly compare the angles of the vectors within

a neighborhood and an inability to adjust parameters to the changing topography of the wafer. A final enhancement to the ensemble algorithm was to add pre-filtering for suspicious datapoints. This reduced the number of data points reviewed and sped up the algorithm. It also prevented the two component algorithms from being too aggressive in finding outliers as the pre-filtering set a threshold as to what would be considered an outlier. This reduced the number of false positives. Figure 3.13 provides a diagram of the algorithm.

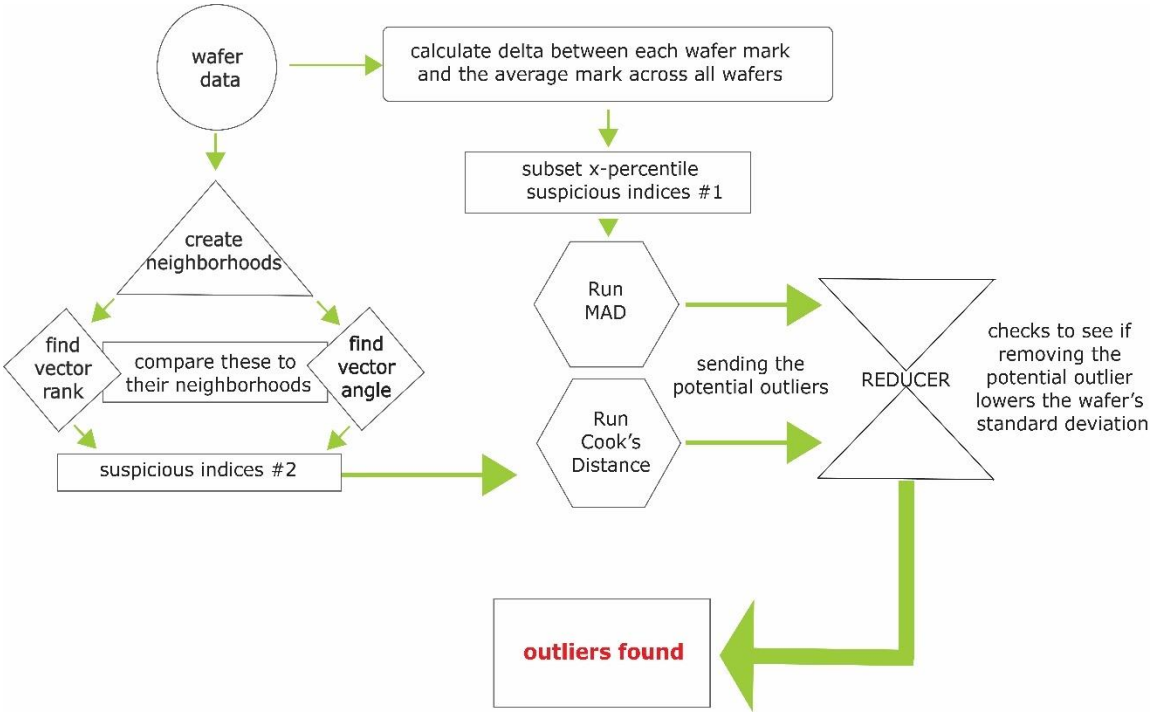


Figure 3.13 – An Overview of the Ensemble Outlier Detection Algorithm

3.8.1 The First Wave

The ensemble algorithm was partitioned into two "waves" for the data to flow through. Each wave consisted of the same three steps. The first step subset the data down to a set of "suspicious indices" and then the next step ran the outlier detection algorithm on the subset of data. The final step was a secondary test on the results to check if deletion of the datapoints suggested to be outliers would reduce the mean 3-sigma of the dataset.

The first step in the first wave of the algorithm is to subtract the difference of the datapoints in all fields of wafer #1 from the average of the datapoints in all fields for all four wafers. Next the absolute values of the difference between wafer #1 datapoints and the averages is taken and ordered from largest to smallest. The data is then subset for just the datapoints in the n-percentile of the data. This parameter was hardcoded to the 99^{th} percentile after test simulations. Therefore, only one percent of the wafer #1 data with the largest absolute value of the deltas is used and the rest of the data is ignored. This one percent is called the "suspicious indices." (Each field is identified through an index value, so "suspicious indices" can be thought of as "suspicious fields.") Since the process is run separately on the *x* and *y* values and then the indices of these values are later combined into one list, the suspicious indices can actually range from 1% of the data all the way to 2% of the data if there is no crossover in the indices found for *x* and *y*. In practice, though, the number remained close to 1%.

Note that the action above, subtracting the average of the data for all four wafers from one wafer, is the same action that caused the propagation of part of wafer #1's outliers to the other three wafers in Figure 3.10. The reason it can be used here is that it is only used to identify the "suspicious indices;" the actual value derived from subtraction is never used. Instead, the original value of the datapoints in wafer #1 for the suspicious indices will be used. In the worst case, a large outlier on one of the other three wafers will cause an index to look suspicious for wafer #1, but it will be discarded in the following step of the algorithm when the outlier detection algorithm is run on the data. An additional note here is that a better approach for the algorithm would be to subtract the median rather than the average, as advised in section 3.6.1, but one of the team's existing Matlab functions was used here to quickly handle the averaging and subtraction. Since any false suspicious indices created by the averages would be easily dealt with by the outlier detection algorithms, it was not necessary to create a new function that used the median instead of the average.

Next, the MAD algorithm is run using the "Across Wafers" approach. To calculate the MAD number, the entire dataset is used. Then, when comparing the values to the derived MAD number, only the suspicious indices are considered. MAD is run separately on the *x* and *y* values and the findings are combined, so if the *x* value for an index is found to be an outlier then the corresponding *y* value is also kept as an outlier.

All indices found by the MAD algorithm to be outliers are run through a secondary test. This test iterates through the indices labeled as outliers and checks to see if the removal of that value lowers the dataset's mean 3-sigma. Only those indices that would reduce the dataset's mean 3-sigma through its deletion are considered outliers and deleted from the dataset. This process reduces the instances of false positives.

3.8.2 The Second Wave

For the second wave of the ensemble outlier detection algorithm, neighborhoods are built with the full dataset minus any indices that were already found to be outliers by the MAD algorithm. Then, a list of suspicious indices is determined by the number of degrees difference of an angle from the average of the angles in its neighborhood. Angles are determined using the radian of the vector formed by the x and y value. If the angle in any index (field) differed by more than 25 degrees with the average angle of its neighborhood, then it is labeled a suspicious index. Twenty-five degrees was chosen through testing and hardcoded. Additionally, the length of vectors was ranked in length across the wafer from the value of 1 (shortest) to 3,037 (longest). Any index ranking below 100 or above 2600 was labeled as a suspicious index.

Only the intersection of suspicious indices found for the ranks and for the angles were retained and Cook's Distance was applied to this subset of indices after a linear regression was run. Using indices determined by Cook's Distance to be highly influential, the same secondary test for impact on mean 3-sigma that was run for the datapoints determined by MAD to be outliers was used. Notably, this process of measuring the impact of deleting a datapoint is extremely similar to the process that was already run in Cook's Distance.

The indices found to be influential by Cook's Distance were combined with the indices found to be outliers by MAD and these were the indices output by the algorithm to be the outliers in the dataset. Notably, both the "Across Wafers" and "By Wafer (with Neighborhoods)" approaches were used in the algorithm. Table 3.4Table 3.4 below gives a step-though of the ensemble algorithm's operation.

