Investigating the Automated Detection of Faulty Semi-Conductors

Nate Hofmann

April 25, 2020

Abstract

In modern semi-conductor manufacturing processes, losses in the yield of silicon wafers remains a problem due to excursions occurring early in the assembly process - but not being detected until much later in production. In addition, the semi-conductors being assembled are under constant monitoring through the continuous collection of signals and specific measurement points. There is value in investigating if such yield excursions can be identified earlier using the accumulated signals and measurement points. Value may also be derived from the perspective of machine learning due to the approaches taken to address the noisy, imbalanced, low sampled, and high dimensional nature of the examined data.

1 Introduction

In spite all of the advances made in modern semi-conductor manufacturing processes, the theoretical maximum yield of silicon wafers continues eluding a solution elusive due to parts of the process shifting out of specification at the start of assembly process - but remaining unobserved until much later in the assembly procedure. If these yield excursions were to be caught earlier, there would notable increases in process throughput, decreased time to learning, and reduction in per unit production costs to be realized. It is also worth mentioning that such processes are almost always under persistent watch via the uninterrupted collection of semi-conductor signals and measurement points. Therefore using these collected signals and measurements, there is value in exploring if they can be utilized to detect yield excursions earlier.

In addition, the data to be examined for this analysis provides a unique combination of challenges to be dealt with. Specifically the collected data is noisy due to random errors introduced in its measurement and processing, as well as a large number of collected signals and measurement points for each sample. In addition the data features a low sample count, as well as a heavy skewing towards non-faulty semi-conductors (those without yield excursions). Attempting to solve the aforementioned detection problem in the presence of these hurdles presents a unique case study of various machine learning techniques in conjunction with each other.

1.1 Accreditation

The problem and dataset this paper examine are from the UCI Machine Learning Repository [1]. All technical analysis was accomplished with the Python machine learning library Scikit-Learn [2].

2 Technical Approach

2.1 Dataset Description

This investigation was framed as a binary classification problem consisting of 1567 semiconductors and their 591 collected, unnamed signals and measurement points. Of the semi-conductors, 104 were marked as faulty (experienced a yield excursion, or positive samples) and 1463 as working (or negative) samples. Around one percent of all feature values were labelled as missing. Every feature is a real valued number assumed to be drawn from a normal distribution - as well as being independent of every other feature.

2.2 Preprocessing

Before the modelling of this dataset began, all constant features (feature columns whose standard deviation was zero) were removed - reducing the number of features from 591 to 477. Missing values filled in with the mean value of the feature column that value was apart of. For the duration of this paper, feature columns will be referenced by their column index, a number in the range of 0 to 476.

2.3 Experimental Setup

The binary classification problem was examined under the lens of four different models of increasing expressiveness: a logistical regression model, a naive Bayes classifier, a support vector classifier (SVC), and a neural network. However due to extremely long training times and lacking early trials - the few tests taken with neural networks will not be elaborated on.

Due to the limited amount of data available for training, model examination and hyperparameter tuning occurred using nested *k*l*-fold cross validation. The data was split into *k=10* outer folds where each fold was used as a test set with the other *k-1* folds combined into a training set - giving 10 separate instances of model evaluation. During each instance of training, each training set was further split into *l=5* folds to perform cross validated grid search on a given set of hyperparameters to. The best performing hyperparameters from these inner sessions of cross validation were then used as the hyperparameters for the model in outer instance of training. The chosen values of *k* and *l* were selected due to *k=10* being a commonly used value of *k* across multiple domains, and a value of *l=5* to keep size of inner training and test sets at substantive level.

To prevent training on unrepresentative data - due to the skewed nature of the dataset in favor of negative samples over positive ones, stratified cross validation was employed was employed for all sessions of inner and outer cross validation. During each splitting of the dataset (or a subset of the dataset) at the start of each instance of cross validation, each created fold was split such that it maintained a ratio of classes matching that of the whole dataset.

To assist the training of each model, z-score standardization was applied to each set of training data along each feature column. Standardization using the means and standard deviations of each feature column derived from the training set was also applied to the corresponding training sets.

Two other techniques were applied to boost learning on an imbalanced dataset. In the presence of imbalanced data, the scoring metric normally used for binary classification - accuracy - fails to bestow a model with any meaningful predictive power. Instead of promoting the model to learn to predict both classes successfully, accuracy encourages a model to mimic the dataset's underlying class distribution. The model learns to simply predict the majority class for all of its predictions, thus giving a good score - but failing to have actually learned anything meaningful. This motivates the need for alternative scoring metrics that push a model to learn to successfully predict the minority class.

