Big Data Analytics for Semiconductor Manufacturing

Team NULL

鄒侑霖
NTU CSIE
b00902022@ntu.edu.tw

李廣和
NTU CSIE
b00902055@ntu.edu.tw

宋昊恩
NTU CSIE
b00902064@ntu.edu.tw

ABSTRACT

Semiconductor is the foundation of modern embedded systems. In this high computation power era, the circuit probing value of semiconductor production is widely concerned. In this series of competition held by TSMC, we derive a chain of preprocessing, model learning, and model prediction. Furthermore, we have a deep observation into the importance of each feature and give detailed explanation and discussion in this report.

Keywords

Semiconductor manufacturing, machine learning, data mining

1. Introductions

In previous competition, we utilized only categorical features regarding "carrier", "tool", "fab", "recipe", "chamber" for CP prediction, and gained minimized mean-square-error (MSE) of *4.49* in terms of validation with the GBM model.

In current competition, more than *100,000* additional FDC data are given and the biggest challenge is to process such great amount of data. To solve this problem, we create novel and realistic framework for precisely picking the key features determining the values of CP in terms of regression performance. This sharply juxtaposes dimension reduction methods as it is based on the principal element decomposition disregarding the regression performance. In addition, traditional dimension reduction is notorious for its high space and time complexity, whereas our method can be done in the memory and computation limitation given for this competition. As a conclusion, after processing more than *100,000* features, we found *83939* key features determining a precise CP value, and got an overwhelming mean square error of *2.01*

The rest of the report is organized as follows: In section 2, we will define our main problem we have solved in this competition. Section 3 provides our proposed framework in details. An in-depth framework and modeling procedure are given. Afterwards, we elaborate our experiment design in section 4. In section 5, we show complete experiment results and give discussion in section 6. Section 7 is the conclusion of this report.

2. Problem Definition

In the 2nd stage, lots of FDC data has been given in order to reach a better result. However, with explosive amount of data, the most demanding challenge for us is to find a way to deal with the high dimensional data problem. The tremendous amount of data stops us from applying general statistical and data mining methods. Moreover, a great amount of features may be useless for determining precise CP value. Not until we pick up the right part of important features can we use these information. So, in order to make our target clear, we state the problem definition here:

Given semiconductor production features, we aim to:

1. *Create novel & realistic high dimensional data mining framework*
2. *Precisely predict the CP and the key factor resulting in such outcome.*

3. Methodology

3.1 Why to use machine learning

In this Big Data era, machine learning plays an important role. Suppose there is a perfect function f that satisfy $f(x_i) = y_i$ for each raw data x_i and its label y_i, machine learning focus on finding a hypothesis set g which approximate the target function f by minimizing a loss function (like MSE in this competition).

Traditional statistical methods try to make inference about f by analyzing data, and their results are guaranteed with math assumption. However, it focuses on the correctness, not the computation of the process. Therefore, when facing large data, traditional statistical methods may not be applied easily due to its high computation complexity or miscellaneous progress.

In contrast, machine learning focuses on the computation of target value basing on the proofs of statistics. By substituting complicated process with mathematical problems, machine learning is a much easier and faster way to find a good result. Furthermore, thanks to the explosive progress on the computation ability of computers, many models can be implemented and more and more problems can be solved. By analyzing performance of different features and models, we can also find out some special characteristics in data which is related to our target.

Due to its advantage, we decide to apply machine learning models which help us find out secrets hidden in the data:

i. Support Vector Regression(SVR)

Support Vector Machine (SVM) is a supervised learning model which is widely used in solving Machine Learning problems. The main concept of SVM is finding the largest separation plane between positive and negative samples as Figure 1. Further research was extended into regression problem like SVR by mimicking the behavior of SVM and optimizing the absolute error outside the separation plane.

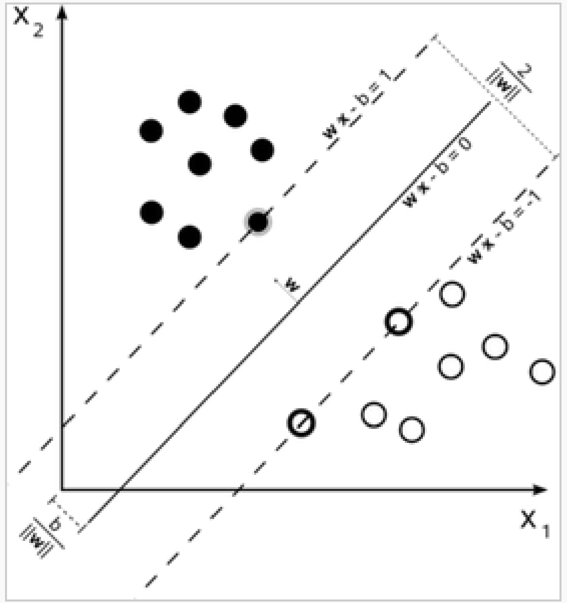


Figure 1: SVM tries to find out the maximum boundary. (en.wikipedia.org)