Step	*Description*
Step 1:	Calculates delta between each wafer and the average of all wafers
Step 2:	Subsets the largest (98--can change this) percentile of the deltas
Step 3:	Call the index numbers of this subset the 'suspicious indices'
Step 4:	Run the MAD method for all indices across the wafers, and create the 'removethese' subset
Step 5:	For the removethese subset, see if inserting a NaN in their place lowers the standard deviation
Step 6:	Record the indices that did lower the std. dev.
Step 7:	Call this subset MADkeepthese, and check if it finds all outliers. - if it does, we're done. Insert NaNs in those indices and record the overlay
Step 8:	Prepare a secondary test, first create neighborhoods, calculate the degree for each vector and then rank the areas on the wafer.
Step 9:	Create a new subset based on the degrees and ranks, call this 'supicious2'
Step 10:	Find the intersection of points in the original 'suspicious indices' and in 'supicious2'
Step 11	Run Cooks Distance on the points in the intersection of the 2 subsets in the previous step
Step 12:	For the removethese subset of Cooks, see if inserting a NaN in their place lowers the standard deviation - call these points CDkeepthese
Step 13:	Take the unique indices with the combined MADkeepthese (Step 7) and the CDkeepthese indices, these are your final_indices
Step 14:	Collect the successes and failures for finding the outliers at each amplitude.
Step 15:	Calculate overlay on final_indices
Step 16:	Calculate the delta
Step 17:	Calculate the TP, FP, TN, FN, Accuracy and Precision

Table 3.4 – Steps for the Ensemble Algorithm

Chapter 4 | Results

In benchmarking ASML's Weighted Mean algorithm to other outlier detection algorithms, initial simulations established that it was possible to find an algorithm that could outperform the Weighted Mean on some metrics but it was very difficult to beat it on all metrics. As displayed in the final results in this section, the Weighted Mean algorithm has an average accuracy of 90.76%, precision of 60.5% and recall of 90.5%. A large part of the algorithm's success is that it returns very few false positives. This chapter will first present some of the intermediate results and show where these algorithms could and could not beat the Weighted Mean algorithm. Then the results for the Ensemble algorithm run on two different, out-of-sample datasets are presented.

4.1 Implementation in R to Facilitate Analysis

To expedite analysis, some algorithms were initially tested in R and only moved to the benchmarking platform in Matlab if the algorithm showed promising results in identifying outliers. The reason for this was to and take advantage of existing libraries in R and avoid having to build the algorithm in Matlab before being able to see the performance. For basic statistical algorithms, this was not necessary as they could be quickly implemented in Matlab. Nonetheless, the algorithms tested in R used the same data as those tested in Matlab. The original dataset plus noise and outliers was created in the benchmarking platform then downloaded to a csv file that was uploaded to R. This was done multiple times so that there were several versions of the dataset available for the simulations in R.

4.2 Algorithms that Did Not Pass the First Metric

The first metric considered was the algorithm's ability to successfully find the artificial outliers. If an algorithm did not score well here (an average of 8.5, or better, out of 10 artificial outliers), it was not tested further. Statistical algorithms scored well in locating the artificial outliers. Table 4.1 below shows the outlier detection algorithms that did not produce sufficient results to merit further analysis

Model	*# of Artificial Outliers Detected (Average out of 10)*
Local Outlier Factor (LOF)	7
Isolation Forest	7
Cerioli outlier detection	7
K-means clustering	6.5
Boxplot	6
DBSCAN	6
Grubbs test	6
ABOD	*

*The ABOD algorithm did not complete running within 3 hours, so it was not used.

Table 4.1 – Unsuccessful Algorithms

4.3 "By Wafer" Approach Sample Results

Initial simulations were done with the "By Wafer" approach and showed promising results for statistical models such as Mahalanobis, MAD, and Cook's Distance. One of the plots made during early testing to illustrate findings can be seen in Figure 4.1. Later illustrations were made with vector plots instead. On the x-axis is the distance from the origin of the field for the x data and the y-axis represents the distance from the field's origin for the y data. Red points are the datapoints that the outlier detection algorithm has identified as an outlier while blue points are not considered to be outliers. The goal of this testing was to better understand how sensitive the algorithms were, or in other words, visualize where the thresholds for outliers began.

Two artificial datapoints were introduced, and the circled points on the upper right-hand side of each plot represents the outliers. Both Mahalanobis and MAD identified one of the two points as outliers, while the other point fell just along the edge of each algorithm's threshold for outliers. Cook's Distance and Weighted Mean labelled both points as outliers. From these plots, it can be seen that MAD and Cook's Distance are very focused on absolute distance from the origin in their detection of outliers while Mahalanobis and Weighted Mean appear to look at more relative distances. It can already be seen here that Weighted Mean identifies fewer datapoints as outliers and, therefore, tended to return fewer false positives. Parameters on the other algorithms could be adjusted to increase their sensitivity to outliers, but the trade-off was an increased number of false positives.

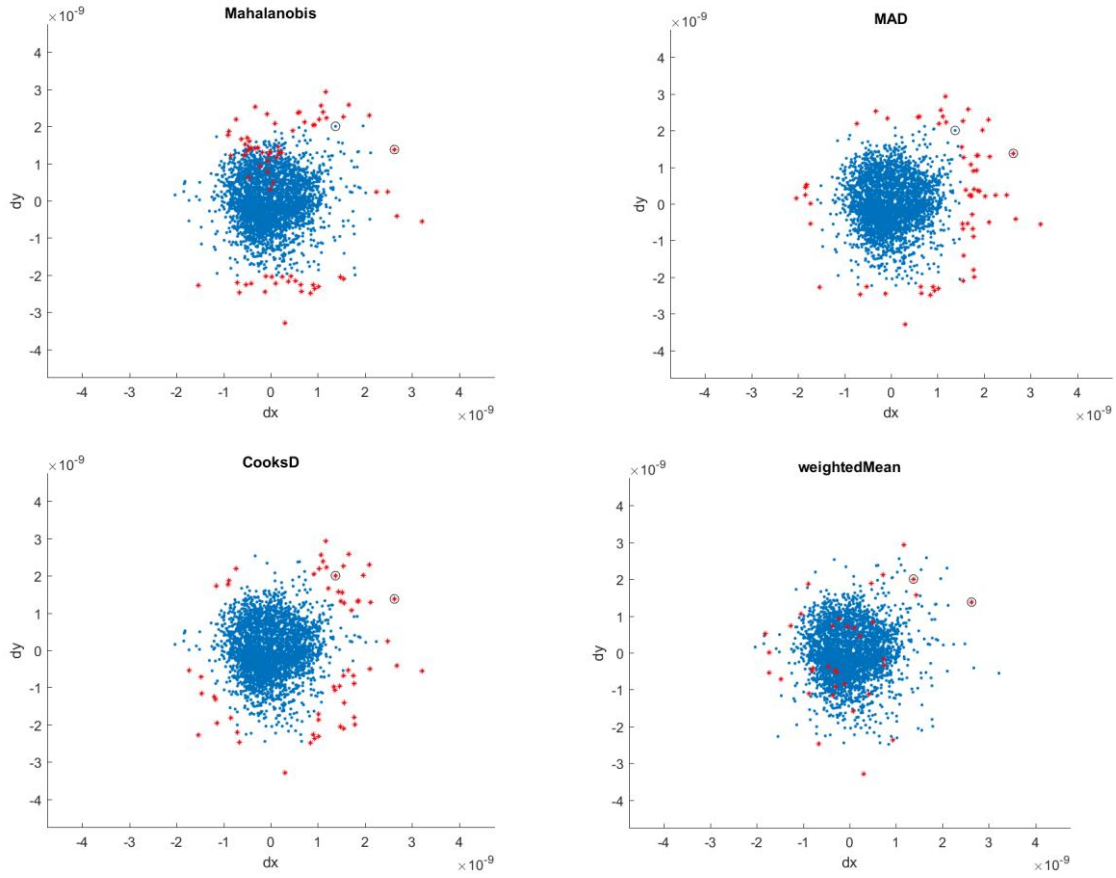


Figure 4.1 – Plot of Algorithm Sensitivities

In the next evolution of testing, Cook's Distance by Neighborhood was added. This algorithm broke areas of the wafer down into small neighborhoods and Cook's Distance was repeatedly run on the small subsets of data. Table 4.*2* shows a randomly chosen example of one simulation and the successes of the various outlier detection algorithms, along with Cook's Distance by Neighborhood, are shown over the simulation. The first row is the shortest length of the outliers at one picometer, and the tenth row represents the outlier with the longest length of four nanometers. The algorithms score a "1" if the artificial outlier is found, or a "0" if it is not found. The sum of successes is not entirely relevant because a successful find of an outlier at 1 picometer is more important than success at 4 nanometers.

