Mercedes-Benz Bench Test Time Estimation

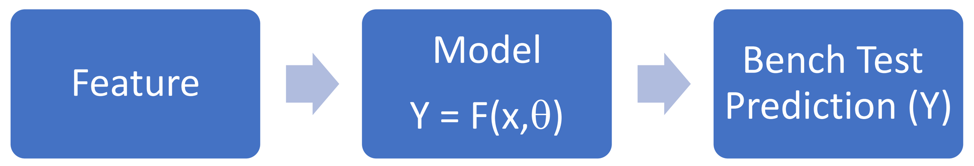
UC San Diego

Lanjihong Ma, Kexiong Wu, Bo Xiao, Zihang Yu

Overview

It would be promising to manage test time and further optimize the whole test system if there exists an algorithm to predict the bench test time with a combination chosen from roughly 400 features.

We aim to design a model with input as interested feature combinations and output as the correspondent bench test time prediction.



The input for our model is feature combination:

x_n=1 means the n^{th}feature is included in the bench test.

x_n=0 means the n^{th}feature is removed from the bench test.

The feature combination is represented by a array which only consists of 1 or 0.

Data

- **Data Size**
  - 4209 Training Sets and 4209 Test Size.
  - 376 Logistic features and 8 Classification features.

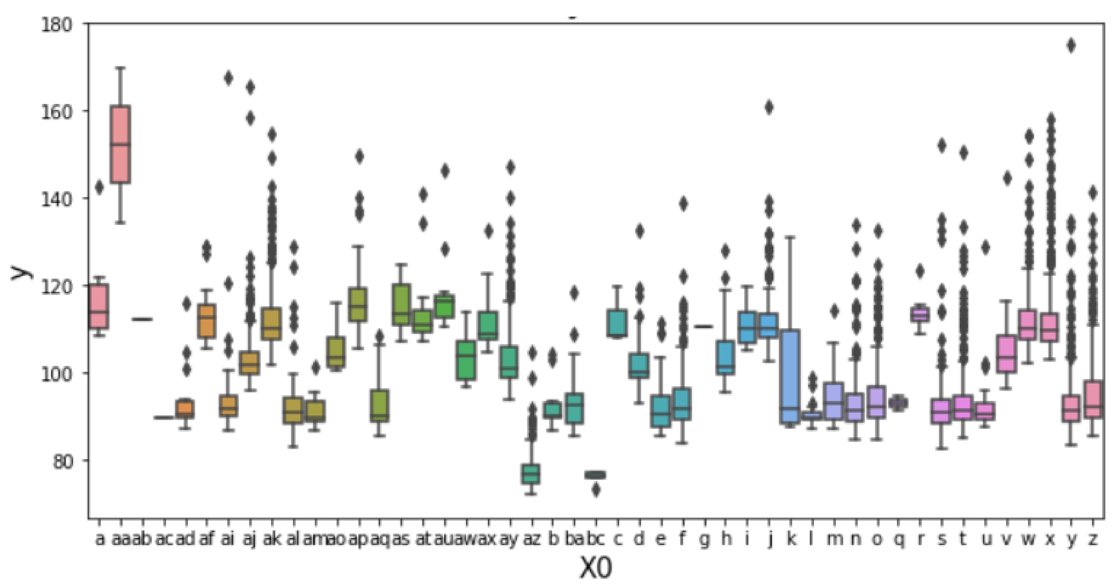


Figure 1. Feature distribution

Feature Preprocessing

- **Feature Selection**

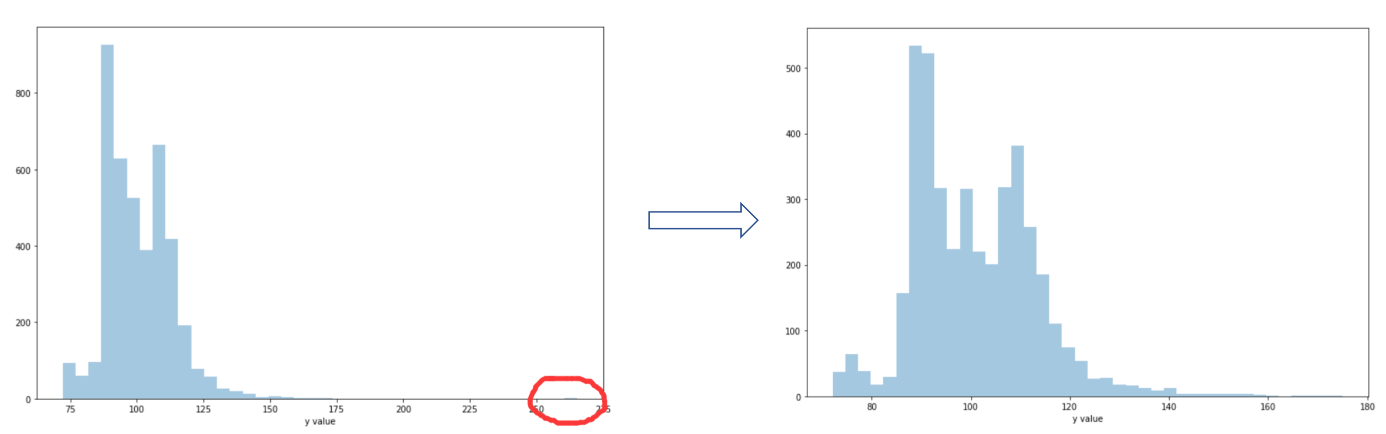


Figure 2. Feature selection. For the data distribution, we remove the outlier from raw input data.

- **Feature Extraction**

The goal of feature extraction is to minimize the effect from sparse signals and separate classification features for further processing.

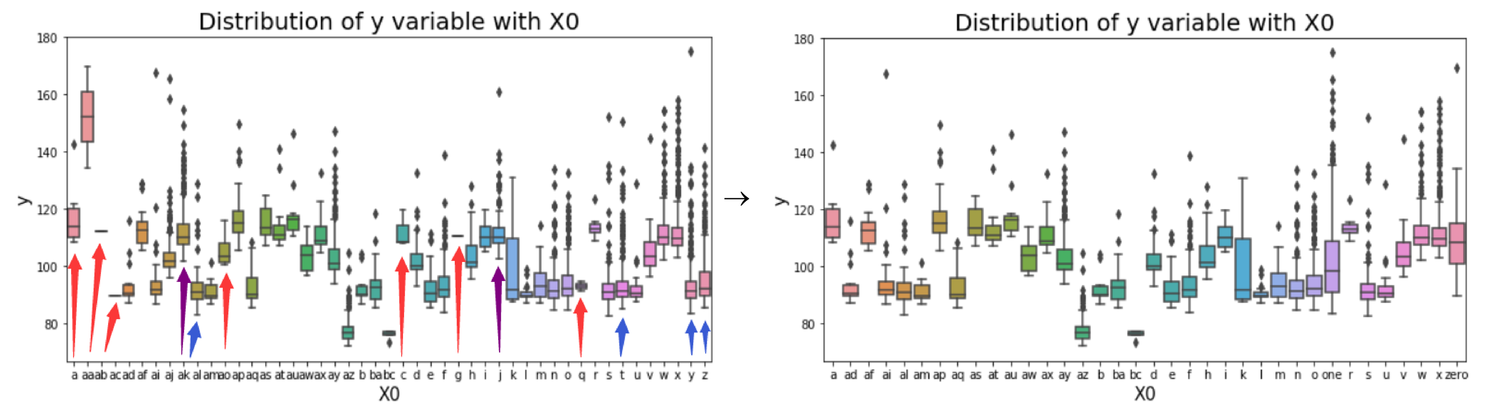


Figure 3. Feature extraction. Here we set the threshold as 0.1% and then calculate the KL divergence between the distributions for classification features and manually combined those close distributions. Red arrows show the sparse signals which are supposed to be filtered out. Blue and purple arrows show close distributions need to be combined.

Model

- **Model Candidates for Selection**

XGBoost	K-Nearest Neighbors	Random Forest