The optimization function of SVR is given as follows (primal form):

$$\text{minimize} \frac{1}{2}||w||_2$$

$$\text{subject to} \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases}$$

ii. Kernel SVR

Although SVR has integrated the meaning of largest separation plane, the model complexity is still restricted as it can only captures linear behavior from the single weighting vector w. In order to build a more powerful model, researchers came up with the usage of kernel function to approximate projecting the feature space into higher dimensional space. For example, using a 2nd order polynomial kernel is approximately equal to solving the SVR in the 2nd order polynomial expansion of the feature set in the same problem. Moreover, by using a Gaussian kernel function, we can even achieve projecting the feature space into infinite dimensional space. We can see the illustration as Figure 2 to gain further understanding.

The optimization function of kernel SVR is given as follows (dual form):

$$\min \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M}(\alpha_n^{\vee}-\alpha_n^{\wedge})(\alpha_n^{\vee}-\alpha_n^{\wedge})k_{n,m}$$

$$+\sum_{n=1}^{N}((\varepsilon+y_n)*\alpha_n^{\wedge}+(\varepsilon-y_n)*\alpha_n^{\vee})$$

$$\text{such that }\sum_{n=1}^{N}1*(\alpha_n^{\vee}-\alpha_n^{\wedge})=0$$

$$0\le\alpha_n^{\wedge}\le C, 0\le\alpha_n^{\vee}\le C$$

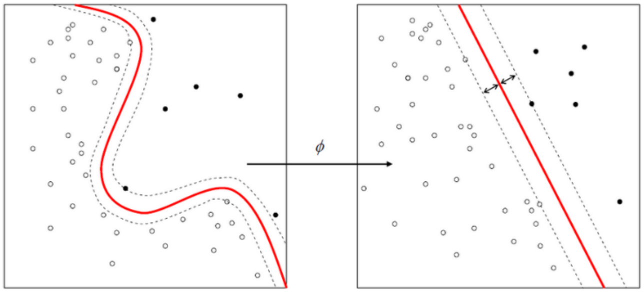


Figure 2: Use transformation matrix ϕ to transform instances space. (www.npmjs.com)

iii. Gradient Boosting Machine(GBM)

GBM is a tree-based model, whose main idea is to solve an optimization problem for new weak learners that minimizes the aggregated prediction. GBM is a model which is highly used in competition and usually performs well. Nevertheless, it is a powerful model in fitting data, which makes parameter tuning and validation set important. In this competition, we focus on tuning two parameters: trees number and shrinkage, both of them are related to the speed of convergence in GBM. With the help of validation set, we can tune these parameters to decide how deep should the model goes, and efficiently eliminate the problem of overfitting. The optimization function of original gradient boosting is given as follows:

$$F_m(x)=F_{m-1}(x)-\gamma_m\sum_{i=1}^{n}\nabla_f L(y_i,F_{m-1}(x_i)),$$

$$\gamma_m=\arg\min_{\gamma}\sum_{i=1}^{n}L\left(y_i,F_{m-1}(x_i)-\gamma\frac{\partial L(y_i,F_{m-1}(x_i))}{\partial f(x_i)}\right)$$

Besides, the package we use also support interaction between features, while 1 indicates additive model and 2 means two-way interaction. The effect of the interaction somehow like the kernel-trick in SVM, which transforms each sample to a higher dimension space. However, the higher the interaction is, it not only takes longer for a GBM model to train, but also has a higher risk in overfitting training data. Here, we mostly set it to 3, where yields good performance and efficiency.

iv. RF

RF (Random Forest) is another well-known tree-based model for supervised learning problem. It is called a “forest” because RF is an aggregation model, which consists of many individual, single decision trees. Instances will be sampled by random to build each decision tree to increase the diversity. This strategy is called Bootstrapping. Since RF uses the average prediction results from decision trees, it reduces the possibility of overfitting training data, and gives stable performance in most of real world problems.