Artificial Outlier Length	*Mahalanobis*	*MAD*	*Cook's D*	*Cook's D Neighborhood*	*Weighted Mean*
Results from a single simulation					
1.0000E-12	0	0	1	1	1
4.4533E-10	0	1	1	1	1
8.8967E-10	1	0	1	1	0
1.3340E-09	0	1	1	1	1
1.7783E-09	1	1	1	1	1
2.2227E-09	1	1	1	1	1
2.6670E-09	1	1	1	1	1
3.1113E-09	1	1	1	1	1
3.5557E-09	1	1	1	1	1
4.0000E-09	1	1	1	1	1
SUM:	7	8	10	10	9

Table 4.2 – *Success in Detecting Outliers "By Wafer"*
The number 1 indicates the outlier was successfully found while 0 indicates the outlier was not found.

In this particular simulation, it appears that the outlier with the third shortest length was placed in a challenging area of the wafer. The miss by Mahalanobis Distance of the outlier with the fourth shortest length is concerning, as outliers of that length were expected to be reliably found at every simulation. Generally, finding the smallest outlier was considered an accomplishment, but missing a larger one that the other algorithms found represents a greater negative than the positive of finding the smallest outlier. Only Cook's Distance manages to find all ten outliers in both the full wafer and the by neighborhood approaches.

Figure 4.2 below provides additional evidence that Cook's Distance by Neighborhood could be the most effective algorithm. Cook's Distance by Neighborhood produces deltas of 0.03 nm for *x* and 0.04 nm for *y,* compared to 0.08 nm and 0.10 nm respectively, for the Weighted Mean algorithm. As previously noted, the delta is calculated by subtracting the reference machine fingerprint from the estimated machine fingerprint. The reference machine fingerprint is the average of the datapoints on each of the four wafers before any artificial outliers are added. The estimated machine fingerprint is the average of the datapoints on each of the four wafers after the artificial outliers have been added but also after the datapoints identified by the algorithm as an outlier are deleted. This means the deltas will be longer where there are remaining outliers and false positives. The overall delta of the Cook's Distance "By Wafer" approach was worse than the Cook's Distance "By Neighborhood" approach because Cook's Distance was more aggressive in identifying datapoints as outliers and ends up with many false positives which cause greater deltas. The neighborhood approach limits the number of false positives since the lengths of the vectors are more similar in the subsets of datapoints.

Figure 4.2 – Deltas for "By Wafer" Approach

Delta is calculated by subtracting the Reference Machine Fingerprint from the Estimated Machine Fingerprint. Vectors remaining on the wafers after subtraction are a result of the algorithm failing to detect an outlier or due to false positives that were removed. Delta is calculated by summing the remaining vectors.

Figure 4.3 and Figure 4.4 show where all algorithms underperformed the Weighted Mean. The maximum x metric is the maximum delta within the x data at 3-sigma. The maximum *y* metric is the same measurement for the *y* data. Shorter deltas are superior as they indicate that the estimated machine fingerprint that is returned after outlier detection is more similar to the reference machine fingerprint. All outlier detection algorithms have maximum x's and y's that are higher than the maximum values seen in the Weighted Mean data.

In Figure 4.3 the y-axis shows the length of the maximum absolute value of the *x* deltas (and *y* deltas for and Figure 4.4). The x-axis for both Figure 4.3 and Figure 4.4 represents the length of the outlier. There are 3 values for each outlier length on the plot, as 3 simulations are being plotted simultaneously. A snapshot of the remaining deltas was taken for each of the 10 outliers after the outlier detection was run.

To best understand these plots, view the final plot entitled "No Outlier Method Applied." On this plot at the final datapoint for outliers of length 4 nm it can be seen that the values for each of the three simulations is between 0.8 and 1.2. Since no outlier detection was applied, we know that the 4 nm outlier remains on wafer #1 which was averaged with wafers 2, 3 and 4. We see that the average max delta at zero is ~0.2, so when wafer #1 is averaged with wafers 2, 3, and 4 we get (4.0 + ~0.2 + ~0.2 + ~0.2) / 4 = ~1.15. Similar calculations can be done for the smaller outliers to understand why the plot shows increasing values of the max x (and max y). For both Max x and Max y, the plotted points for the Weighted Mean algorithm remain consistently below 0.2 nm at all outlier lengths, while the other algorithms show higher values.

