Higgs Boson Challenge : Finding evidence of the particle's presence using Machine Learning

Niccolo Sacchi, Antonio [???] & Valentin Nigolian
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—The goal of this project was to use machine learning algorithms to effectively predict the decay signature of particle collisions at CERN, to understand if this event's signature was the result of a Higgs boson (signal) or something else (background). We have developed binary classification techniques for signal/background separation, and applied our methods to a test dataset in the Kaggle platform. A score of 0.829 was achieved after several implementations, which is not an ending point but a solid base point for further and more complex studies.**

I. Introduction

To achieve such a challenging and complex goal we needed to split up the process into subtasks, starting from a simple model, evaluating it, and trying to figure out how it could be improved. First of all we performed some exploratory data analysis to understand the training and the test dataset. The training data consist of 250000 events, with an ID column, a label column, and 30 feature columns. The test set has 568238 events, and is missing the label column. We tried to run different models on the whole dataset at first, and despite it was not performing well we figured out that the logistic regression was the way to go. The following section shows how we managed to get a final good result improving step by step trough feature processing and methods tuning.

II. Models and Methods

There were two main parts of developing our ML system, data analysis and algorithmic design. While the first one focused on the nature, intricacies and interconnections of the raw data and its preparation, the second one focused on the treatment of said data after refining. Note that both aspects were studied and improved separately and concurrently. Let us now delve a bit further into those two aspects.

A. Data Analysis and Exploration

With a dataset of 250'000 items and 30 features, there was a lot of data to work with. To get a better sense of its nature, we used various mathematical tools. Those were applied on the whole dataset (train + test) when possible and only on the training set when the analysis was related to the prediction. The following are the tools we used :

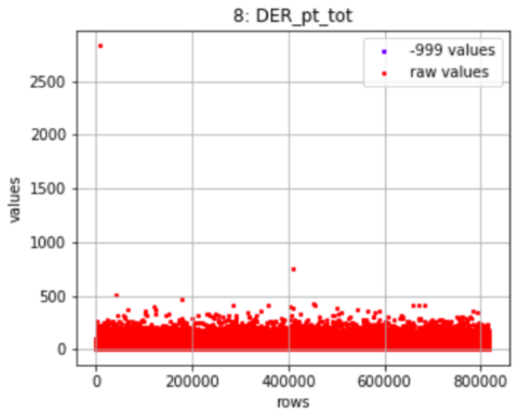

Figure 1. Values of all data items for the DER_pt_tot feature. The graph shows a relatively small number of outliers. In this example, all items with values above 400 for this feature were removed.

- Outliers Analysis The first thing to do to get a better sense of the data was to plot it. We thus plotted the whole dataset by feature and observed if could find any outliers or discrepancies. There were indeed multiple outliers which we removed from the dataset. See fig. 1 for an example.
- Distribution Analsysis : We wanted to see how features values were distributed over both the signal data and the background data. Indeed, we made the assumptions that if two features had a similar distribution between the two sets, then this feature would only have limited prediction power. Among all features, four were spotted to have the almost exact same distributions. Those are ”PRI_tau_phi”, ”PRI_lep_pt”, ”PRI_lep_phi” and ”PRI_met_phi”. We decided to try to drop those columns to see to what extent it changed the resulting predictions, if any.
- PRI_jet_num-based Split : After looking further at the data, we noticed that there were a lot of -999 values spread on the dataset. More importantly, we noticed that the amount of those values depends greatly on one particular feature, which happens to be the same categorical feature : ”PRI_jet_num”, representing the number of ”jet events” (a physics term unknown to us) occurring at every event. For instance, if this feature has value 0, then 10 other features will have only -999 values and 26% of the first feature will be -999, as per if it has a 3 value, then no feature is all -999

and only 1.4% of the first feature will be -999. For this reason, we decided to drop the following features depending on the ”PRI_jet_num” feature’s value. This lists the features dropped by their index in the feature list.

