Semiconductor Manufacturing Process Test Result Classification

Mingrui Chen

Data Science Initiative, Brown University

GitHub

1. Introduction

1.1 Semiconductor Manufacturing & Machine Learning

With the rapid development of the semiconductor manufacturing and along with the growth of material science, semiconductor production has become far more complicated, could involve hundreds of wafer fabrication steps (Liu et al., 2022). While equipment sensors provide massive amounts of data, like pressure, flow rate, temperature, etc. (Fan et al., 2020), the latent relations are intricated. Manually finding the key procedures that lead to the failure could be time-consuming and expensive, hence techniques like Machine Learning are applied on sensor data collected for fault detection and classification (FDC) purpose (Chen et al., 2017), to achieve quality control and lower production cost.

1.2 Data Set & Project Objective

The dataset is from the UCI ML repository SECOM (McCann & Johnston, 2008), which records 591 features (each represent measured signal data), and 1567 instances (each stands for a single production entity). The target variable is label '-1/1', which represents the 'Pass/Fail' yield in the house line testing, and there are 104 fails in total of 1567 examples which means that the dataset is imbalanced.

However, to noticed that it's not provided by source that which actual processing procedure the sensors feature belonged to, so we assume it as general procedures across all steps without special treatment. Features are all numerical, and the missing values fraction depends. The date stamp only stands for the record time and not relate to the production time, hence i.i.d dataset.

My objective for this project is to classify the binary test result based on signal data and find key factors that contribute to yield excursion downstream in the manufacturing process if possible.

Properties	**Description**
Shape	(1567, 592)
Target (y) variable	'-1/1' labels which represent 'Pass/Fail'
Predictor (X) variables	Numerical sensor signal data
Missing values	Yes
i.i.d dataset	Yes
Imbalanced dataset	Yes
Univariate column	116 columns, which only contain constant values for each row.

Table 1: Dataset properties and descriptions

2. EDA

The purpose of this step is to explore the data distributions and key attributes, therefore find useful information to help determine data preprocessing procedures and following split & pipeline strategy.

2.1 X features distributions

In this dataset, 116 features are univariate with constant value, examples are like feature 4, 5 shown in the figure 1. These columns are dropped in the data preprocessing step, since the removal of constant features won't influence the model performance.