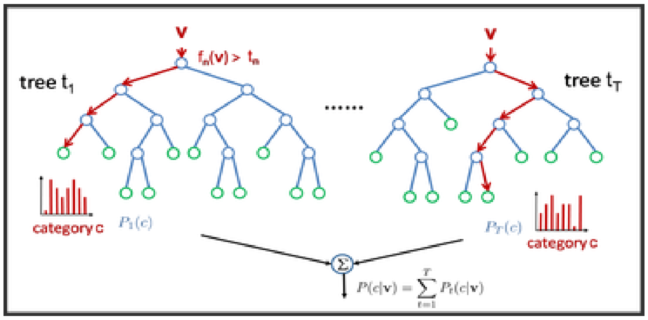


Figure 3: Random Forest aggregates Decision Trees (http://www.iis.ee.ic.ac.uk/icvl/iccv09_tutorial.html)

3.2 Feature Selection

After we have abundant of models, can we easily apply it to the target task? Unfortunately, due to the explosively large number of features, we cannot even load the entire training data into memory. In order to deal with high dimensional data. Two well-known statistical method is available. The first one is called dimension reduction and the second one is called feature selection. Obviously, even if neglecting the large time complexity of dimension reduction method, we still cannot load the data into memory. Therefore, we apply a two stage feature selection framework in order to apply machine learning methods. In another words, we want to find the most important set of feature for determining the value of CP:

$$\underset{feature\ set}{\operatorname{argmax}}\ regressor\ performance(feature\ set)$$

Notice that in order to select features yielding the best performance, we must have a measurement and a held-out dataset (i.e. validation set) to validate our performance. The details of these will be given in the following sections. Now we can just assume that we have a measurement and a trustworthy validation set.

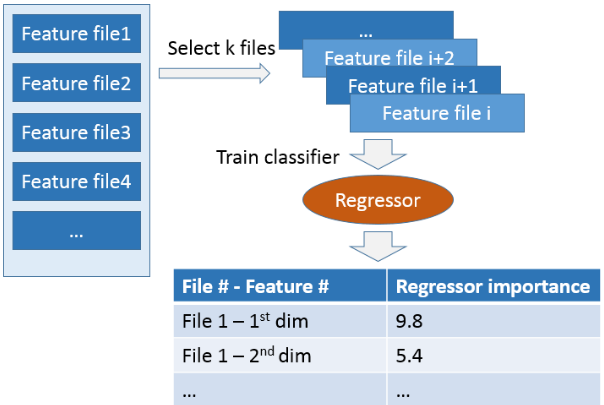
i. *Bunch of File Selection*

The very first problem of using computational power to solve the task is that we need to load the features into the main memory, while the memory usage is only limited to 11G, smaller than the total dataset size in this competition. That is, at any given time, we can only load part of the dataset into memory. As a result, we propose *Bunch of File Selection* as the following illustration as Figure 4.

The key idea of the Bunch of File Selection method is that, we maintain a table recording the mapping: [feature # : importance]; then in each iteration, we randomly select k feature files, feed them to a regressor, then use the validation performance and regressor feature-importance to calculate the feature importance as following:

$$Feature\_Weight[file_A, i_{th}\ feautre] = \frac{Regressor\_Feature\_Wegiht[file_A, i_{th}\ feautre]}{Validation\_Performance(this\ feature\ set)}$$

Since some of machine learning models can support the usage of feature importance, such as the weighting vector in linear regression and selected dimension in tree-based models, we can use those as a proxy of regressor feature importance. In addition, after obtaining such feature importance in each iteration, we should also divide each value of importance by the validation performance during this iteration. It is because if two features a, b have identical feature importance value, but the performance of their belonging feature sets are different. At this time, we should not give two features the same weighting in the feature importance.



File # - Feature #	Regressor importance
File 1 – 1st dim	9.8
File 1 – 2nd dim	5.4
...	...

Figure 4: Bunch of File Selection

ii. Compact Genetic Algorithm

Since the *Bunch of File Selection* method has selected part of features, now we can assume that all features can be loaded into memory after selection. Now, we can start to use some optimization algorithm for selecting key features. One of well-known search algorithm in Artificial Intelligence is Genetic Algorithm, but a shortage of using old-style simple genetic algorithm is that we should maintain a big matrix, called *population*, consuming our memory. So instead of simple genetic algorithm,

