The Magic of Feature Engineering: Higgs Boson Detection using Simple Models

Yiren Cao, Lucas Dodgson, Zhentao Liu *EPFL, Switzerland*

***Abstract*—In this report we investigate a series of models on the Higgs Boson detection task. By combining multiple feature engineering methods, including customized outlier clipping, cross terms generation, and principle component analysis, we are able to drastically improve the performance of the models applied to this task, reaching an accuracy of 83.7%.**

I. Introduction

Machine learning (ML) has become one of the most powerful methods for solving data science problems in many fields. One such example is the classification task of detecting the Higgs Boson particle. This task presents multiple challenges: 1) A significant proportion of the values are missing in the dataset. 2) The distributions of some features are skewed and present multiple outliers. 3) The dataset is quite large, and training will be computationally expensive when complex models are involved.

In this paper, we propose a hybrid model to solve this task. Our model combines unsupervised approaches (PCA, KNN) with supervised methods (various types of regression) in different phases of the pipeline. To begin with, we utilize special properties linked to *PRI-jet-num* of this dataset to train separated models for different subgroups, resulting in a better final performance compared to using a single model for the entire dataset. Moreover, our main area of interest goes to task-specific feature engineering, such as how to handle missing data, perform feature transformation, and conduct feature selection. Innovatively, we generate cross terms combined with PCA which drastically improves the final accuracy.

We start off by introducing the feature engineering (II), and then we conduct the model selection (III), followed by our results and discussions (IV). In the end, we draw a summary (V).

II. Feature Engineering

Figure 1 shows our feature engineering pipeline. After splitting the data into subgroups, we clip outliers, fill in missing data, generate cross terms, perform the polynomial expansion followed by PCA for feature selection, hyperparameter tuning and finally model selection.

Exploratory Data Analysis. The dataset contains 30 features with 250000 entries in the training set and 568238 entries in the test set. The label to predict is binary. All variables are floating points, except *PRI-jet-num* which is a discrete variable containing only integer values (0 to 3). While looking into the dataset, we observe that the large proportion of missing values is seemingly strongly correlated to *PRI-jet-num*. In addition, we also observe extreme outliners in many columns which we will handle at a later stage.

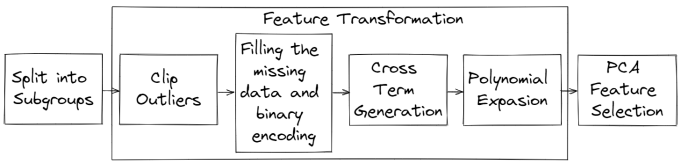

Figure 1. Pipeline of our feature engineering

Splitting the dataset.

Based on *PRI-jet-num*, we divide the dataset into 3 subgroups, depending on if the value is 0, 1, or some other value. As a result, some features may only be present in one subgroup but not in another. We believe it is possible that each *PRI-jet-num* is associated with a different population in the real world and that utilizing different models to train and forecast various populations will be more accurate and natural.

Furthermore, to evaluate if splitting into subgroups is sensible, we perform the following feature engineering and model selection on both the entire dataset and the *PRI-jet-num* subgroups separately.

Clipping Outliers. Manual inspection of the distributions reveals that multiple features have a few extreme outliers. Therefore, for each feature we trim such outliers.

Filling Missing Values. We evaluated a variety of options including filling with zero, the mean, the median as well as the KNN-based method. Whereas iterating the similarity computation for all the feature pairs is computationally expensive. Therefore, to strike a balance between performance and efficiency, we use a Monte-Carlo-based KNN method to fill in the missing values.

For each jet-num subgroup, we remove the column if it is completely missing, which largely reduces the dimension for several subgroups. Then for the remaining missing values, we fill them up using one of the above-outlined techniques. Additionally, since the missing values play an important role in the dataset, we preserve this information by adding a new column indicating if a value was originally missing or not.

Cross Term Generation. As we know, some features potentially interact with others. By adding interaction terms to the regression models, the relationships among the variables in the model will be expanded, which may result in better performance. Despite the fact that including k-way ($k > 2$) interactive terms can be more beneficial, we focus on

pairwise interaction terms in order to maintain a reasonable computation cost. Hence we perform manual selection and add the corresponding cross terms.