  - jet = 0 : [4, 5, 6, 12, 22, 23, 24, 25, 26, 27, 28, 29]
  - jet = 1 : [4, 5, 6, 12, 22, 26, 27, 28]
  - jet = 2 : [22]
  - jet = 3 : [22]

  Note that we also dropped column 22 which corresponds to the jet feature itself. Indeed, we figured that by giving as much importance to this feature, we would not need it any more to classify our data.
- Correlation Analysis : After splitting the data into four subsets and making the assumption that two (or more) highly-correlated features must have roughly the same impact when predicting the label of an item, we computed the correlations between each pairs of features for different correlation values (e.g. ¿0.6, ¿0.8, ¿0.9 and =1) and identified the more or less correlated features.
- Standardization : To standardize the data we decided to use the whole dataset to compute the means and standard deviations while carefully not considering the -999 values and hardcoded those values so we could reuse them later without having to compute them every time.
- Percentiles Computation : A technique we tried and that allowed to increase the ratio of correctly predicted outputs is what we called dimension expansion. We divided each feature in N features using the percentiles. e.g. if N=2 we took the median and split a column in 2 new columns, one with all the entries below the median and one with all the entries above the median. This allowed to split the dimension of the input in different intervals in which the regression can be trained independently (imagine if you have to fit an exponential with a 1-degree polynomial, you split the input in 2 and use two 1-degree polynomials instead of 1). Similarly to the standardization, we computed the distributions of the items among the percentiles and stored them so we could reuse them without having to compute them again every time.
- We concatenated to each one of the four datasets its logarithm and noticed that the performance of our model increased. Therefore, we concluded that the output potentially depended logarithmically on some of the features.

With those tools applied to our data, we were ready to actually train our model.

B. *Algorithms and Techniques*

While preparing the data is an important part of any ML system, it only makes sense for specific techniques.

1) Method Selection: We thus tried different approaches using the methods presented in class and we used a small set of visualization tools to evaluate them :

- Bias/Variance Estimation : We tested our models by training them with an increasing number of data items and visualized how the performance changed. This allowed to see if the lack of performance came from high bias and/or high variance, thus giving hints about whether we should reduce the number of features or on the contrary, increase it.
-

Using these tools on ridge regression, stochastic gradient descent and logistic regression and it appeared that logistic regression was generally better than the other methods. We thus chose to refine this method and spend more time trying to increase its performance.

2) Hyperparameters Selection: After several trials we understood that a high polynomial degree did not increase much the ratio of correct predictions. Therefore, we applied some of the aforementioned techniques to increase the performance of our model. Here we list how and why we choose our hyperparameters:

- We used a 2-degree polynomial. Thanks to dimension expansion we could use a simpler polynomial in smaller intervals of each feature, i.e. we splitted every feature into five intervals therefore increasing the number of features by a factor of five. Moreover, we concatenated to each one of the four datasets its logarithm (before building the polynomial). These steps were taken in the following order: (1) split the features, (2) compute and concatenate the logarithm of the dataset, (3) build the polynomial.
- We used an array of gammas. We noticed that the higher the degree the lower was the gamma needed to have a stable gradient descent leading to a higher coverging time. Therefore, instead of using a scalar gamma we used an array of gammas and tuned them independently. In particular, this array of gammas is as long as the gradient of L(w) and allowed us to choose a different step size for different parts of the gradient therefore achieving more versatility. Another solution we tried is to normalize after building the polynomial, but we achieved a higher converging time with the latter therefore we opted for the array of gammas.

Notice that we decided not to use cross validation with logistic gradient descent because (1) we were using low degree polynomials therefore reducing the variance of our model and (2) we already used cross validation with ridge regression and did not obtained any noticeable overfitting.

III. Results

Since we split our dataset into four groups depending on the jet number, we obtained different precision depending on the group. In total, our model yielded a precision of (???)% on the training set and resulted on a (???)% precision on the test data according to the Kaggle website. Those results allowed us to each the top (???)% of the leaderboard on Kaggle. While not being as high as we hoped, we found those results to be good enough, especially when considering that we started from 65%.

IV. Discussion

Overall, we are satisfied with our results, although it was exponentially more difficult to increase our score and required a lot of work. There are a few techniques given in the course, such as cross-validation and model selection methods, that we tried to use but to no significant avail. Indeed, we did not find any sign of over-fitting in our system and thus saw no use of cross-validation. [something about model selection] We could not think of much more to do to increase our score on the data exploration side since we already spent a lot of time on it. Furthermore, we experimented in length with our algorithms without seeing any significant differences in the results and inferred that to reach truly higher precision, we would have to design a completely different method and use other, more complex algorithms. We found this consistent with the fact that we are less than 2% away from the best results on Kaggle, showing that no other group could achieve something really better than our model. It is true that those few percents make a big difference, but we would expect more advanced methods (e.g. Neural Networks or Decision Trees) to reach at least above the 90% barrier.

Acknowledgements

The author thanks Christian Sigg for his careful reading and helpful suggestions.