Machine Learning - Higgs Boson Project

Lucien Michaël Iseli, Florian Maxime Charles Ravasi and Jules Eliot Gottraux
Master of Data Science, EPFL, Switzerland

I. Introduction

In this project we try to determine if a given event's signature was the result of a Higgs boson (signal) or some other particle (background) thanks to a vector representing the decay signature of an event. The data set is provided by the EPFL on the website AIcrowd and separated into two parts: a test and a training set. Therefore, the goal is to choose an appropriate machine learning model, such as logistic regression, least squares or ridge regression and train it on the training set, in order to ultimately get the best results on the test set.

II. Feature engineering

A. Data pre-processing

When we first loaded the raw data and ran a simple model, whether it be linear regression or logistic regression, we got an accuracy of 66% on training. In fact the data is unbalanced, two thirds of the data points are not Higgs boson and the remaining third are Higgs boson. So it was as good as always saying no. We needed to process the data to improve our results.

To do that we first used visualization tools, as we can gain a lot of insights on the data using visualization. Thus, we plotted the distribution of each feature. To have a better view of how the values are distributed we displayed them for each prediction, picking the same amount of -1 and 1 predictions. Here is a typical example of which we learned a lot:

We noticed on figure 1 that the distribution is very strange and with this scale on the y-axis we can't really see exactly what's going on. But using this information and by going through the data there are features with many values set to exactly -999. Seems weird to have most of the values around 0 and a lot of them with exactly -999, we concluded that these values are in fact unknown.

Knowing that, we decided to normalize the data without taking into account the -999 values and then setting the -999 values to 0. That way, they shouldn't have much impact on the decision, since $0 \times w$ will give 0, thus not contribute. Normalization is a good tool to avoid ill-conditioning and to balance the weight of each feature. You can see the plot at figure 2.

Another interesting thing we noticed, is that the 22nd feature is discrete. It only has 4 values: $0, 1, 2, 3$, we thought

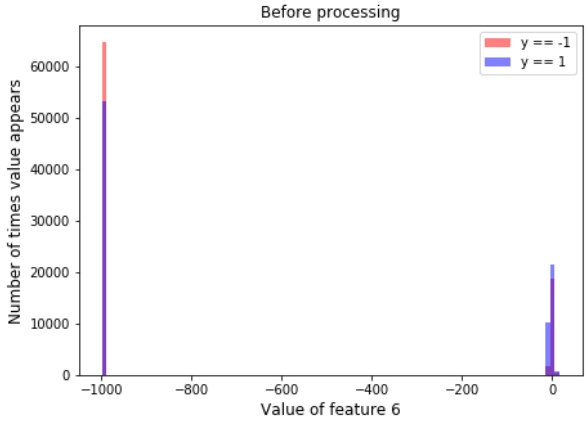

Figure 1. The feature 06 before processing

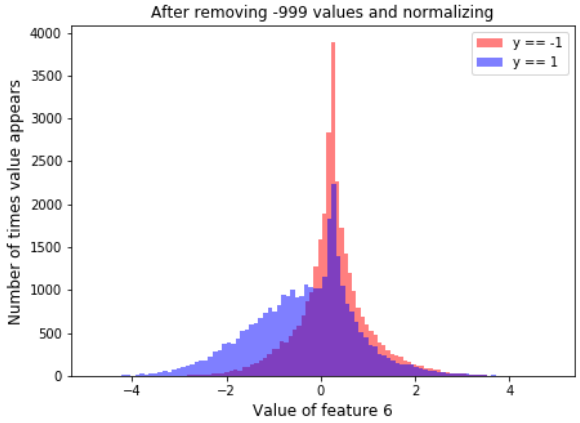

Figure 2. The feature 06 after processing, not displaying the -999 values

that maybe this feature represents some category. Maybe they represent different particles. So we tried to train them differently. By normalizing the data as specified and training these 4 categories differently we managed to get to 78% accuracy on training.

B. Feature expansion

Then we managed to get to 82% accuracy on training and testing by adding feature expansion. To do that we simply tried different basic functions and see which one improved the accuracy the most. We tried:

- Exponential
- Polynomial expansion with and without cross-products
- Cosinus
- Sinus
- Square root

The ones that gave the best results were: cosinus, sinus, degree 4 and square root. Thus, we used these for feature expansion.

III. Model choosing

The logistic regression seemed to be inefficient on the data set since the maximum value we obtained was around 71%. Therefore, we decided to opt for a linear regression model. Since the matrix was not invertible when using the closed form formula from least squares, we chose the ridge regression and tried several values for the regularization parameter lambda. The latter's optimal value ended up being quite small, i.e around 3.6×10^{-7}. In order to compute a good approximation of the lambda we split the train data set that was given and for which we already had the predictions into a new training and a validation set. We then ran ridge regression computing the weights for the training set for each lambda between two bounds and using it on the validation set in order to know what the best lambda is. The figure below illustrates this process 3. Finally, gradient descent did not seem to improve the result, on the contrary we obtained lower accuracies.

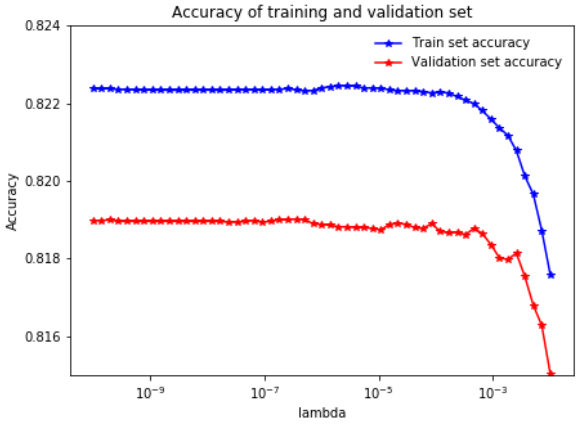

Figure 3. The training and validation set accuracy

IV. Conclusion

In conclusion, we started with a 66% accuracy with no feature processing because of the unbalanced data. Then, we managed to up it to 78% by analyzing data and processing the features, as explained in section II. Then we got up to 82% by adding feature expansion and validation, as described in section II-B. We use linear regression and based on our observations, we concluded that feature engineering is really the key to have a more successful classifier, the algorithm comes next. We unfortunately underestimated the importance of cleaning and processing the data. We lost a bit of time in the beginning, because we were trying to change the algorithm we run instead to have better results instead of changing the input.

V. Further improvements

To further improve the results we could try using other algorithms. We also noticed that some features have more unknown values to known values; some features look like uniform distribution or look like they don't give any information on the classification. We could try removing some of these features to reduce complexity. We could also dive into the actual physics behind the Higgs boson to have a better idea about what the best feature expansion looks like; we could try to add features that are redundant in the physics theory behind Higgs boson. There is always a lot more that can be done when working with machine learning, these are some clues we had while working on this project.