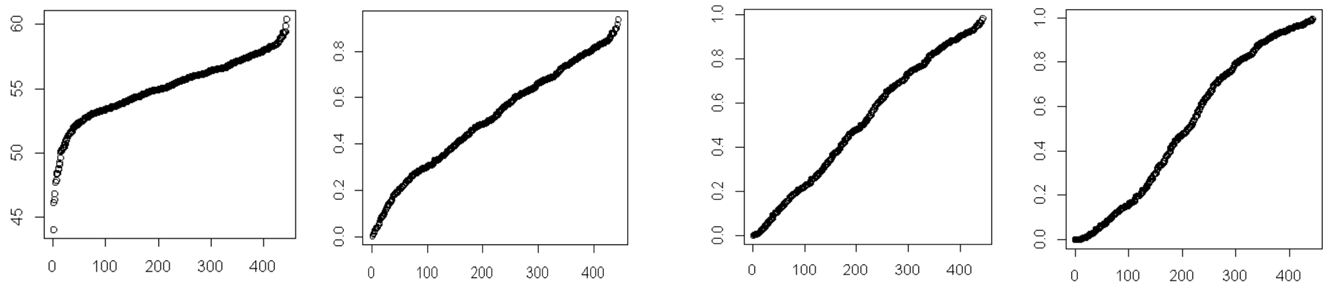


Figure 6: Label distribution. The names start from left to right are: original distribution, distribution after sigmoid (c = 0.5), distribution after sigmoid (c = 0.75) and distribution after sigmoid (c = 1).

we adopt compact genetic algorithm(cGA), which is capable of finding optimal given large enough population size while only maintaining probability values for each dimension. The illustration of cGA is depicted as Figure 5.

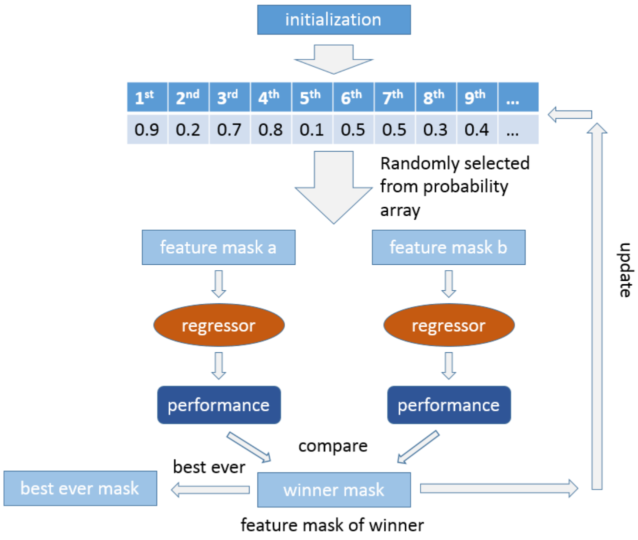


Figure 5: compact genetic algorithm

As the figure, we first initial a probability vector, whose elements are all valued 0.5. And in each iteration, we pick some feature from the probability vector. For example, if the length of probability vector is 5 and the [1, 0, 0, 1, 0] is the outcome of selection. Then the resulting feature mask is [1, 0, 0, 1, 0] which means we only use the 1st and 4th feature as the feature set for regressor. Finally, we choose the better feature mask based on validation performance to update probability vector.

Another advantage of this method is that it is an anytime algorithm, so as long as a best-ever feature mask occurs, that is, the subset of feature yielding the best performance, we can directly output the mask and observe key features. As a conclusion of this subsection, with the above feature selection methods, we not only solve the memory limitation problem but also reduce redundant features. Moreover, based on the validation performance, we also obtain a decisive feature set yielding the best prediction performance. The feature set can also be extended for analysis for operation administrators as the meaning of feature set is the key to determine the value of CP.

3.3 Label Transformation

Beside analysis on training data, we also pay attention on CP, the label data. We plot the value of CP after sorted and get Figure 6. As you can see, these values form a line which is steep at two ends and clam in the middle, indicating most of CP are closed to the mean value. After seeing this graph, an idea comes to our mind: why not do a transformation on label and make the slope smoother? A smooth slope means labels uniformly scatter in an interval and we think it might leads to a better performance if we predict on the new label and do the inverse transform function on our prediction. Therefore, we subtract the mean value from each CP and do the sigmoid transformation on them:

$$S(x) = \frac{1}{1 + e^{-c*x}}$$

In most of cases, c in the equation above equals to 1, however, to control how smooth the result is, we make it a parameter which can be tuned to find a function that has the best performance. Here, we test the transformation with c = 0.5, 0.75, 1.0 and get Figure 6. In these graphs, we can tell that the slope is the smoothest when c = 0.75, which is nearly a perfect straight line. Except sigmoid, we also apply log function on CP, turning them to smaller values which scatter in a smaller interval. We use all three result of sigmoid, log and the original one in our experiment, in order to find out which one leads to the best performance

3.4 Model Ensemble

Although in the 1st subsection of this section, we proposed several models to solve the problem, each model may have their pros and cons. Due to the difference of optimization function, different model may better fit to part of the dataset and perform poorer in the rest of the dataset. So in order to reach even higher result, we can do some ensemble in our models, that is, to utilize the results of models to create a new result. Some intuition can be explained for this activity: if the models are all weak, we can use the collective intelligence to solve a hard problem; if a model is strong, the risk of overfitting can be reduced by cooperating with other models. The detailed equation of blending is given as follows:

$$
\begin{aligned}
&New\ prediction[i] \\
&= \textstyle\sum_j weight[j] * prediction[j][i] \\
&for\ all\ target\ i, model\ j
\end{aligned}
$$

The weighting of each model is obtained by the brute force search for best validation performance.

