Predict time taken to detect defects in E-Beam Wafer Inspection Tool

Shweta Patidar

shweta24@stanford.edu

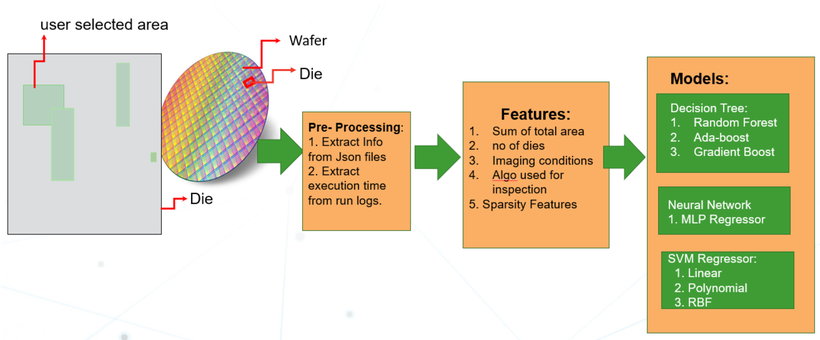
Stanford University

Problem Statement

Wafer inspection tools are used to detect defects on wafer in chip manufacturing process. User gives list of rectangles in a silicon wafer; Inspection tool detects defects in given region. It involves several steps like taking the images of user interested region, and then processing the images using deep- learning based techniques. Execution time of entire process depends on several factors:

1. how many rectangles user has given to inspect,
2. how much total area they cover in a wafer,
3. Imaging conditions to take the images like pixel size, pixel density etc.
4. Imaging processing strategies, some strategy takes less time as compare to others.
5. Hardware conditions-inspection involves several hardware like stage, optics, and their condition can vary based on time and frequency of maintenance.

our task is given all the above parameters predict the execution time of entire process.



Data

Data is collected from 2 different tools having different hardware conditions. We have json files for different runs. Collected all information from json files by desterilizing them, and execution time is collected from run logs.
17058 samples are collected.

References

1. Lawrence, Ramon. "Using neural networks to forecast stock market prices." *University of Manitoba* 333 (1997).
2. Khaidem, Luckyson, Snehanshu Saha, and Sudeepa Roy Dey. "Predicting the direction of stock market prices using random forest." *arXiv preprint arXiv:1605.00003* (2016).

Video link:

https://drive.google.com/file/d/1IhjOecFNuRF2LlCVnUFPlphW0Ck8ZFM7/view

Features Extraction

We experimented on three different types of Features:

1. Sum of area of all rectangles, frame average, pixel density, image size, die group size, no of dies, Is Playback Run, Is RTC Enabled, Sampling percentage, tool used (different sizes of rectangles and how many such rectangles are present of the same size), these way we have 143 features these are too large.

2. Sum of area of all rectangles, frame average, pixel density, image size, die group size, no of dies, Is Playback Run, Is RTC Enabled, Sampling percentage, die Fields count, die sites count, die Centre distance, die field distances, tool used, now we have only 14 features, it will help model to run efficiently.

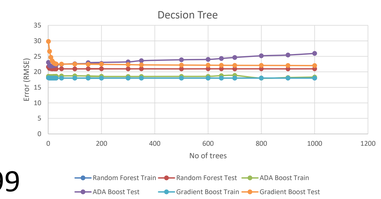
Here we computed some additional features to estimate the sparsity of rectangles, for example die site count will represent no of images to be grabbed, and die field count will represent no of stage moves required based on given set of inputs.

3. Features included in 2nd and some additional features representing sizes of rectangles and how many such rectangles are present of that size

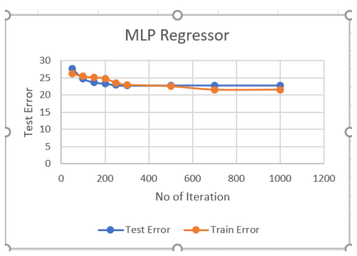
Models Comparison

We used features – (2) to perform model comparison.

1. Decision Tree Comparison:
We used Random Forest,
Ada boost, Gradient boost,
as we can see Random Forest gets the lowest error rate- 20.99 (RMSE).
on test set.



2. MLP: we applied multi layer perceptron regressor with activation function as “relu” and solver as adam, lowest error rate with 700 or more iterations – 22.83 (RMSE)



3. SVM Regressor – We used Linear, Polynomial and RBF kernel.

	Linear(RMSE)	Poly(RMSE)	RBF(RMSE)
Train Error	30.65	22.54	22.37
Test Error	32.04	24.54	23.78

RMSE – root mean square error

Results and Discussion

	Features 1	Features 2	Features 3
Random Forest Test	24	21	18
Random Forest Train	23	18	17.69
MLP Test	27	22.8	24
MLP Train	26	22	23

Error is RMSE(root mean square error)

1. We tested on three different types of features, and feature- 3 achieves better performance in case of Random forest, but with MLP feature – 2 is performing slightly better, may be because of data-set, neural network needs some more data to train.
2. We used data from different tools, and every tool has different hardware specifications. When we tested using the single tool data, we got higher performance, so our model is not able to differentiate between different tool data.

	Features 1	Features 2	Features 3
Random Forest Test	18	18	16.18
Random Forest Train	17.4	17.38	15.61
MLP Test	20	17.8	16.74
MLP Train	19.6	16.07	15.52

3. We can see from above table that for single tool data Random Forest is performing better than MLP.
4. Based on prediction we have observed that all models are not able to accurately predict the execution time where execution time is more than one hour, reason is we have lots of training samples for less execution time, and very less are for larger execution time.

Conclusion and future Work

1. Adding more samples having bigger value of execution time can help.
2. Adding some more information in feature related to tool configuration, so that tool model can differentiate samples from different tools.

Acknowledgements

Thanks to Professor Percy Liang, Professor Dorsa Sadigh, TA Sharman W Tan for their guidance.