Specifically for the models in this investigation, the recall - the number of true positives divided by the sum of true positives and false negatives - and precision - the number of true positives divided by sum of true positives and false positives - of the minority class were used for training and scoring via a the balanced F-measure or F1 score - a evenly weighted combination of recall and precision. The F1 score is defined as the precision multiplied by the recall, divided by the sum of the two, multiplied by two. It is intended to equally weigh recall and precision in training, to assess not only how many items of the minority class are labelled - but also how many false positives are granted.

In addition, whenever possible, cost weights and prior probabilities were adjust to give more emphasis to the minority positive samples and less to the majority negative samples. Each class was given weight equal to the inverse proportion of the class within the whole dataset.

To help fight difficult learning stemming from the the dataset's curse of dimensionality - having a large number of features and a disproportionately smaller number of data points - a number of different feature selection techniques were employed to see if a smaller number of more predictive features could be identified and trained with to increase learning potential and model interpretability. Specifically, the top 1%, 5%, 10%, or 20% of all 477 features were selected for training via their ranking by one of two metrics, mutual info - ranking variables by their ability to obtain information about the label (and vice versa) individually - or an ANOVA F-value - ranking features by their ability to discriminate between the two classes on their own.

Each model was evaluated with an instance of *k*l*-folds cross validation for each metric paired with each selected percentage of features. As a baseline, each model was also evaluated with every feature present in training. Ideally one would like to see higher scoring models the fewer features that are selected.

To prevent information leakage between the training and test sets, feature selection occurred only on the training sets. Following this logic, since model evaluation occurred through cross validation - feature selection occurred separately within each outer fold of instead of only once before cross validation was started for each model. However due to feature selection occurring on k different selection of the dataset, the possibility for each fold to select different sets of features for evaluation exists. Ideally each fold selects the exact same set of features, but in practice this will not occur - especially when large subsets of features are considered. While some deviation in the selection of feature subsets is tolerable, if the differences between each feature subsets are too drastic - this should be taken as a sign that there is a fundamental problem with the dataset or feature selection method.

This motivates the need for a "feature stability" score to quantify the deviation in selected feature subsets between each fold, where higher scores signal a more less deviations between folds and a more "stable" dataset and feature selection method. For this investigation, a feature stability score f is defined as:

$$f = \frac{(k * d) - b}{(k - 1) * d}$$

Where k is the number of outer cross validation folds, d the number of features selected within each fold, and b the number of unique features selected across *all* folds. This is intended to be a very rough heuristic, simply signalling that the closer to 1 f is - the more similar the features selected by each fold are - with $f = 1$ being the scenario where each fold selected the same number set of features.

2.3.1 Model and Hyperparameter Specifications

For each instance of cross validated grid search occurring within each session of outer cross validation, the following hyperparameters for each model were selected as the pool of hyperparameters to be chosen from ... for the outer model of training.

For the logistical regression models, the maximum number of training iterations was set at 10000 - and the solving method selected was 'liblinear' due to the small number of datapoints, The regularization strength weights considered for tuning were those on the logarithmic scale from .01 to 10.

For naive Bayes classifiers, the only hyperparameter considered for tuning was whether or not to set the class weights prior to training.

For SVCs, the only kernel utilized was the radial basis function due to general effectiveness in multiple domains - with the maximum number of iterations for training set at 10 million. For hyperparameter tuning, values of gamma on a logarithmic scale from .001 to 10 were considered, as well as regularization parameters on a logarithmic scale from .01 to 10.

3 Experimental Results

Every type of model was evaluated with a variety of feature selection methods (a metric and some percentage of features) using nested cross validation - being trained and scored using the F1 score. However in addition the F1 score, the balanced accuracy of each model was also noted - along with the individual components of precision and recall. The success and interpretation of each model and its assigned feature selection method will be considered using these four metrics. Since each metric was tracked during each session of outer cross validation evaluation (*k=10* of training and testing on different splits of the data), the average of these outer fold-level values makes the value of the metric for the model and feature selection method overall. The exact values for each metric can be found in the tables following this section.

Looking at all the results as a whole, the general trends appear to be models and their feature selection methods having balanced accuracies in the range of 50-70%, F1 scores from 5-30%, recall scores from 40-85%, precision scores from around 6-17%, and features selection stability scores from 55-90%. In other words, it appears that overall that every model and its feature selection method had subpar accuracy in identifying both classes, but given the precision and recall scores - was able to successfully identify most positive (faulty) cases, but with a very high false positive rate. In addition, most evaluations ended with very similar feature subsets across folds. This would suggest that the dataset and feature selection metrics are stable and capable of yielding insightful results with a minority of classes.

Being more specific, it appears models assigned the ANOVA f-value metric for feature selection tended to have higher scores on all metrics than those using mutual information as a metric. This might suggest that simply discriminating between the two classes is more performent than trying to establish mutual information.