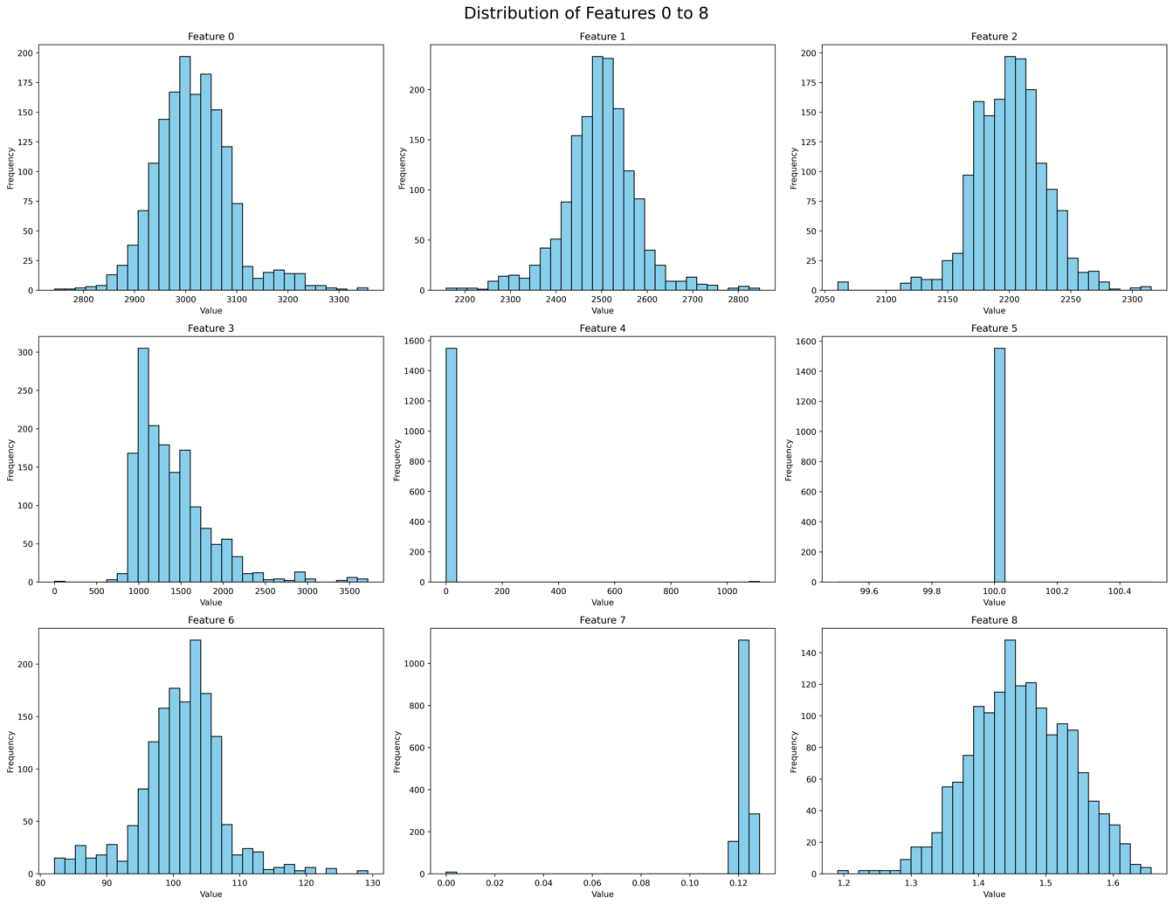

Figure 1: Feature 0-8 distribution

From the data source, missing data in the x features are assumed to be no signal, so I replace the missing values with 0 when using *Logistic Regression, SVC, Random Forest Classifier*, and kept the null values for *XGBoost* built-in module to handle. 91% of the features have the missing fraction smaller than 3%.

2.2 Target variable distribution & Correlation

In the correlation heatmap of the features, we can see that some of them have strong correlation with other features. To avoid Multicollinearity and prevent redundant information (noises), also as before training feature selection, features with absolute correlation stronger than 0.9 are dropped, leading to a removal of 220 columns.

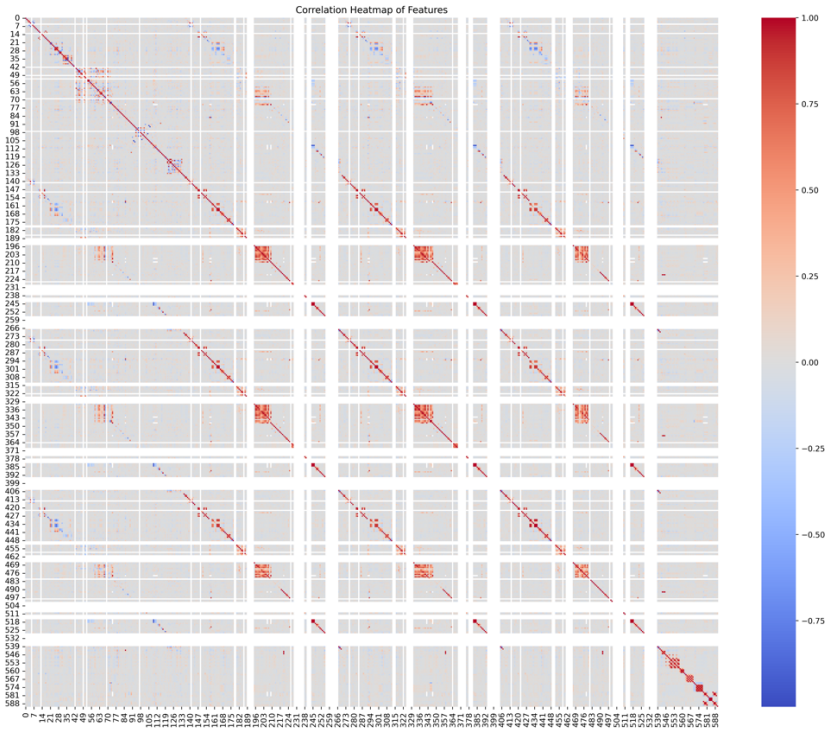

Figure 2: Correlation Heatmap of Features

Across all the features, the feature 59 has the highest absolute correlation with target y variable equals to 0.16. This indicates the weak correlation between features and the target y, hence no need to drop features in this step.