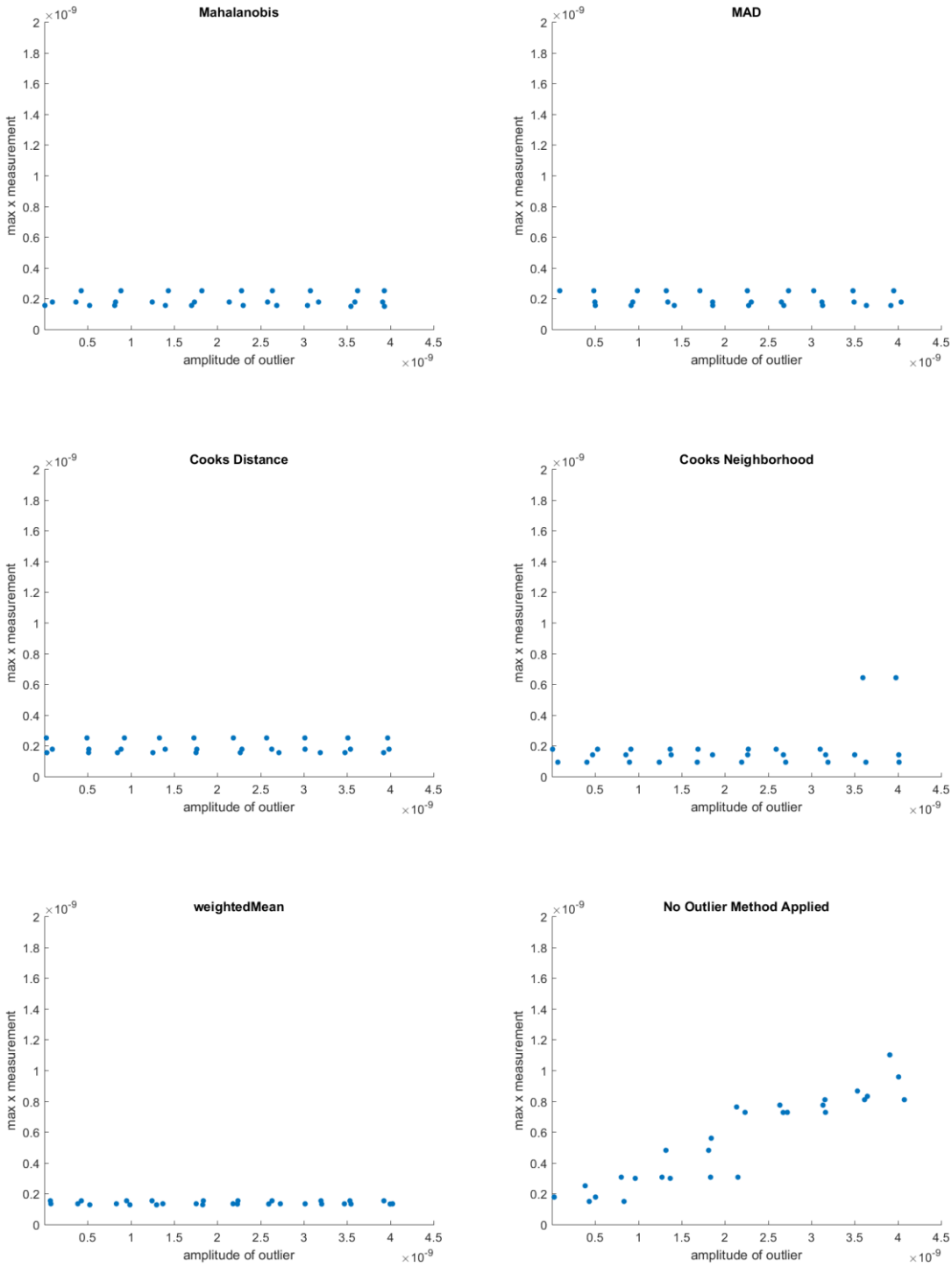


Figure 4.3 – Maximum absolute *x* values of the delta

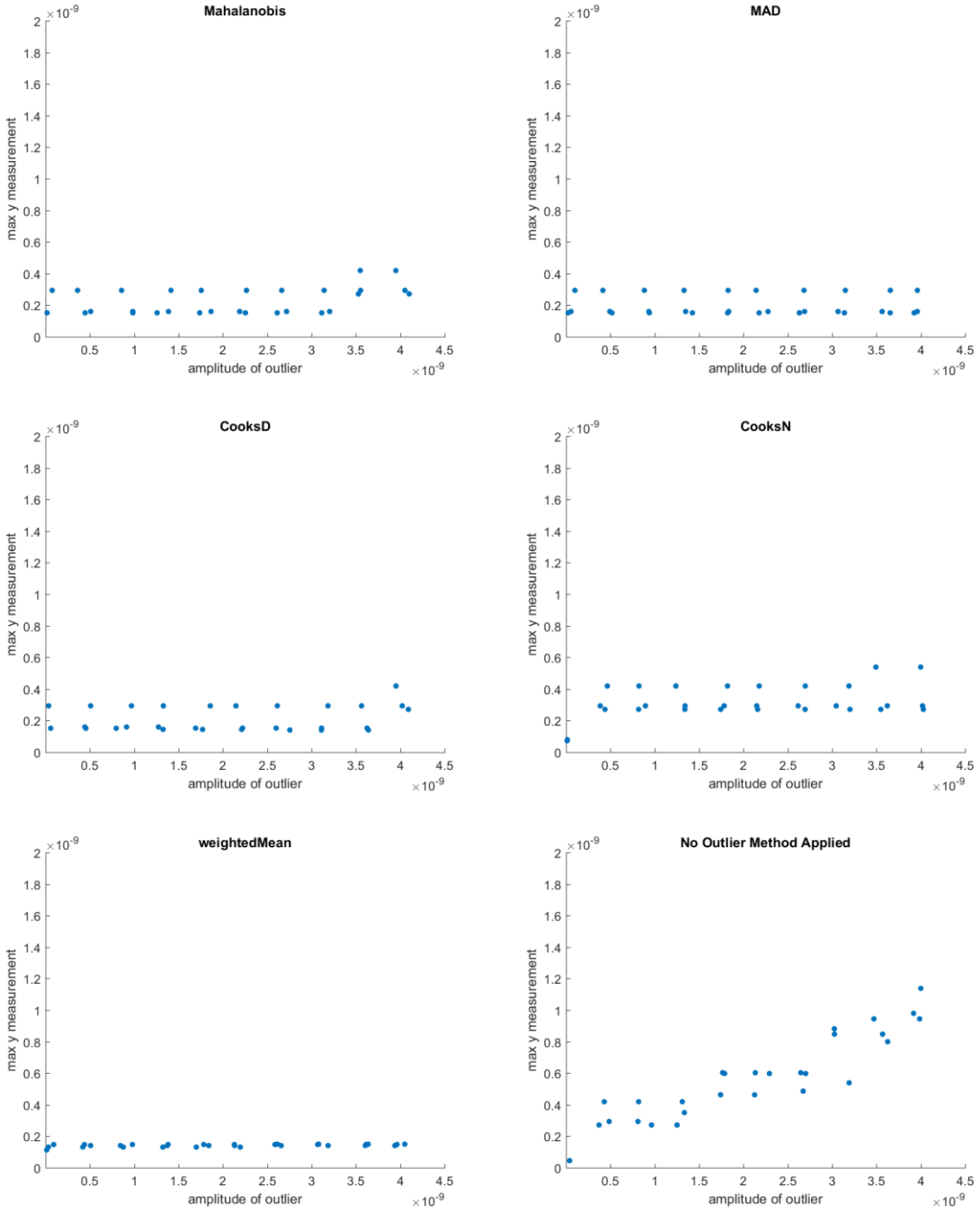


Figure 4.4 – Maximum absolute *y* values of the delta

Figure 4.5 shows the mean 3-sigma values for *x*. The Weighted Mean algorithm underperformed all algorithms for this metric, with its higher values across each outlier length. (The mean 3-sigma measures the average delta after the values are ordered by absolute value and the largest 0.3% are removed. The x and y values are calculated separately to determine Mean 3-sigma X

and Mean 3-sigma Y. The Weighted Mean underperformed on both of these measures.) The reason for the higher mean 3-sigma values for the Weighted Mean is due to the difference in the calculation of the mean and Weighted Mean. The difference in calculations results in the Weighted Mean approach always having small deltas for every field. This can easily be seen Figure 4.5. These small differences add up to a higher overall delta and average delta – even after the largest 0.03% of absolute values are removed.

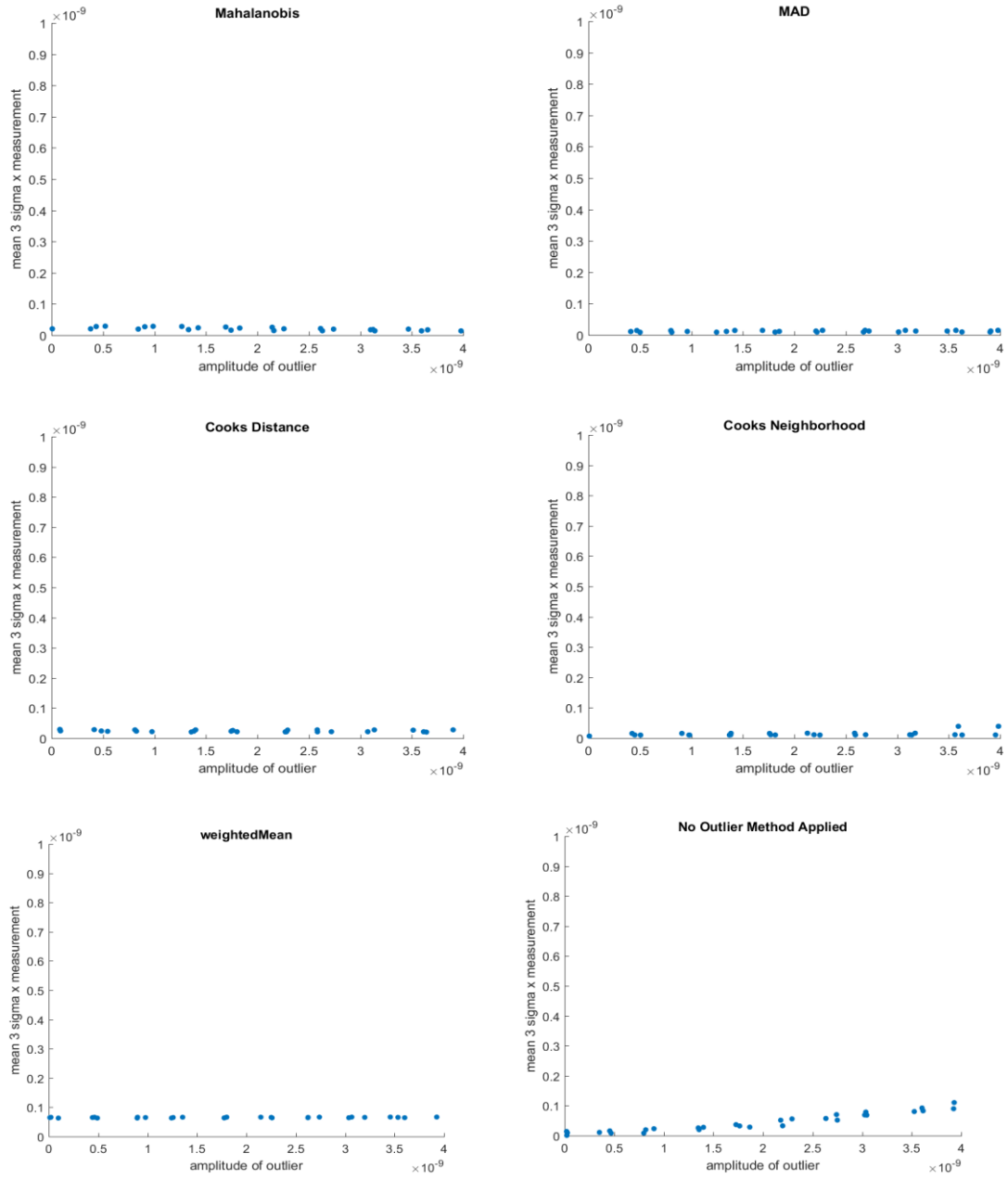


Figure 4.5 – Mean 3-Sigma *x*

4.4 "Across Wafers" Approach Sample Results

Below in Table 4.3 are the results from 2 simulations with the "Across Wafers" approach. This approach uses four datapoints – one from each wafer for each corresponding field. MAD returns the same results as the Weighted Mean in these simulations. These results also provide a glimpse at how MAD and Cook's Distance could work in conjunction as an ensemble algorithm with one algorithm detecting the outlier where the other does not.