Dimension Reduction. Adding interactive terms and polynomial expansion significantly increases the dimensions for each subgroup. To avoid the curse of dimensionality, we perform PCA to obtain a lower-rank approximation and filter out some redundant information. It also helps to mitigate overfitting.

Tuning and Feature Selection. We then directly train our model on the lower-dimension representation. This brings the advantage that it also increases the computational speed. For both the polynomial degree and the PCA dimension, we use a hyper-parameter search to choose the best one.

III. Model Selection

We focus on the following 6 basic models: Least squares regression using normal equations, linear regression using both classical gradient descent and stochastic gradient descent, ridge regression using normal equations, and both logistic and regularized logistic regression using classical gradient descent.

IV. Results and Discussions

To evaluate the performance of our model, we used an 80 - 20% train-validation split. For each stage of our pipeline, we compare the different choices made for that stage. First, we evaluate if splitting the dataset brings any advantage. We fill the missing values with the mean and do the remainder of the pipeline as specified above, except for the omission of the cross-term step. We evaluate the 6 models with optimal hyper-parameters and choose the best result from those. Table I contains the results for this. We see that splitting the dataset and then using different models per subset brings a significant advantage compared to using the full dataset.

Method	Accuracy	F1 Score
Trained using combined dataset	0.814	0.716
Trained using split dataset	0.823	0.728

Table I
Comparison of the effect splitting our dataset by PRI-jet-num has.

Next, we compare the different ways of filling in the missing data. Table II shows, that all our methods perform very similarly. While filling in the missing data is extremely important, the exact method of doing so is less. In the following, and in our final model we thus use the median to fill in our missing data.

Method	Accuracy	F1 Score
Zero	0.823	0.726
Mean	0.823	0.728
Median	0.822	0.726
KNN-0.2	0.822	0.725

Table II
Comparison of different filling missing value methods

Table III then contains the comparison of the six different models. The interesting result is that the least squares method, which is the simplest and fastest model here, performs the best overall by a significant margin.

Method	Accuracy ↑	F1 Score
least squares regression using normal equations	0.823	0.728
ridge regression using normal equation	0.815	0.715
logistic regression	0.747	0.584
regularized logistic regression	0.747	0.597
linear regression (stochastic gradient descent)	0.742	0.538
linear regression (classic gradient descent)	0.739	0.535

Table III
Comparison of different models

Finally, we analyze the effect of adding cross terms, we perform this analysis per jet number. In Table IV we can see the accuracy for each sub-dataset on the validation set and the benefit that the cross terms bring. We can see, that when using simple models, adding cross terms brings a significant advantage as it enables the model to learn from the interactions of our features.

Method	JetNum = 0	JetNum = 1	JetNum = others
Using cross term	0.851	0.813	0.842
Without cross term	0.848	0.807	0.832

Table IV
Comparison of accuracy when using cross term

We also evaluated, if using a combination of our models in a max-voting system would bring further improvements, but this was not the case. In future work, some potential improvements in model combination could be: Train on more models and the use of advanced model combination methods such as bagging and boosting.

As such, for our final model that gives 0.837 accuracy and 0.754 F1 score on the test set on AIcrowd, we choose the least squares on the split datasets with missing values filled by median and cross terms addition. For our hyperparameters, we settle on a polynomial degree of 17, 14, 8, a PCA dimension reduction of 1310, 1280, 1149, select 66, 83, 152 pairwise interactive terms for each of the three subgroups, JetNum = 0, JetNum = 1 and JetNum = others, respectively.

V. Summary

In this report, we show how careful feature engineering can allow even simple models, such as least squares, to perform extremely well in the difficult classification task. By spotting the pattern present in the dataset, and adding cross terms, well-performing models can be derived. There is future work on a variety of topics that we would like to explore, including adding k-way interactive terms to the dataset, using ensemble methods to combine multiple predictors, using other methods like the FM model to generate cross terms rather than manually creating them, and utilizing complex models such as neural networks.