3.5 Framework

As a conclusion of this section, we provide figure 7 as our final framework to summarize the processing flow. Initially a 2 stage feature selection is presented. *Bunch of File Selection* is used for filtering redundant features in the large dataset. Then *compact Genetic Algorithm* is provided to select decisive features.

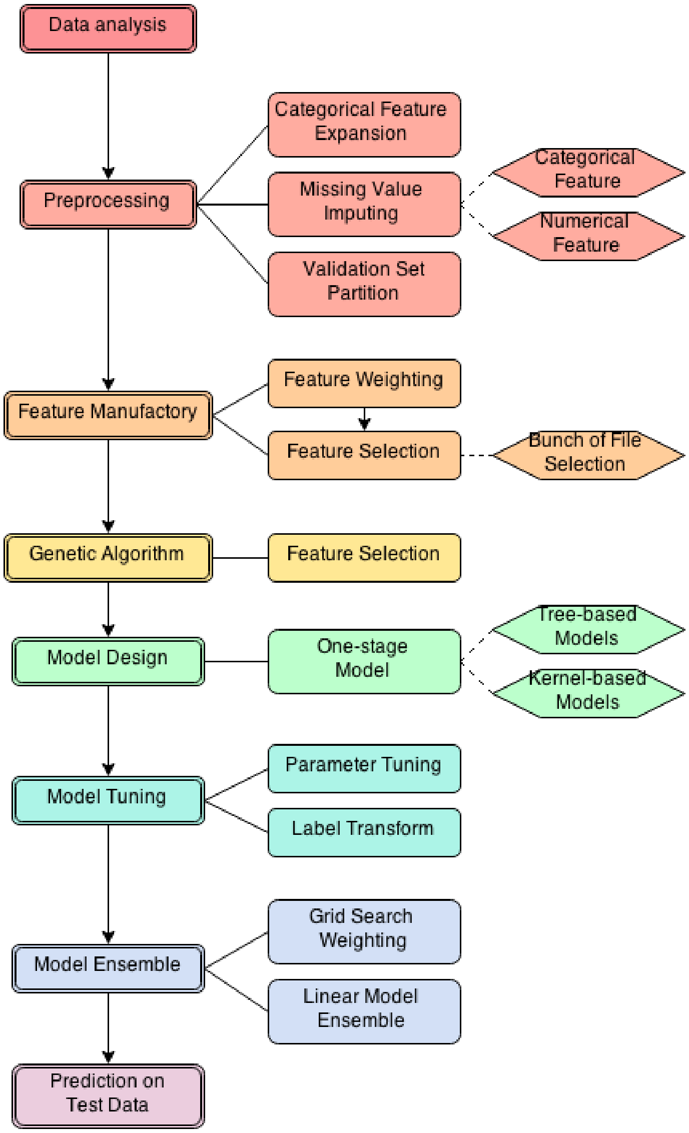


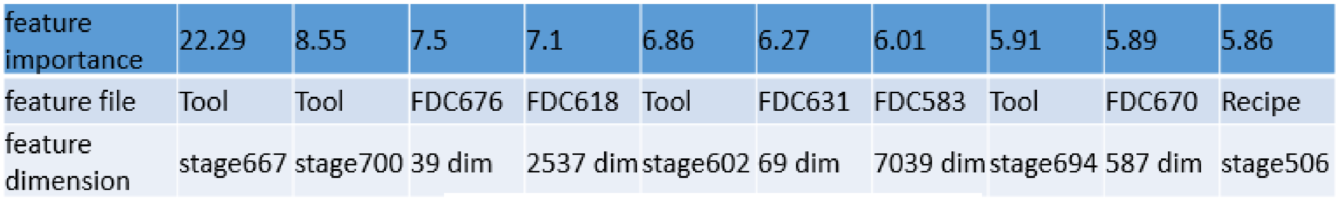
Figure 6: our final framework

Afterwards, we conduct experiment on various models and ensemble to create precise result.