Artificial Outlier Length	*Mahalanobis*	*MAD*	*Cook's D*	*Weighted Mean*
		Simulation 1		
1.000000E-12	0	0	0	0
4.453333E-10	0	1	0	1
8.896667E-10	1	1	1	1
1.334000E-09	1	1	1	1
1.778333E-09	1	1	1	1
2.222667E-09	1	1	1	1
2.667000E-09	1	1	1	1
3.111333E-09	1	1	1	1
3.555667E-09	1	1	1	1
4.000000E-09	1	1	1	1
SUM:	**8**	**9**	**8**	**9**
		Simulation 2		
1.000000E-12	0	1	0	1
4.453333E-10	0	0	1	0
8.896667E-10	1	1	1	1
1.334000E-09	1	1	1	1
1.778333E-09	1	1	1	1
2.222667E-09	1	1	1	1
2.667000E-09	1	1	1	1
3.111333E-09	1	1	1	1
3.555667E-09	1	1	1	1
4.000000E-09	1	1	1	1
SUM:	**8**	**9**	**9**	**9**

Table 4.3 – Success in Detecting Outliers "Across Wafers"

Similar to the "By Wafer" approach, in the "Across Wafers" approach all algorithms outperform Weighted Mean on the mean 3-sigma measurements for both *x* and y, but underperform Weighted Mean for the Max x and Max *y* metrics. The results for the delta metric were mixed. MAD outperformed the Weighted Mean delta but Mahalanobis underperformed it and Cook's Distance returned deltas similar to the Weighted Mean.

4.5 Final Results with the Ensemble Algorithm

As can be seen in the intermediate results in the previous sections, several algorithms compared well with the WEC team's Weighted Mean outlier detection algorithm but none managed to beat the Weighted Mean on all metrics. Using what was learned by the neighborhood approach with Cook's Distance and seeing the potentially collaborative relationship between MAD and Cook's Distance in successful outlier identification, the ensemble algorithm was created. With two separate runs of 100 simulations, each run on different out-of-sample datasets, the ensemble model showed an average score of ~9.4 while Weighted Mean had an average of ~9.05. The ensemble algorithm had a success rate for the first run that was 3.5% higher than the Weighted Mean. For the second run, the ensemble algorithm's success rate was 4% higher than that of the Weighted Mean. Table 4.4 shows these results.

RECALL

	Ensemble Algorithm	*Weighted Mean*
Run 1	941	908
New Dataset 1	94.10%	90.80%
Run 2	940	902
New Dataset 2	94%	90.20%

Table 4.4 – Recall over 2 runs of 100 simulations each

Each run of 100 simulations was made with separate datasets that had never been used before. 10 artificial outliers were placed in each simulation; yielding a total of 1,000 outliers for each run. For Run 1, the ensemble algorithm found 941 of the 1000 outliers for a recall, or success rate, of 94.1%. It performed consistently in the second run with 94% recall.

The ensemble algorithm outperformed the Weighted Mean again in terms of the delta measurement. It had a much lower delta, 0.01nm for both *x* and *y*, while Weighted Mean had a delta of 0.08nm for *x* and 0.09nm for *y*. (Originally, the ensemble algorithm was named "Overlay," so that is the name in the label that appears in Figure 4.6 for the ensemble algorithm in the following plots.)

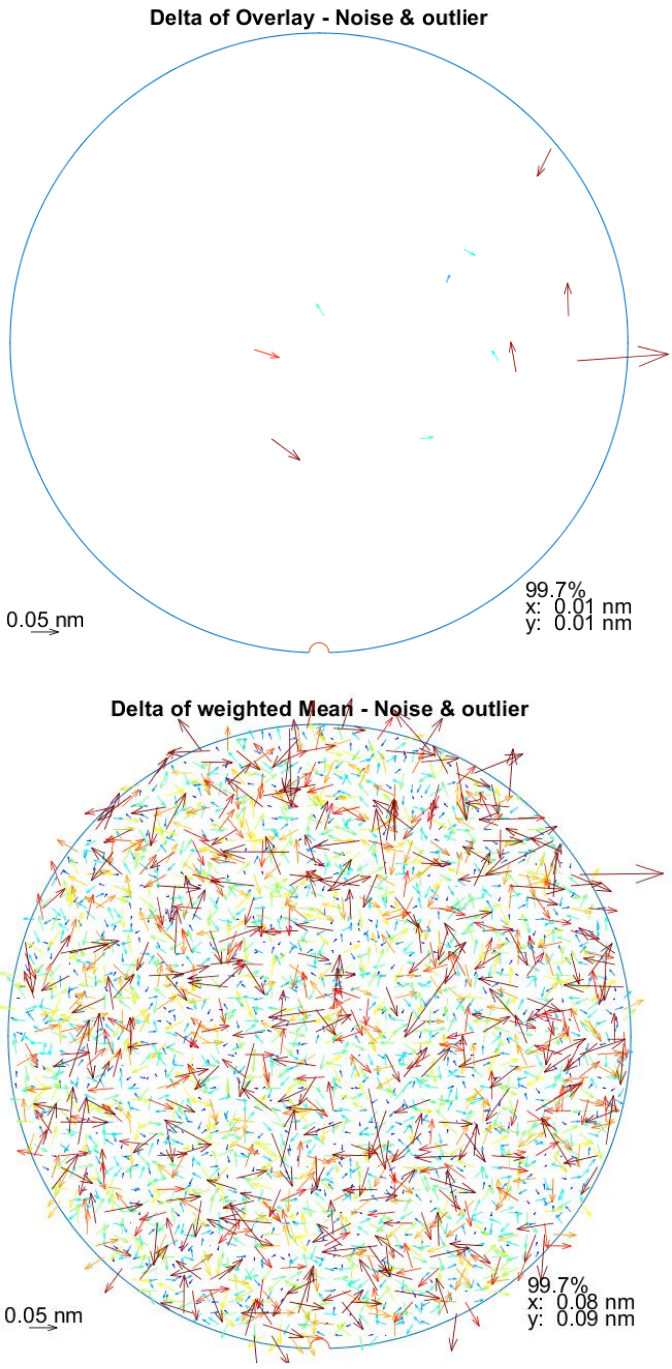


Figure 4.6 – Comparison of the Deltas for the Ensemble Model and the Weighted Mean

The ensemble algorithm ("Overlay") has a lower (better) delta than the Weighted Mean, as can be seen in the x and y values on the lower, right-hand side of each vector plot.

On the metric that been the most difficult to beat the Weighted Mean, Max *x* and Max *y*, the ensemble algorithm outperformed the Weighted Mean with a lower maximum value for both as shown in Figure 4.7 and Figure 4.8. The ensemble algorithm came in lower at ~.205nm versus the Weighted Mean's ~.210 nm for *x*. For *y*, the ensemble returned ~.20 nm and the Weighted Mean returned ~.21 nm over 100 simulations.