In addition, across all models and feature selection methods - having fewer features did usually result in higher metric scores, at least for percentages higher than 5%. A percentage of features smaller that appears to lead to diminished performance. This can be taken as definitive proof that only a minority of classes actually convey predictive power, while other only add noise.

It is also worth noting that it higher precision score usually appear to come with substantial degradation in recall performance. This possibly implies that it is much easier to identify a positive class than it is to be accurate about that prediction - with improvement in that latter coming at a large cost to the former.

At the level of individual models however, any naive Bayes classifier by far performed the worse on each metric. Logistical regression models and SVCs with mutual information as their feature selection metric appear to do noticeably better - but still pale in comparison to logistical regression models and SVC using ANOVA f-value. Between the latter of the two choices using ANOVA f-value however, any logistical regression model likely beats out SVCs due to their simplicity.

Taking in all these insights, the recommended model to any organization looking to implement suggest a system like this would be a logistical regression model using ANOVA f-value to select 1-5% of the best available features (assuming they are similar to the ones tested here). This would likely require the company to have a high tolerance for false positive, and a somewhat lower tolerance for still missing some faulty semi-conductors. This decision would likely hinge on whether or not it is cheaper for a company to wrongly stop the production of falsely identified working semi-conductors, or to keep processing broken semi-conductors. This decision would likely have to come from those with more domain knowledge.

An interesting takeaway from this investigation however is that, again as noted at the start of this publication, the powerful expressive capabilities of neural networks were still not able to be applied to this problem. However, the most successful model being of a simple logistical regression model likely gives credit to the idea that such a highly expressive model would only severely overfit to the data (as already suggested by the data, even with feature reduction, and tiny number of samples).

4 Limitations

It is worth noting that while this investigation appears to have given some insightful information to the problem and its associated data, there are still many limitations to what was carried out along with many other possible avenues of further inquiry. Specifically, only two feature selection methods were considered - there are a vast number of other techniques to be applied to this dataset such as principle component analysis, evolutionary algorithms, and more.

Another caveat important to note is that the conclusions from this analysis should still be taken with a grain of salt. The lack of additional data makes it difficult validation any claims from the evaluation set used here, making any claims given likely overly optimistic since all assessments occurred on the same data.

Table 1: Logistical Regression with Mutual Information

% of Features	1	5	10	20	100
F1	.1317	.1403	.1539	.1602	.1971
Precision	.0757	.0806	.09	.0967	.1263
Recall	.5155	.5455	.5455	.4855	.5482
Balanced Accuracy	.5394	.5509	.5741	.5777	.6285
Feature Selection Score	.5778	.7454	.8357	.8784	1

Table 2: Logistical Regression with ANOVA F-Value

% of Features	1	5	10	20	100
F1	.2624	.2272	.2141	.2089	.1942
Precision	.1673	.136	.1322	.1284	.1348
Recall	.6145	.7036	.5873	.6	.4027
Balanced Accuracy	.6976	.6895	.6538	.6517	.6036
Feature Selection Score	.8667	.9028	.8379	.9193	1

Table 3: Naive Bayes with Mutual Information

% of Features	1	5	10	20	100
F1	.0516	.1341	.1378	.1306	.1258
Precision	.0839	.0813	.0761	.0707	.0678
Recall	.0881	.6464	.8036	.8646	.8764
Balanced Accuracy	.4999	.5482	.5546	.5283	.5097
Feature Selection Score	.5556	.787	.8379	.8784	1

Table 4: Naive Bayes with ANOVA F-Value

% of Features	1	5	10	20	100
F1	.1042	.1171	.1426	.1438	.1209
Precision	.1283	.1223	.1436	.1459	.0651
Recall	.0964	.1136	.1427	.1446	.8373
Balanced Accuracy	.5215	.5304	.5451	.5436	0.4925
Feature Selection Score	.9111	.9075	.9005	.9158	1

Table 5: SVC with Mutual Information

% of Features	1	5	10	20	100
F1	.1438	.1156	.1754	.1587	.1025
Precision	.0863	.0714	.1055	.1009	.0605
Recall	.5018	.52	.7264	.38	.57
Balanced Accuracy	.5525	.5224	.5848	.5921	.5189
Feature Selection Score	.7111	.7685	.8379	.8842	1

Table 6: SVC with ANOVA F-Value

% of Features	1	5	10	20	100
F1	.2654	.2419	.2102	.1945	.1139
Precision	.1756	.1533	.1324	.1267	.0708
Recall	.5563	.5873	.5182	.4327	.5182
Balanced Accuracy	.6835	.6743	.6381	.6124	0.5355
Feature Selection Score	.8889	.875	.9074	.9099	1

References

[1] Dua, D. and Graff, C. (2019) UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Pedregosa et al. (2019) Scikit-learn: Machine Learning in Python [https://scikit-learn.org/stable/index.html]