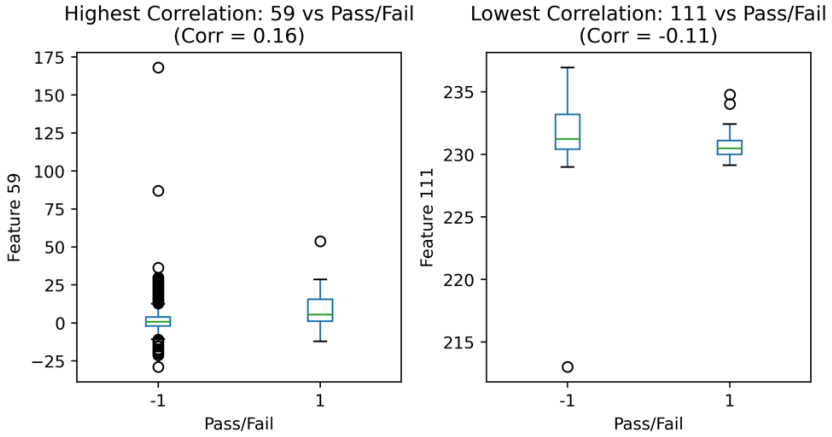

Figure 3: Highest / Lowest Feature Correlation with Target y

3. Methods

3.1 Split / Cross Validation Strategy

Stratified train/test split with a ratio of 80/20 is applied, since the dataset is imbalnced and only have limited instances (1567). After that, *StratifiedKFold* with k = 3 folds is used inside the *GridSearchCV* Pipeline. Considering that there are only 104 fails in total dataset, having too many folds may lead to the minority group under represented and influence the model performance, k euqlas to 3 is chossen.

3.2 Preprocessor

Data preprocess include 4 steps:

1. Drop univariate features (with constant values)
2. Drop features with |corrrelation| > 0.9
3. Missing value imputation (different strategy applied here)
4. Apply *StandardScalar*

All features are numerical and continuous, so I use sklearn's *StandardScalar* to standardize each feature. Missing data imputation strategies are applied as discussed in 2.1 X features distribution.

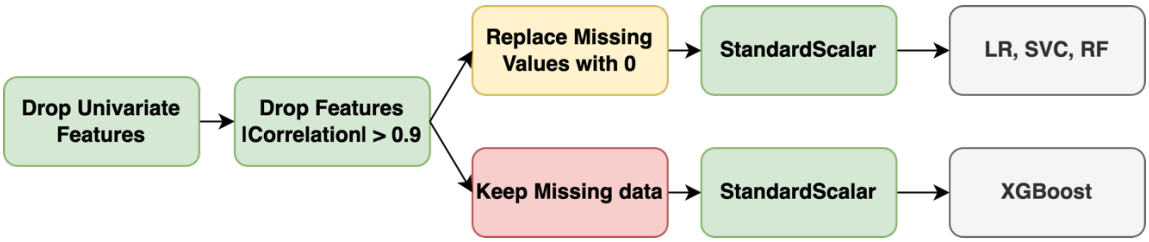

Figure 4: Data Preprocessing Steps

3.3 Evaluation Metric

The chosen evaluation metric is *F1.5.*

F_beta score is selected firstly due to the imbalanced dataset, and I chose *F1.5* which towards more on the recall because it matches the purpose of failure detection, even at the cost of increasing false positives (pass results being classified as fail).

3.4 Hyperparameter Tuning & ML Pipeline

Logistic Regression, Support vector classifier, random forest classifier and XGBoost classifier are 4 selected ML algos given the binary classification problem. To optimize the model performance, hyperparameters tunning is applied by systematically selecting the best combination of model parameters.

For the logistic regression, *'saga'* solver is selected because it's the only solver that support *elastic net* penalty, and '*l1*', '*l2*' regularization penalty is applied using *solver = 'liblinear'*.

