Big Data Analytics for Semiconductor Manufactoring

BDC103C
NULL

February 3, 2015

Outline

Problem Definition

Given *Carrier*, *Chamber*, *Fab*, *Recipe*, *Tool* and tremendous amount of *FDC* data, we aim to:

- Create **novel** and **realistic** high dimensional data mining framework
- **Precisely** predict the value of CP
- Obtain the **decisive** factors that determine the outcome

Proposed Framework

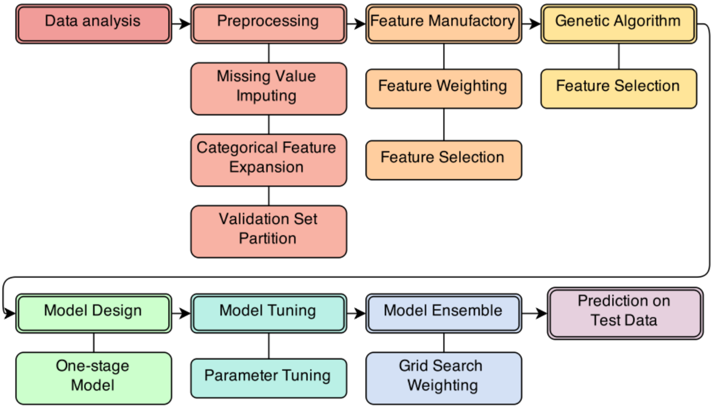


Figure: our framework

Missing Value Imputation

Categorical Features

- Most frequent elements

Numerical Features

- Average value of elements

Time-series Features

- Median value of neighbor instances

Categorical Feature Expansion

For range-based model, such as SVM.

Take *Chamber* feature for example:

- Original: $\begin{bmatrix} chamber1 \\ chamber2 \\ \vdots \end{bmatrix}$
- Expanded: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \end{bmatrix}$

Validation Set

Prepare a hold-out set to validate our internal performance

Figure: subtrain and validation

How do we determine the portion of validation set / training set?

- Each instance get $\frac{1}{n}$ probability of being training data
- $\lim_{n \to \infty}(1 - \frac{1}{n})^n = \frac{1}{e} = 0.368$
- We use 0.368 training data as validation set and leave the rest as subtrain

High dimensional data

Two well-known methods for dealing with high dimensional data:

- Dimension reduction, such as PCA, auto-encoder, ...
- Feature selection

Discussion:

- Main difficulty of applying dimension reduction is its huge complexity; and there is no performance guarantee. (**time-consuming**)
- In contrast, applying feature selection based method can solve this problem. (**realistic**)
- Feature selection will also boost regressor performance. (**effective**)

Bunch of file selection

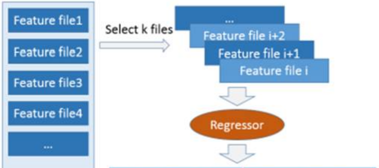
Lots of feature available. How can we extract meaningful features?

- **Divide**:
  - Sample a portion of features
- **Conquer**:
  - Extract meaningful features
  - Maintain importance in a table

Bunch of file selection

How can we extract meaningful features?

- Based on feature importance in certain regressor (tree-based model usually)



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

Figure: first-stage feature selection

Discussion

How do we fill values in the feature importance table?

- Divide and **Conquer**?
  - Fill the importance of features from tree-based model directly.

Discussion

How do we fill values in the feature importance table?

- **Divide** and **Conquer**?
  - Fill the importance of features from tree-based model directly. (X)
- Divide and Conquer:
  - Performance of each model reflects the behavior of a small subset of features.
  - Normalization is needed!!
- Feature Importance$[f_i] = \frac{\text{Regressor Importance}[f_i]}{\text{Feature Set Performance}[f]}\ \forall$ subset f

Compact Genetic Algorithm

What will we do next?

- The previous method is based on regressor importance.
  That is, we can abandon what our model cannot learn from.
- However, we want to further obtain which feature is **decisive**.
  That is, we want to optimize:

$$\underset{\text{featureSet}}{\operatorname{argmax}} \text{Regressor Performance}(\text{featureSet})$$

Compact Genetic Algorithm

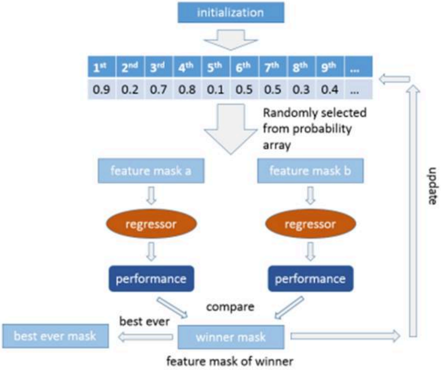


Figure: second-stage feature selection

Tree-based Model

Gradient Boosting Machine (GBM)