Stochastic Gradient Decent	Linear Regression	Extra Trees
Support Vector Machine	Decision Tree	Gradient Boosting

For XGBoost, the model is tree ensemble including a set of classification and regression trees. We can write the model in the form

$$\hat{y}_i = \sum_k^K f_k(x_i),\ f_k \in \mathcal{F}$$

where K is the number of trees, f is a function in the functional space F, and F is the set of all possible CARTs. Therefore, our objective to optimize can be written as

$$obj(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

The next step is tree boosting, inclosing regularization, additive training, compute loss function. Then we get

$$obj^{(t)} = \sum_{i=1}^n \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) + constant$$

Where the g_i and h_i are defined as

$$g_i = \partial_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right);\quad h_i = \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$$

Renormalizing the tree model, we can write the objective value with the t-th tree as

$$obj^{(t)} \approx \sum_{i=1}^n [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^T w_j^2$$

$$= \sum_{j=1}^T [\left(\sum_{i \in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2] + \gamma T$$

Where $I_j = \{i | q(x_i) = j\}$ is the set of indices of data points assigned to the j-th leaf. We can have $G_j = \sum_{i \in I_j} g_i$ and $H_j = \sum_{i \in I_j} h_i$

$$obj^{(t)} = \sum_j^T [G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2] + \gamma T$$

The best w_j and best objective reduction we can get is

$$w_j^* = -\frac{G_j}{H_j + \lambda};\quad obj^* = -\frac{1}{2}\sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$$

The last equation measures how good a tree structure q(x) is. The smaller the score is, the better the structure is.