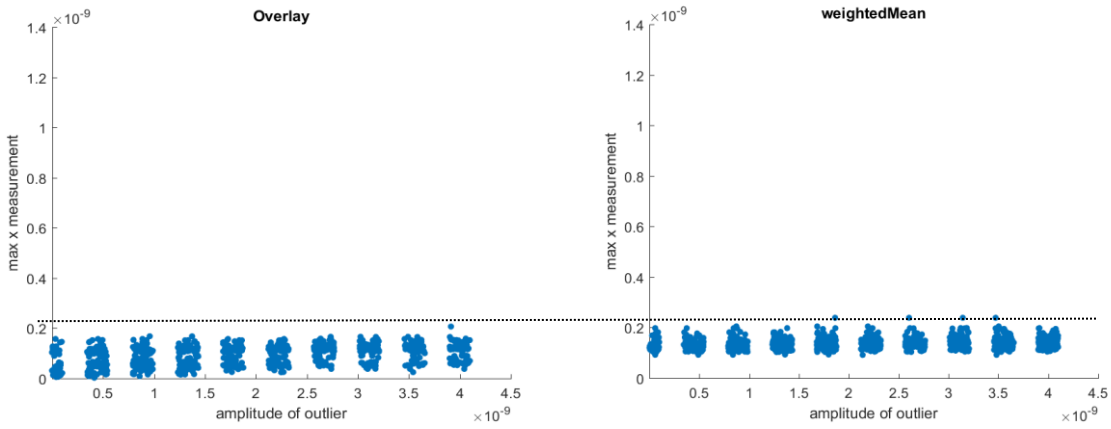


Figure 4.7 – Maximum absolute *x* values, final comparison

The dotted line above shows the values for the ensemble algorithm (left, with the title "Overlay") coming in with lower (better) values than then Weighted Mean algorithm.

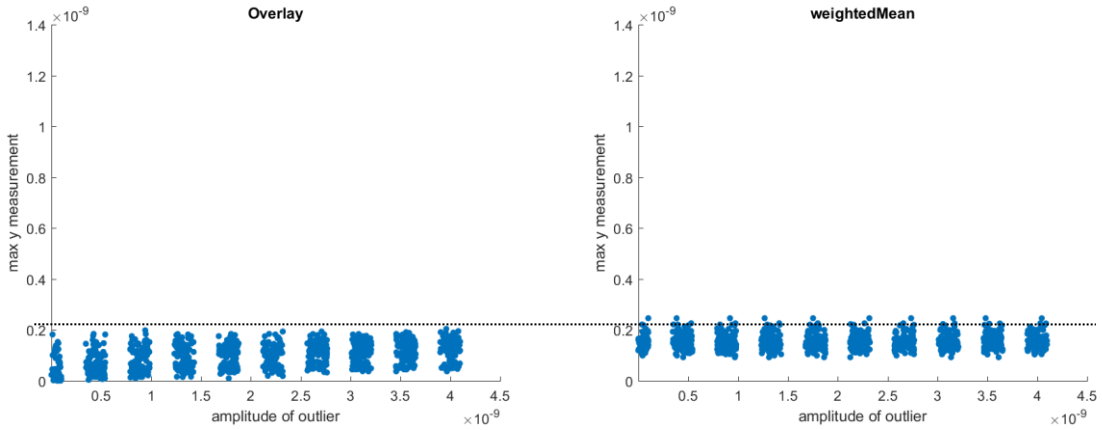


Figure 4.8 – Maximum absolute *y* values, final comparison

The dotted line above shows the values for the ensemble algorithm (left, with the title "Overlay") coming in with lower (better) values than then Weighted Mean algorithm.

The ensemble algorithm outperformed the Weighted Mean on the mean 3-sigma metric with ~.19 nm for *x* vs. the Weighted Mean's ~.6 nm. In the case of y, the ensemble algorithm returned ~.19 nm again while the Weighted Mean returned ~.64 nm over 100 simulations. These results can be seen in Figure 4.9 and Figure 4.10.

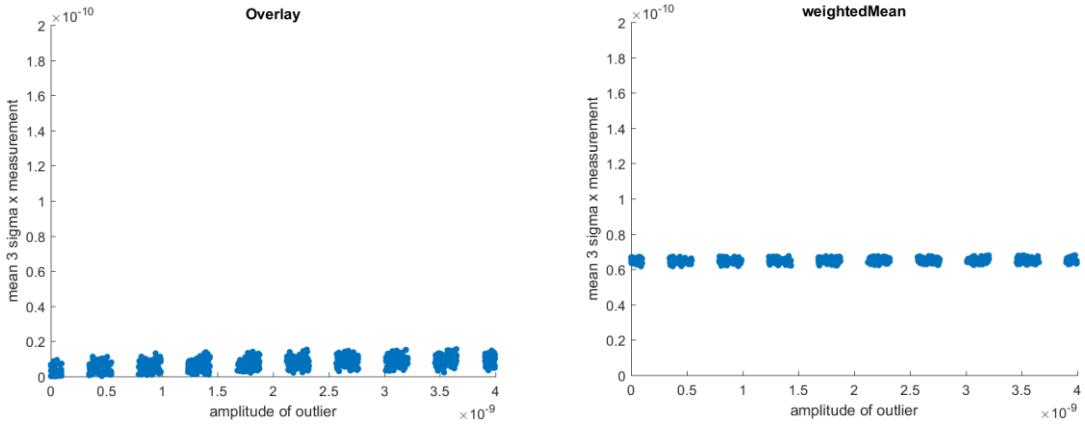


Figure 4.9 – Mean 3-Sigma *x*

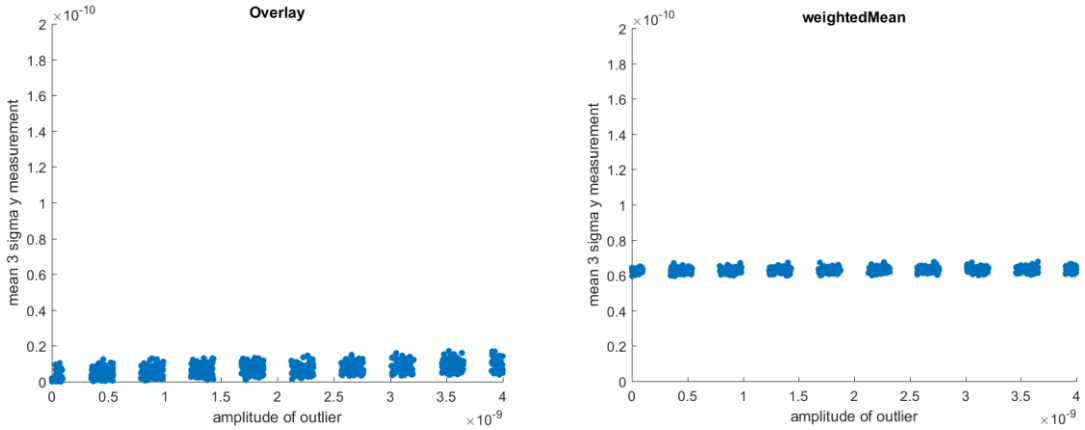


Figure 4.10 – Mean 3-Sigma *y*

The ensemble algorithm returned an average accuracy of 99.86% vs. 99.76% for the Weighted Mean over two runs of 100 simulations. Run 1 and Run 2 were completed on different out-of-sample datasets. The two runs are shown below in Table 4.5.

ACCURACY

	Ensemble Algorithm	*Weighted Mean*
Run 1	99.90%	99.76%
Run 2	99.81%	99.76%

Table 4.5 – Accuracy

Finally, for precision the ensemble algorithm returned an average of 74% over two runs of 100 simulations, shown in Table 4.6, and the Weighted Mean returned and average of 60.5%. This is a measurement of the proportion of those identified as outliers that were correct. The difference in Run 1 and Run 2 was due to a higher number of false positives in the second run. This may have been due to the fact the each of these runs were on different datasets and the machine fingerprint of the first dataset proved less challenging to the algorithm.

PRECISION

	Ensemble Algorithm	*Weighted Mean*
Run 1	82.05%	60.69%
Run 2	66.03%	60.41%

Table 4.6 – Precision

Run 1 and Run 2 were completed on different datasets. The higher precision on Run 1 was due to it having fewer false positives than in Run 2. The dataset for Run 1 may have contained much less noise.

Chapter 5 | Discussion

The results for the ensemble algorithm versus the Weighted Mean algorithm show that this hybrid approach leads to increased performance in outlier detection at all measures. As noted, several times in the results chapter, a key element in the ensemble algorithm's outperformance was its ability to reduce false positives. This was a challenging task since reducing false positives and increasing outlier detection capabilities are opposing forces. Adjusting an algorithm's parameters to increase its sensitivity to outliers typically means allowing more false positives to be returned.