	Logistic Regression	Support Vector Classifier	Random Forest Classifier	XGBoost Classifier
Parameter Tuned	*C* - log scale	*C* - log scale	*Max depth*	*Max depth*
	['liblinear'], ['saga']	*Gamma* - log scale	*Max Features*	*subsample*
	['l1', 'l2', 'elastic net']	/	/	*colsample_bytree*
	/	/	/	*Learning rate*
Best Parameters	*C = 0.01, l1_ratio = 0.7, penalty = 'elasticnet', solver = 'saga'*	*C = 1, Gamma = 0.001*	*Max_depth = 2, Max_features = 0.75*	*Max_depth = 1, Subsample = 0.9, Colsample_bytress = 0.7, Learning rate = 0.1*

Table 2: Hyperparameter Tuning

Feature selection portions are tunned in the RF and XGBoost using *'max_features'* and *'colsample_bytrees'*. Also, to handle the imbalanced dataset, *class_weight='balanced'* is used for LR, SVC and RF, and *'scale_pos_weight'* is used for XGBoost.

3.5 ML Pipelines

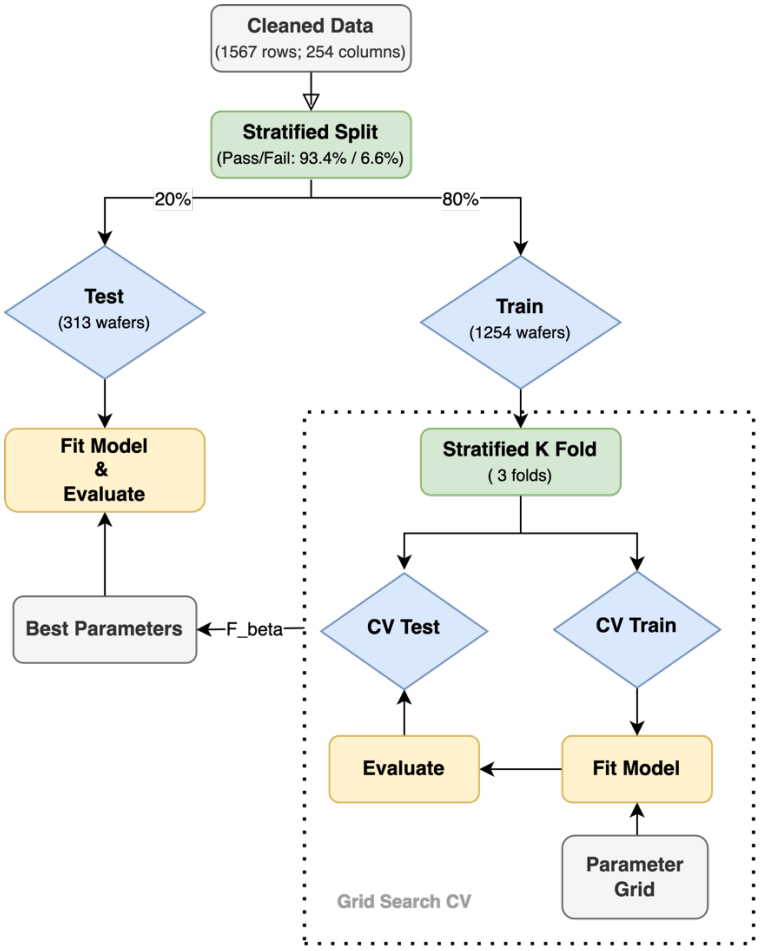

Figure 5: ML Pipeline & GridSearchCV

To measure the uncertainty and randomness, the random states within *range (5)* is used. For each random state, cleaned data is stratified splitted using sklearn's *train_test_split (..., stratify = y)* into 80/20 train/test sets, apply *StandardScalar ()* for both train/test set, and then use *StratifiedKFold (k=3)* to cross validation.

GridSearchCV is used to loop through parameter combos, evaluate on the validation set, and then fit the best parameters on the test set, for each ML algos under each random state. F_1.5 is optimized while using GridSearchCV.

4. Results

Using *F1.5* as evaluation metric, the *baseline_score* is predicting all instances as the minority group can calculate the F score on the whole dataset, giving a baseline *F1.5 = 0.1877*.

4.1 Model Performances

The step in the figure is repeated 5 times under *random states = range (5)*, and the results are summarized by mean and standard deviation of each model's F1.5 score across 5 random states, to measure the uncertainty and balance the performance with randomness.