Result

- **Test Data**
  - Divide raw data into 75% training set and 25% cross-validation set.
  - 4209 test samples.
- **Model Evaluation**

We choose the R-square score to assess our model.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where total sum of squares $SS_{tot} = \sum_i (y_i - \bar{y})^2$, sum of squares of residuals $SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$

Model	Train error
Linear Regression	-1.0227464886939429e+24
Stochastic Gradient Decent	-1.0520412148310016e+17
Support Vector Regression(Linear Kernel)	0.5310703895479278
Support Vector Regression(Polynomial Kernel)	0.4292692789452468
Support Vector Regression(RBF Kernel)	0.4258513541680711
K-Nearest Neighbors	0.4839903962740162
Decision Tree	0.30902602134884694
Random Forest	0.5671153256808917
Extra Tree	0.3853554827238316
Gradient Boosting	0.6406754701863584

Table 1. Train error of different model

- **Model Combination**
  - For cross-validation set, we have highest R-2 score using following model combination

90% Gradient Boost + 8% Random Forest + 2% Support vector(linear Kernel)

  - For test data set, we add XGBoost and achieve highest score in Kaggle.

80% XGBoost + 10% Random Forest + 5% Extremely Randomized Forest + 5% Gradient Boosting

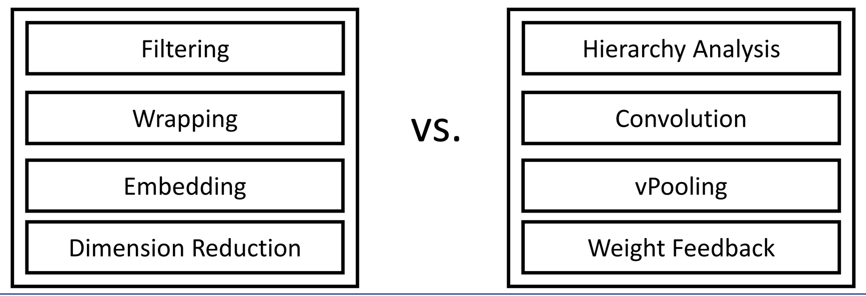
Discussion

In our project, we finally can predict bench test time if we can know specific feature combinations. With an accurate bench test prediction, the manufactures could arrange their resources and time during their manufacturing process in a more efficient way. Also with the accurate bench test prediction, factories can detect malfunctions during the bench test if the bench test takes longer than the model predicts.

Since the process is a discrete problem rather than a continuous problem, the coefficient in discrete problem cannot be so linearly sensitive, so a lower R^2 score is allowed. Before we started, we estimated R squared could be something about 0.5. After we finished the project, our R^2 score is 0.55647, which meets our expectation. Besides, we found that the highest score on Kaggle is 0.5550 which also strongly matches our expectation at the beginning.

Future Work

- Using Deep Neural Network for data featuring.
  - Get Feature Hierarchy by training each layer.
  - The training is based on previous layer's output.
- Detect the latent structures of our data, with feature hierarchy.
- Deep Neural Network enables the automatic feature extraction.
- Feature Extraction without DNN vs. with DNN



Contact

Kexiong Wu
University of California, San Diego
9500 Gilman Dr., La Jolla, CA 92093
kew135@eng.ucsd.edu
(858)260-9076

References

1. Ben-Hur, Asa; Horn, David; Siegelmann, Hava; and Vapnik, Vladimir N.; "Support vector clustering"; (2001); Journal of Machine Learning Research, 2: 125–137
2. Davies, Alex; Ghahramani, Zoubin (2014). "The Random Forest Kernel and other kernels for big data from random partitions". arXiv:1402.4293
3. J.H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine“.
4. Devore, Jay L. (2011). Probability and Statistics for Engineering and the Sciences (8th ed.). Boston, MA: Cengage Learning. pp. 508–510. ISBN 0-538-73352-7
5. Samuel, Arthur L. (1988). "Some Studies in Machine Learning Using the Game of Checkers. I". Computer Games I. Springer, New York, NY. pp. 335–365. doi:10.1007/978-1-4613-8716-9_14. ISBN 9781461387183.