4. Experiment Design

4.1 oostrapping

In order to measure our performance and not to overfit training data, we need to find out a validation set. There are several ways to select a validation set, including cross validation, leave-one-out validation, bootstrapping, cut by distribution etc. However, both cross validation and leave-one-out validation are time consuming, which takes too much time for each experiment, besides, there is no



feature importance	22.29	8.55	7.5	7.1	6.86	6.27	6.01	5.91	5.89	5.86
feature file	Tool	Tool	FDC676	FDC618	Tool	FDC631	FDC583	Tool	FDC670	Recipe
feature dimension	stage667	stage700	39 dim	2537 dim	stage602	69 dim	7039 dim	stage694	587 dim	stage506

Table 1: top 10 important feature

significant difference between distribution of train and test features. As for bootstrapping, it's a simple way to choose the sub-training set by randomly select a sample from dataset with size N for N time (a sample can be selected more than once), for those data which are not selected, they become the validation set.

4.2 BThe derivation can be demonstrated as following:

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n=\frac{1}{\mathrm{e}}$$

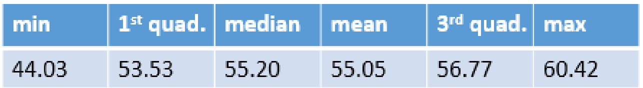
Compare with other methods, bootstrapping is a better way because it doesn't take too much time for each experiment and it is fair in this competition since the distribution of test and train seems quite the same. That's why we choose it as our way to build a validation set.

4.3 Parameter Tuning

For each model, there are several parameters should be tuned to get the best performance. In this competition, we use the simplest way -- grid search, that is, set all the values we want to test to each parameter, try all possible combination for parameters and find out which one have the best performance.

4.4 Evaluation

In order to demonstrate our internal performance, we must set up a metric evaluating our model performance. Although the CP value is the percentage value between 0 and 100. We can model it with several metrics related to information theory, such as mutual information and cross entropy.



min	1st quad.	median	mean	3rd quad.	max
44.03	53.53	55.20	55.05	56.77	60.42

Table 2, 5 number summary of CP

However, according to Table 2, the five number summary of the target label, we can notice that most values of CP are centered among [53.53, 56.77]. Therefore, adopting mutual information and maximum entropy may not be useful because most values of CP are similar. In order to better evaluating the performance, we adopt MSE (mean square error) to evaluate our internal performance.

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n}\left(\widehat{Y_i},Y_i\right)^2$$

5. Experiment Result

According to the framework in section 3 and experiment design in section 4, we conducted a series of thorough experiments and report our in-depth discoveries.

5.1 Feature selection

During our 1st stage Bunch of File Selection, only 83939 features are given positive weighting from totally over 200 feature files. Top 10 important features are listed as Table 1. We can notice that except for Tool features, in semiconductor manufacturing is critical. The best performance 2.26 is obtained during this stage using GBM model. Since only 83939 features are given nonzero weighting in this stage, we keep all feature with positive weight. top-10 important feature are mostly from FDC file. This can also support that the usage of FDC.

As for the 2nd stage, finally 41957 features are selected from cGA. Almost half of the features are again abandoned from our feature