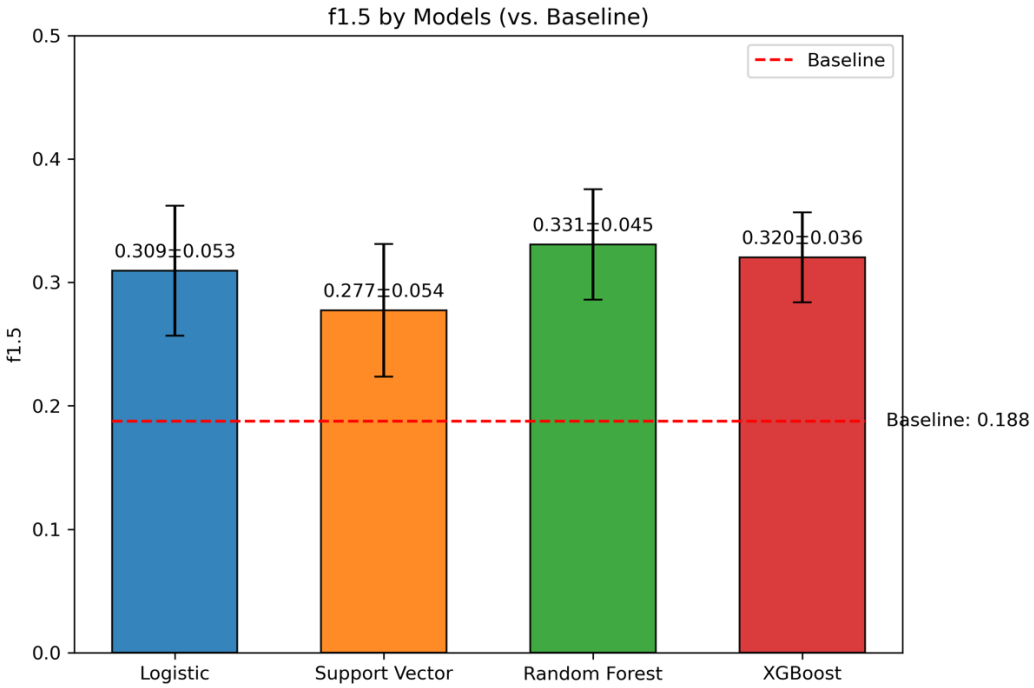

Figure 6: Comparison of F1.5 scores of each Algorithms with the baseline score.

From figure 6, all the models perform above the baseline score but not showing a dramatic improvement. *Random Forest Classifier* shows the best performance under given random states, while *Logistic Regression and XGBoost* having a slightly lower f1.5 score but still higher than the *Support Vector Classifier*, which is the worst satisfying model.

However, aside of the mean score, *XGBoost* has the lowest std value which implies the stable performance under randomness. The high std value for the *LR*, *SVC* models and the comparatively lower std for *RF, XGBoost* implies the non-linear relationships, corresponding to the complex nature of semiconductor manufacturing process.

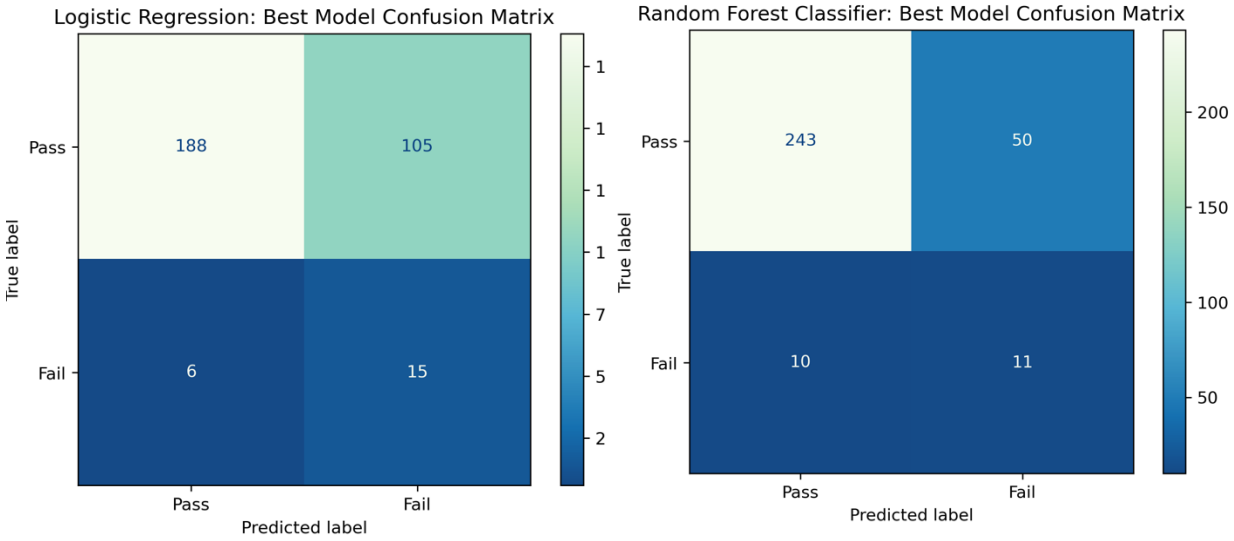

Figure 7 & 8: Confusion Matrix for Best Model using Logistic Regression, XGBoost Classifier