The ensemble algorithm overcame this challenge by filtering the data down to "suspicious indices" in the first step of each "wave and by using the neighborhood approach for Cook's Distance which was another instance of subsetting." Executing the algorithm over this subset of data led to the reduction in false positives while using two different outlier detection algorithms that were not perfectly correlated in their ability to identify outliers enabled the detection performance to remain high.

The ability of filtering and subsetting the data to reduce false positives and the ensembling of algorithms with high performance, but some dissimilarity in their outlier detection abilities, are key insights. The results were well-received by ASML's ORC team and provide a new potential testing direction to build from and iterate further. Further improvements in outlier detection by the team will be one of the many factors that help drive improvement in overlay at ASML.

The team did point to one limitation of the ensemble algorithm and that was the potential need to tune the parameter that chooses the threshold for subsetting the data into "suspicious indices." As stated in section 3.8.1, the algorithm was hardcoded to use the 99^{th} percentile as the cutoff threshold in testing. This percentile was determined by tuning over multiple simulations. A different dataset could, however, potentially perform better with a different level for the parameter. Nonetheless, in testing over two out-of-sample datasets the algorithm continued to perform well with the parameter hardcoded at the 99^{th} percentile, as is clearly shown by the results.

One surprise in the process was that switching to a “By Wafer” approach from the firm’s existing “Across Wafers” approach did not yield the improvements expected from the increased amount of data. (3,037 datapoints vs. 4 datapoints) In fact, the results were quite similar for both approaches. The new approach did, however, did lead to the Neighborhood approach and that opened up new avenues for investigation, namely, the angles of the vectors in a particular neighborhood and the rank of the length of the vectors.

With more time, it would be interesting to test other combinations of algorithms, including some of the algorithms that were discarded earlier in the analysis. Adding elements like subsetting and building neighborhoods could potentially improve their performance and create an ensemble algorithm superior to the one created in this research project.

Chapter 6 | Conclusions

This research set out to answer the question of how well the ASML WEC team's current outlier detection algorithm compared in performance to other outlier detection algorithms. Based on the simulation results, the WEC team's Weighted Mean algorithm performs very well in comparison to the other off-the-shelf outlier detection algorithms. While the other algorithms were able to beat the Weighted Mean on some performance metrics, none were able to beat it on all metrics (particularly not on the success in detecting outliers). A strong argument cannot be made for the WEC team to change their current outlier detection algorithm to an off-the-shelf outlier detection algorithm.

The research followed up the previous question to ask if a superior algorithm could be custom-built using what was learned in the benchmarking process. Based on the results of the simulations for the newly created ensemble outlier detection algorithm in comparison to the Weighted Mean algorithm, the answer is "yes," it is possible to achieve better performance on every metric with a custom algorithm. It was also shown with testing on out-of-sample data that, although the algorithm is highly customized to findings made in benchmarking on a testing dataset, the ensemble algorithm's improved performance generalizes across to out-of-sample datasets.

The methodology used in benchmarking played an important role in building the insights to develop the ensemble algorithm. The initial, although problematic, approach of subtracting the wafer averages to remove the machine fingerprint described at the end of section 3.6.1 played a role in developing the idea to filter and subset the data to suspicious indices for easier detection of outliers. The use of neighborhoods developed in the "By Wafer" approach led to the realization that subsetting reduced false positives. Viewing the results of the benchmarking process also led to the discovery that MAD and Cook's Distance were failing and succeeding at different points on the short end of outlier lengths and might be combined to improve outlier detection.

This project gives a small window of insight into the research ASML conducts across all of its units to make incremental improvements. Insights such as those found in improving outlier detection with the ensemble algorithm enables ASML to continually improve performance at the nanometer scale and maintain its position as a global leader in the lithography industry.

BIBLIOGRAPHY

Aggarwal, C. C. (2013). Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter*, *14*(2), 49-58.

Arnold, B. C., Balakrishnan, N., & Nagaraja, H. N. (1992). *A first course in order statistics* (Vol. 54). Siam.

ASML: ASML at a Glance. (2019) Retrieved from https://www.asml.com/en/company/about-asml/asml-at-a-glance.

Bock, Hans-Hermann, (2008). Origins and extensions of the k-means algorithm in cluster analysis. Journal Electronique d'Histoire des Probabilités et de la Statistique Electronic Journal for History of Probability and Statistics, 4(2).

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000, May). LOF: identifying density-based local outliers. In *ACM sigmod record* (Vol. 29, No. 2, pp. 93-104). ACM.

Cerioli, A. (2010). Multivariate outlier detection with high-breakdown estimators. *Journal of the American Statistical Association*, *105*(489), 147-156.

Chauvenet, W. (1863). A manual of spherical and practical astronomy, 2 vols.

Clarke, P. (2018, February 12) ASML increases dominance of lithography market. Retrieved from https://www.eenewsanalog.com/news/asml-increases-dominance-lithography-market

Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, *19*(1), 15-18.

Dixon, W. J. (1951). Ratios involving extreme values. *The Annals of Mathematical Statistics*, *22*(1), 68-78.

Englund, C., & Verikas, A. (2005). A hybrid approach to outlier detection in the offset lithographic printing process. *Engineering applications of artificial intelligence*, *18*(6), 759-768.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, No. 34, pp. 226-231).

Gauss, C. F. (1816). Bestimmung der genauigkeit der beobachtungen. *Abhandlungen zur Methode der kleinsten Quadrate, 1887*.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics*, *11*(1), 1-21.

Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, *22*(2), 85-126.

Hubert, M., Debruyne, M., & Rousseeuw, P. J. (2018). Minimum covariance determinant and extensions. *Wiley Interdisciplinary Reviews: Computational Statistics*, *10*(3), e1421.

Kriegel, H. P., Schubert, M., & Zimek, A. (2008, August). Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 444-452). ACM.

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *6*(1), 3.

Mack, C.M. (2006) Lithography: Semiconductor Lithography (Photolithography) – The Basic Process Retrieved from http://www.lithoguru.com/scientist/lithobasics.html.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. National Institute of Science of India.

Peirce, B. (1852). Criterion for the rejection of doubtful observations. *The Astronomical Journal*, *2*, 161-163.

Rousseeuw, P. J. Multivariate estimation with high breakdown point (1985) Mathematical Statistics and Applications. *Reidel Publishing, Dordrecht*, 283-297.

Rousseeuw, P. J., and Leroy, A. M., (1987). Robust regression and outlier detection. John Wiley & Sons, Inc. New York, NY, USA.

Rousseeuw, P. J., & Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical association*, *88*(424), 1273-1283.

Spear, Mary Eleanor (1952). *Charting Statistics*. McGraw Hill. p. 166.

Susto, G. A., Beghi, A., & McLoone, S. (2017, May). Anomaly detection through on-line isolation Forest: An application to plasma etching. In *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)* (pp. 89-94). IEEE.

Tukey, J. W. (1977). Exploratory data analysis. *Reading: Addison-Wesley*.

Walczak, B., & Massart, D. L. (1998). Multiple outlier detection revisited. *Chemometrics and intelligent laboratory systems*, *41*(1), 1-15.