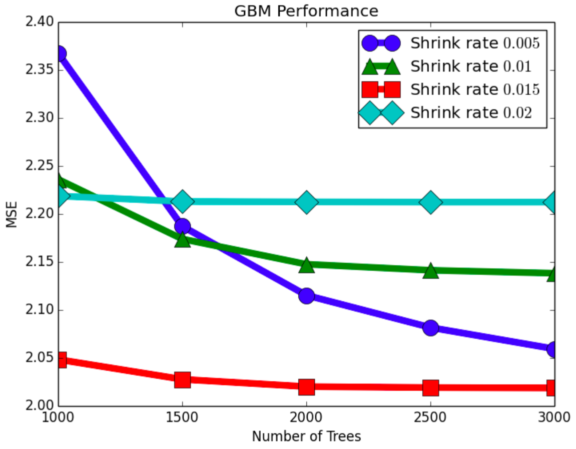


Figure 8: GBM performance with parameters

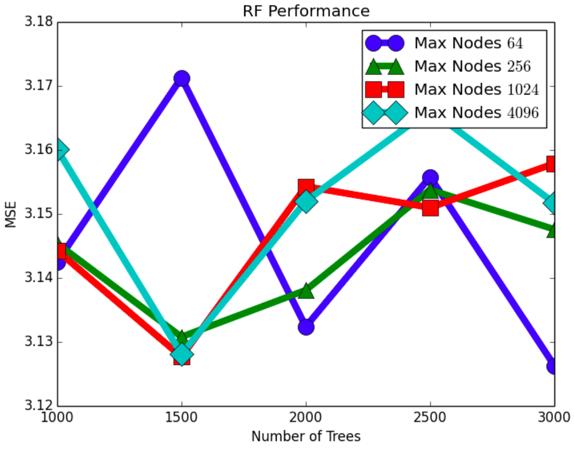


Figure 9: RF performance with parameters

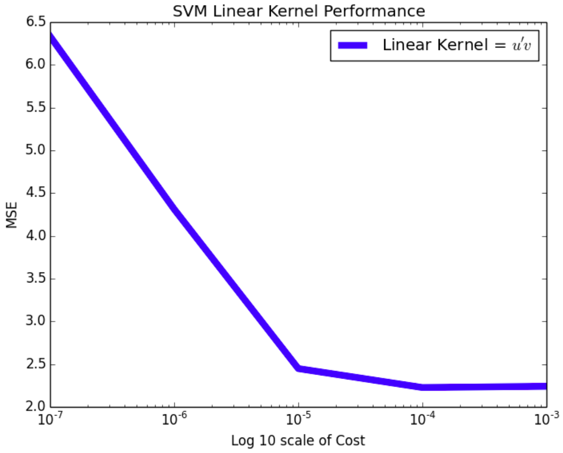


Figure 10: SVML performance with parameters

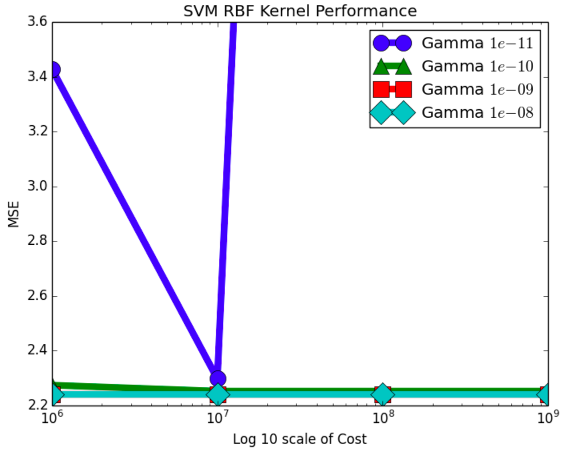


Figure 11: SVMR performance with parameters

selection, but it also boost the performance from 2.26 to 2.01.

From Figure 7, we can clearly see the dramatic proportion difference of FDC data and 1st stage features. Almost all features used in the final feature set are FDC data. However, since the proportion of 1st stage feature are such tiny. They may play a big role in predicting the eventual result. Note that the 1st stage features refers to the feature set given in the competition 1st stage

5.2 Models

Our experiments are designed using four models, including Gradient Boosting Machine (GBM), Random Forest (RF), Support Vector Regression (SVR) with Linear Kernel and Support Vector Regression with Radial Basis Function (RBF) kernel, which are shown as Figure 8 to 11.

We use compact grid-search algorithm to find proper parameters for these models.

For GBM, the minimum MSE is 2.01 when shrinkage rate is 0.015 with 3000 trees. It is

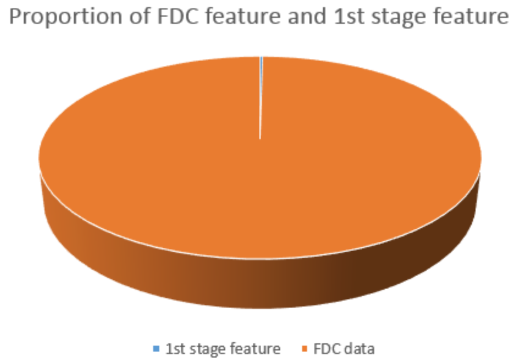


Figure 7: proportion of FDC data and 1st stage

shown that when the number of trees increase, the MSE will decrease for all four shrinkage rates. It is the most stable models in our experiment.

For RF, the minimum MSE is 3.1267 when using maximum 64 nodes and 3000 trees. However, its performance is very unstable with these two parameters in our experiment results.

For SVM with Linear Kernel, the minimum MSE is 2.2268 when cost is 10^{-4}. It is quite surprising that linear model has such outstanding performance.

For SVM with RBF Kernel, the Minimum MSE is 2.2406 with gamma 10^{-9} and cost greater than 10^{5}. It is quite interesting for us that its best behavior will be roughly bounded after cost parameter surpassing certain value. This phenomenon can be observed when gamma is equal to 10^{-10}. This rule will be broken only with extremely large cost, in contrast to gamma, i.e. when cost is 10^{7} or 10^{8} and gamma is 10^{-11}, and its MSE will arise sharply.

After performing a wide-ranged parameter search, we finally find a better setting for parameters. In our experiments, RF and SVM have similar performance, whereas GBM defeats these three models with MSE 2.01.