From figure 7 & 8, the left side is LR model using '*elastic net'* as penalty with *solver = 'saga'*, and the right side is RF model with '*max_depth = 2'* and *'max_features = 0.75'*.

As we can see from the plot, LR having more true fail result correctly predicted, at the cost of predicting almost 1/3 of the true pass results as fail. The RF model have a more balanced performance, having less FP but also more FN.

4.2 Global Feature Importance

To understand the feature contribution to the model predictability, 3 different importances metrics are use. Permutation feature importance, and SHAP values are calculated for the best RF model, and XGBoost total gain for the best XGBoost model.

1. **Permutation Importance:** It's calculated by randomly shuffling the values of a feature and measuring the resulting decrease in the model performance, i.e. *F1.5* score here.
2. **SHAP Global Feature Importance:** It's calculated by aggregating the absolute SHAP values of a feature across all the samples, hence reflect the average contribution of that feature to the predictions.
3. **XGBoost Total Gain:** It's calculated based on how much the feature improves the loss function during the tree construction, focus more on the model training but less directly interpretable compare with SHAP and Permutation Importance.

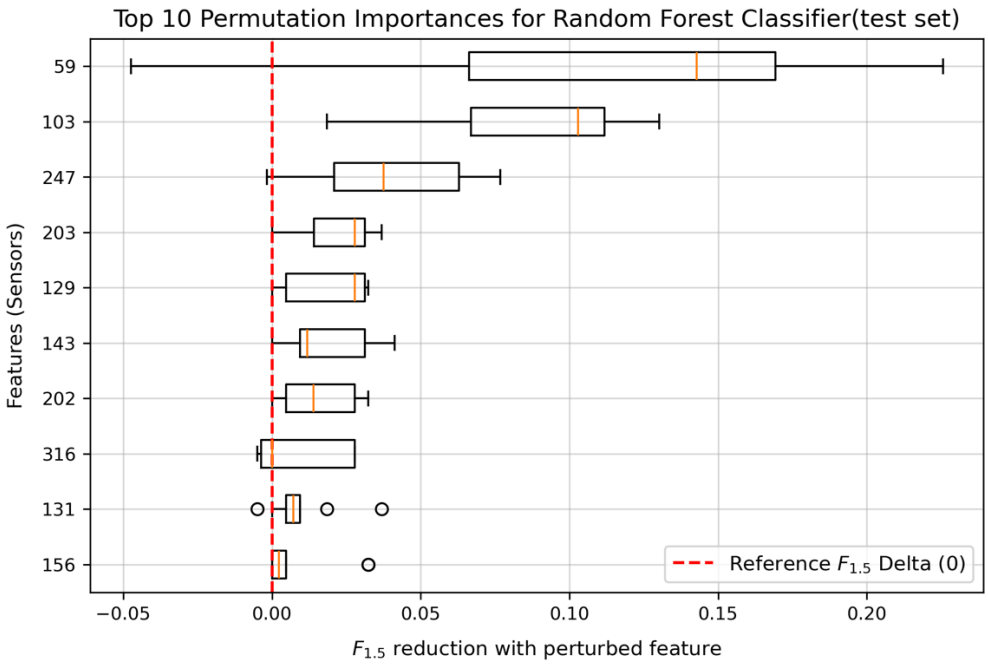

Figure 9: Top 10 Permutation Importances - Random Forest Classifier

From figure 9 and figure 10, we find that '*Feature 59'* has the highest contribution to the model performance in both permutation importance and SHAP global importance. This matches with the EDA check where *'Feature 59'* has the highest correlation with the target variable y. Other shared top 10 features include *Feature 103, 247, 143*, which appear both in figure 9 and 10.