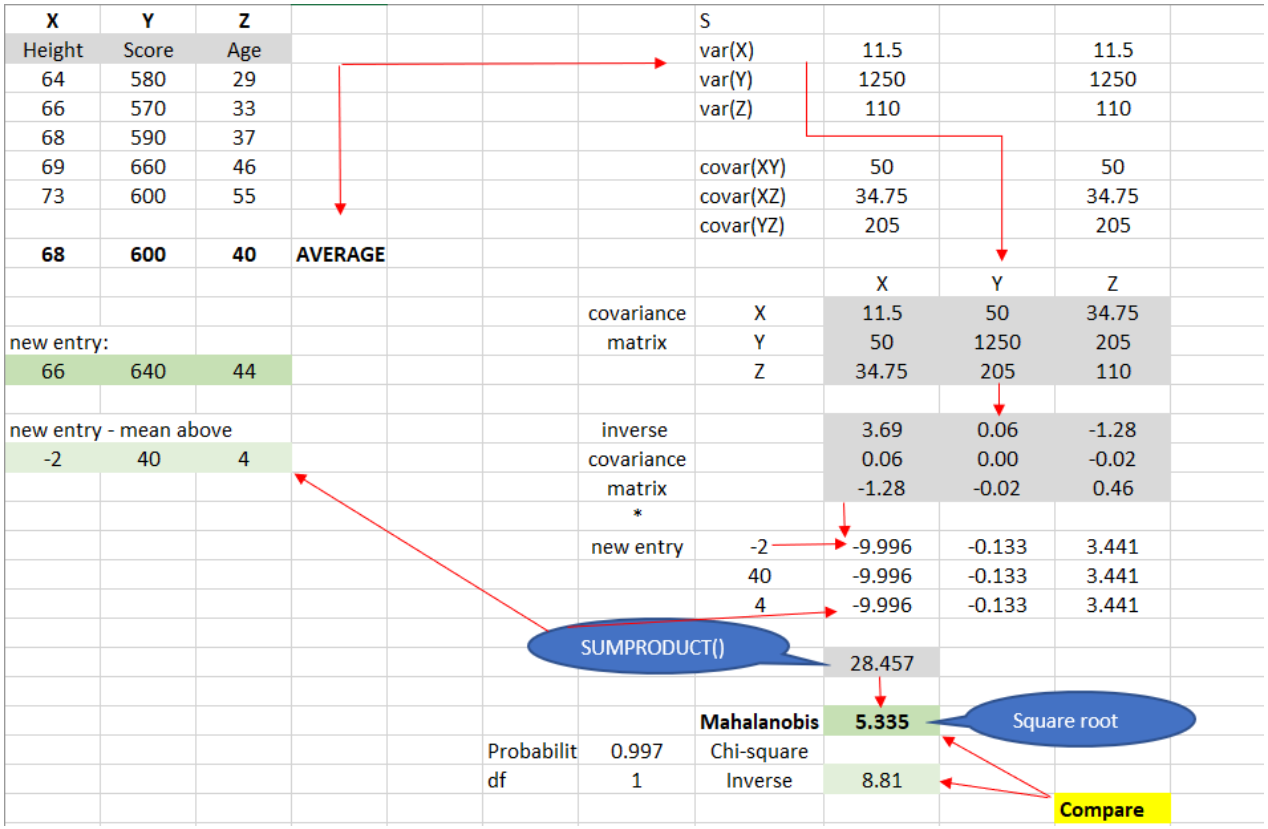
Zimek, A., Campello, R. J., & Sander, J. (2014). Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *Acm Sigkdd Explorations Newsletter*, *15*(1), 11-22.

Intentionally left blank

APPENDIX

MAHALANOBIS DISTANCE

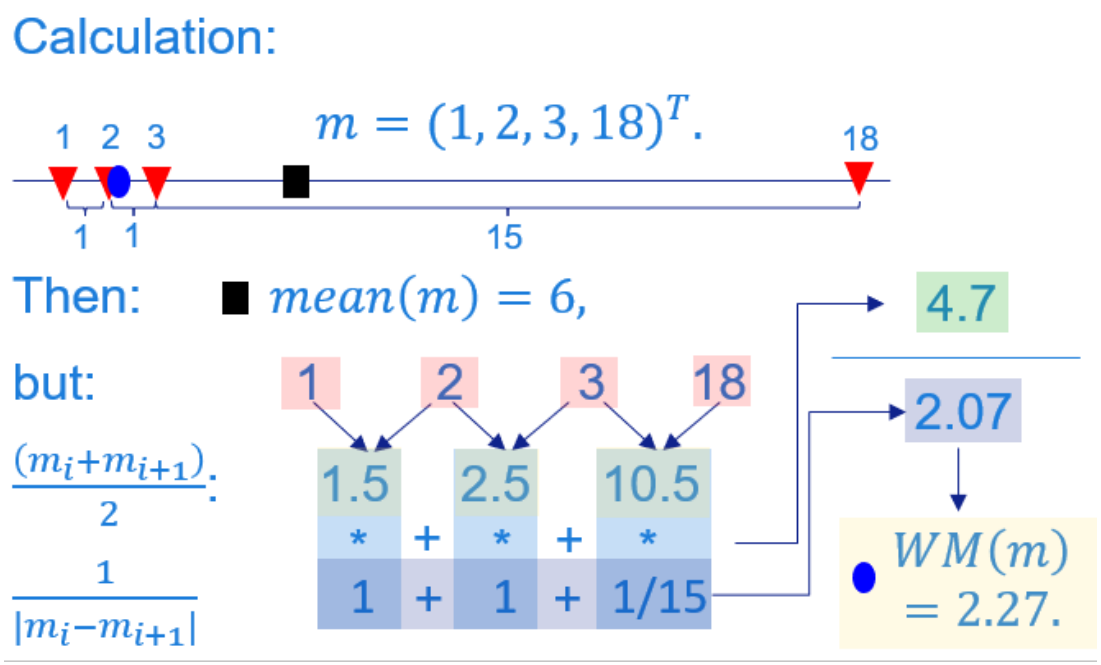
Here is an example of the Mahalanobis distance calculation:



WEIGHTED MEAN

Here is an example of the ASML WEC team's weighted mean calculation:

GIVEN: Four data points: 1, 2, 3, and 18



Mean: 6 · **Median:** 2.5 · **Weighted Average:** 2.27

Local Outlier Factor (LOF)

Here is an example of the Local Outlier Factor calculation:

Example:

Points A B C D

(0,0) (0,1) (1,1) (3,0)

k = 2

Step 1: calculate the **distances** (Euclidean used here, but you can use your favorite)

Step 2: calculate **k_distance(p)** --> $dist_2(x,y)$

Step 3: calculate the **reachability distance** --> $reach\_dist_k(p,o) = max\{k\_distance(o), d(p,o)\}$

Step 4: calculate the **local reachability density of p** --> lrd(p)

Step 5: calculate the **local outlier factor** --> LOF

		$dist_2(x,y)$	rank	k_distance(p) (k=2, so rank 2)		Reachability distance (uses rank1 & rank2 results) k_distance(p)	$dist_2(x,y)$	max(k_distance, $dist_2(x,y)$)	Local Reachability Density			Local Outlier Factor			
	(A,B)	1	1		b	1	1	1.00	(a,b)	= k / (reach_dist2(a,b) + reach_dist2(a,c)		= ((lrd_2(b) + lrd_2(c)) / lrd_2(a)) / k			
a	(A,C)	1.41	2	1.41	c	1.41	1.41	1.41	(a,c lrd_2(a)	=2 / (1 + 1.	0.828	=((0.709 + 0.829) / (0.829)) / 2	0.928	≈ 1	
	(A,D)	3	3												
	(B,A)	1	1&2		a	1.41	1	1.41	(b,a)	= k / (reach_dist2(b,a) + reach_dist2(b,c)		= ((lrd_2(a) + lrd_2(c)) / lrd_2(b)) / k			
b	(B,C)	1	1&2	1	c	1.41	1	1.41	(b,c lrd_2(b)	=2 / (1.41 ·	0.709	=((0.829 + 0.829) / (0.709)) / 2	1.168	≈ 1	
	(B,D)	3.16	3												
	(C,A)	1.41	2		a	1.41	1.41	1.41	(c,a)	= k / (reach_dist2(c,a) + reach_dist2(c,b)		= ((lrd_2(a) + lrd_2(b)) / lrd_2(c)) / k			
c	(C,B)	1.00	1	1.41	b	1	1.00	1.00	(c,b lrd_2(c)	=2 / (1 + 1.	0.828	=((0.829 + 0.709) / (0.829)) / 2	0.928	≈ 1	
	(C,D)	2.24	3												
	(D,A)	3.00	2		a	1.41	3.00	3.00	(d,a)	= k / (reach_dist2(d,a) + reach_dist2(d,c)		= ((lrd_2(a) + lrd_2(c)) / lrd_2(d)) / k			
d	(D,B)	3.16	3	3					lrd_2(d)	=2 / (3 + 2	0.382	=((0.829 + 0.829) / (0.382)) / 2	2.169	≈ 2	outlier
	(D,C)	2.24	1		c	1.41	2.24	2.24	(d,c)