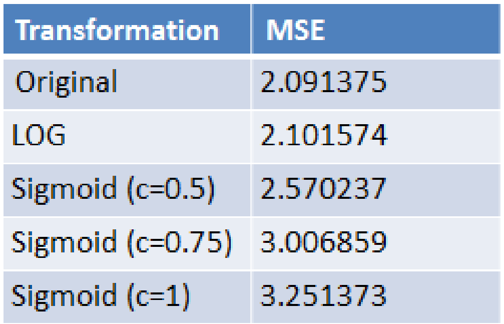
It is shown that the minimum MSE for these four models are close, but the performance of GBM defeats the others.

5.3 Label transformation

We test the performance of transformed label with GBM under same validation set. Unfortunately, performance (see table 3) doesn't improve. As you can see, the performance of log transformation is almost the same as the original one, however, for sigmoid transformation, the bigger the parameter c is, the worse our performance is.

5.4 Ensemble

We combine the result of each model with different weight, in order to find out which leads to the best performance. With GBM weighted 2, linear kernel SVM weighted 1 and ignore other two models, we have a 0.04 improvement on validation set compared with our best single model result (by GBM).



Transformation	MSE
Original	2.091375
LOG	2.101574
Sigmoid (c=0.5)	2.570237
Sigmoid (c=0.75)	3.006859
Sigmoid (c=1)	3.251373

Table 3: Performance of different label transformations

6. Discussion

6.1 Feature analysis

With the needs for processing the huge amount of features and dealing with the huge amount of features, the most challenging part of the competition is the limitation on the computation machine, i.e. the size of main memory. Since we cannot even load the entire data into memory, the selection of quality feature and demission of useless feature is the key point of this competition. A classic method dealing with high dimensional data is called dimension reduction. However, two drawbacks should be considered in this competition: 1st the unignorably large time complexity. 2nd the meaning of each feature will be transformed to latent space, that is, the meaning of each feature will be lost after the processing of dimension reduction method. Our feature selection method is thus proposed based on this consideration and simultaneously solves the above problems. The most noticeable advantage of our method is that we select the most important part of feature so we can easily observe the meaning of important features since we didn't destroy the value of each feature at any given time.

6.2 Model comparison

There are other models that we does not use in this competition, including probabilistic based models and deep learning based algorithms. We do not use probabilistic based models, e.g. Bayesian model, or deep learning based models, e.g. Convolutional Neural Network (CNN) because their high computational cost.

Our experiment results show that GBM has better performance than RF and SVM models. It is acceptable that empirically, GBM has great performance in real world problems than the other models. However, there is another factor which may cause this result. In our two-stage feature selection, because of the time factors, we only use GBM to select features. It is possible that other models are favorable of the abandoned features.

It is worth noting that linear model has great performance when compared with other models. It can be interpreted as that this data is not so hard for machine learning methods. That is to say, we should avoid using too complicated models to prevent overfitting problem.

6.3 Conclusion

In the era of big data, there are lots of opportunities to achieve what we could not do and discover what we could not see. However, not until we taking effort into data mining can we observe the knowledge in the big data. In this competition, we investigate several methods making prediction on the semi-conductor manufacturing data and define the state-of-the-art. During our feature selection step, we also gain knowledge for the feature and target CP, that is, according to the performance of prediction result, we discover critical elements of feature determining a precise CP. This novel method not only solves the memory limitation problem in the competition but also yield important insight of the relation of FDC data, Tool, other attributes and CP. With the framework of our data mining process

In this competition, we make the following contribution:

A. A data mining framework comprises the merits of theoretically guarantee of machine learning and realistic issue of processing data.

B. A correct analysis result ensured by the validation performance.

C. An original analysis method consists of a comprehensive concern and accurate performance is presented.

D. A practical data mining framework ensures the feasibility of processing enormous amount of semiconductor data.

7. REFERENCES

[1] Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." *Neural processing letters* 9.3 (1999): 293-300.Suykens, Johan AK, and Joos Vandewalle.

[2] Smola, Alex J., and Bernhard Schölkopf. "A tutorial on support vector regression." *Statistics and computing* 14.3 (2004): 199-222.

[3] Cristianini, Nello, and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[4] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." *Annals of Statistics* (2001): 1189-1232.

[5] Liaw, Andy, and Matthew Wiener. "Classification and Regression by randomForest." *R news* 2.3 (2002): 18-22.

[6] Kearns, Michael, and Dana Ron. "Algorithmic stability and sanity-check bounds for leave-one-out cross-validation." *Neural Computation* 11.6 (1999): 1427-1453.

[7] Aporntewan, Chatchawit, and Prabhas Chongstitvatana. "A hardware implementation of the compact genetic algorithm." *Proc. 2001 IEEE Congress Evolutionary Computation, Seoul, Korea*. 2001.

[8] Efron, Bradley, and Robert Tibshirani. "Improvements on cross-validation: the 632+ bootstrap method." *Journal of the American Statistical Association*92.438 (1997): 548-560.