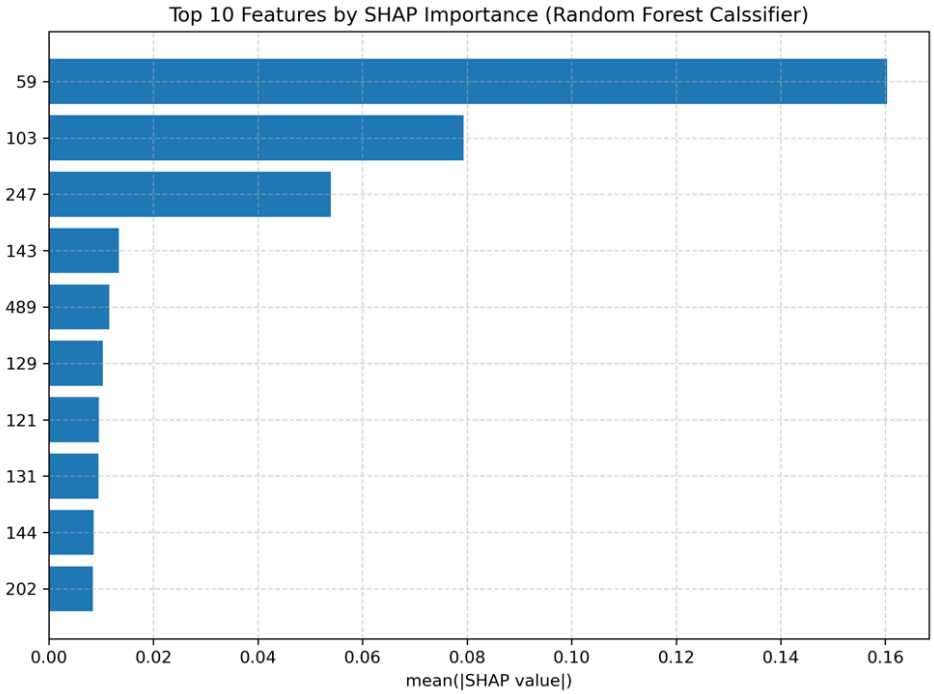

Figure 10: Top 10 SHAP Importances - Random Forest Classifier

For the figure 11, I find some other features that didn't show up in the previous two methods. The reason includes using different models (XGBoost vs. Random Forest), and the different calculation.

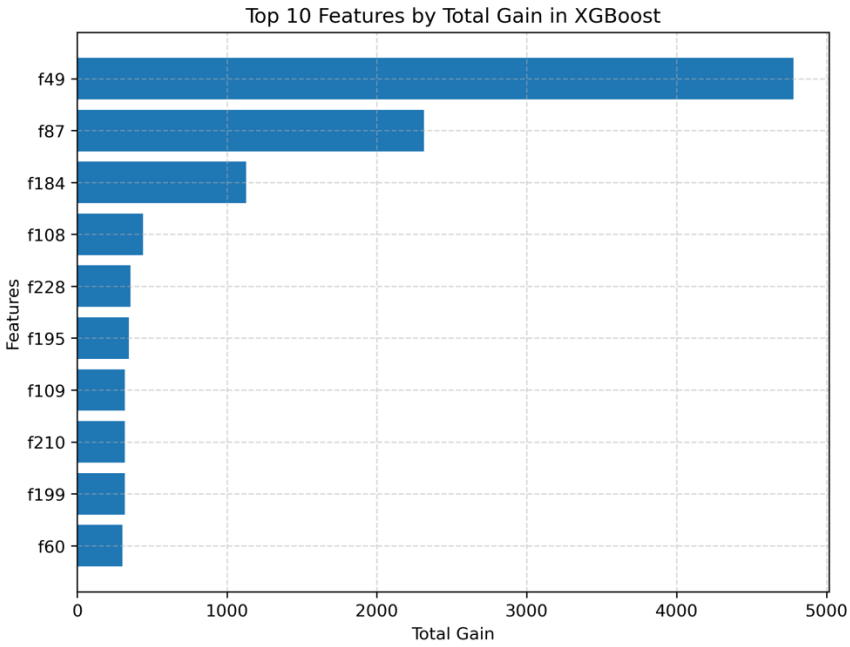

Figure 11: Top 10 Features by XGBoost Total Gain

In the real-world context, the physical equipment sensor corresponds to each feature are the key factors that we would like engineer experts to examine for semiconductor manufacturing failure detection.