- Iteratively build new trees to correct current error
- Well performance in many data mining competition
  - Best performance in preliminaries in our experiment

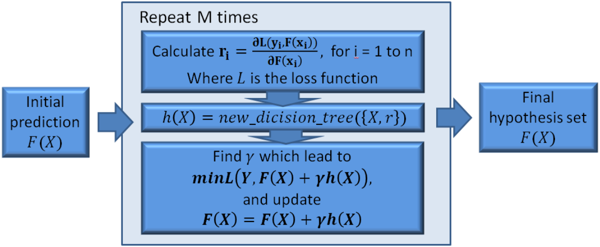


Figure: GBM workflow

Kernel-based Model

Support Vector Regression (SVR)

- Extended from SVM
- Find the hyperplane in feature space having minimized loss
  - Optimization function (primal):

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to } |y_i - \mathbf{w} \cdot x_i - b| \leq \epsilon \ \forall i$$

- Kernel trick
  - Project feature space to higher dimensional space
  - For example, linear kernel, polynomial kernel, RBF kernel, etc.
    - Linear kernel has the best performance in our experiment

Parameter Turing

Grid Search

- Try different combination of parameters
- Put measurement on a fix validation set
- GBM
  - **Tree number, shrinkage**
  - Decide the depth of model
- SVR with linear model
  - **Cost**
  - Decide the fitness of model, in terms of training data

Ensemble

Grid Search

- Give different weight ($1 \sim 10$) to the prediction of different model
  - Models we use: RF, GBM, SVR (Linear and RBF kernel)
- Put measurement on a fix validation set
- In our experiment, we find that RF and SVR with RBF kernel have 0 weight when we get obtain the best performance

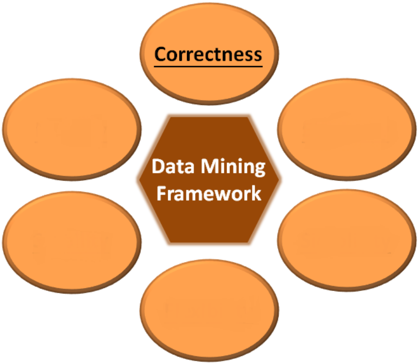
Performance

We take the ensemble result as our final submission

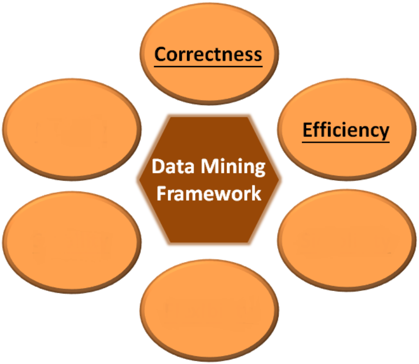
Model	Parameter	MSE (validation)
GBM	N_trees=3000, shrinkage=0.015	2.01
SVR (linear kernel)	Cost=1e-4	2.23
Ensemble	GBM weighted 2, SVR (linear) weighted 1	1.97

Figure: performance comparison

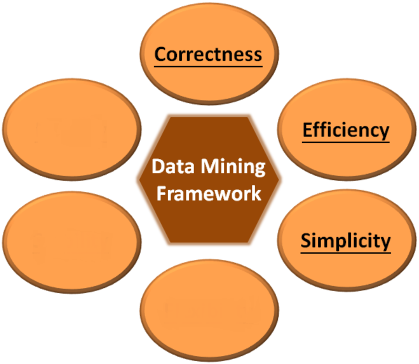
Conclusion



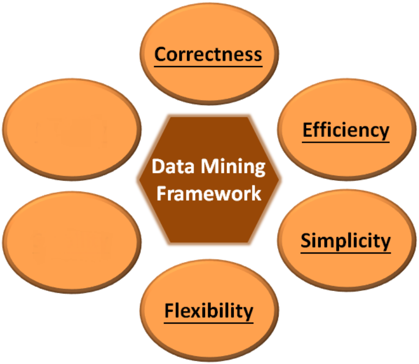
Conclusion



Conclusion


Correctness
Efficiency
Data Mining Framework
Simplicity

Conclusion


Correctness
Efficiency
Data Mining Framework
Simplicity
Flexibility

Conclusion


Correctness
Efficiency
Simplicity
Flexibility
Stability
Data Mining Framework

Conclusion


Correctness
Novelty
Efficiency
Data Mining Framework
Stability
Simplicity
Flexibility