For instance, feature 59 may be the leading factor that influence the failure result, and engineers can not only examine the sensor but also other related sensor in the production chain for feature 59 to achieve quality control and monitor.

4.3 Local Feature Importance

Whereas local feature importance helps us understanding the feature contribution to the whole process, local feature importance serves important role in terms of understanding individual instance prediction.

For example, the figure 12 indicates that for the point 263, certain sensor features (e.g., 205 and 203) pushing the prediction upward with positive influence, while features 247, 103, 59, 202 and 423 on the blue mark pulling downward the prediction outcome. For point 298, feature 103 and 59here plays a different role and push the prediction upward with red mark.

Hence, we could use local feature importance to examine specific failure/pass result and find the key sensors related in real life.

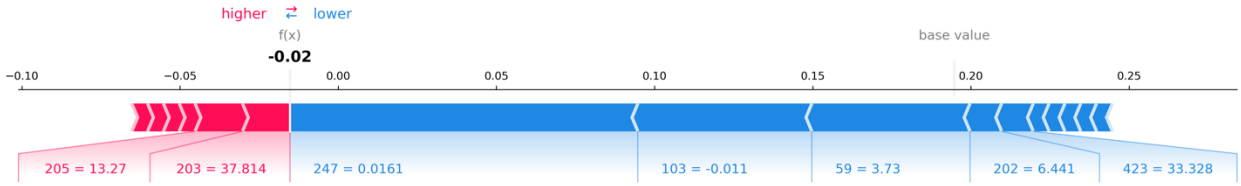

Figure 12: Local feature importance shap value for point 263

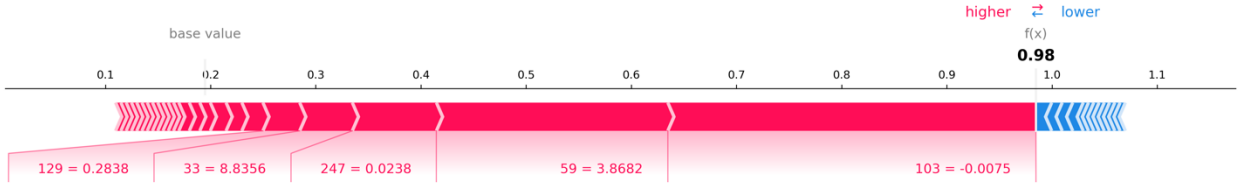

Figure 13: Local feature importance shap value for point 298

5. Outlook

The current best model is Random Forest Classifier, with Logistic Regression and XGBoost classifier has the slightly slower but close performance. Though all the models outperformed the baseline f1.5 score, the results are still not that satisfying as shown in the confusion matrix, where many true passes are classified as fails when we want to maximize failure detection.

For the model performance, I would like to try different other model to increase the predive power, such as K-nearest neighbors (kNNs) and naïve Bayes which other papers has tried on different dataset (Fan et al., 2020), and maybe deep learning methods to better learn from the non-linear relationship.

For the interpretability, it will be great if I can have a chance to advise domain experts and gain more knowledge on the Semiconductor Sensor data. Asking source donator which specific process these sensor data belong to will be another helpful information for understanding the prediction.

6. References

Chen, Y.-J., Wang, B.-C., Wu, J.-Z., Wu, Y.-C., & Chien, C.-F. (2017). Big data analytic for multivariate fault detection and classification in semiconductor manufacturing. *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, 731–736. https://doi.org/10.1109/COASE.2017.8256190

Fan, S.-K. S., Hsu, C.-Y., Tsai, D.-M., He, F., & Cheng, C.-C. (2020). Data-driven approach for fault detection and diagnostic in semiconductor manufacturing. *IEEE Transactions on Automation Science and Engineering, 17*(4), 1925–1936. https://doi.org/10.1109/TASE.2020.2983061

Liu, D.-Y., Xu, L.-M., Lin, X.-M., Wei, X., Yu, W.-J., Wang, Y., & Wei, Z.-M. (2022). Machine learning for semiconductors. *Chip, 1*(4), 100033. https://doi.org/10.1016/j.chip.2022.100033

McCann, M., & Johnston, A. (2008). SECOM [Dataset]. *UCI Machine Learning Repository*. https://doi.org/10.24432/C54305