# Group #3. Project 1. Machine Learning course

Team 3 : Joel Castellon, Mathieu Clavel, Marta Kuziora
*IC Faculty, EPFL, Switzerland*

## I. Introduction

This first Machine Learning project was an attempt at tackling a classification problem, namely, the 'Higgs Boson Machine Learning Challenge' [1], mainly implementing and testing the basic methods we studied. The goal is to use Machine Learning to hopefully improve CERN's ATLAS experiment significance, by better discriminating traces of boson decay within the abundant background noise.

## II. The data

Our data consists of energy measurements of particle collisions recorded at CERN's LHC by the ATLAS detector. Data features are either raw collision data (prefixed PRI for primitive) or already processed by CERN physicists (DER for derived).

A serious flaw of this dataset comes from the numerous undefined or meaningless experiment measurements, reflected by an out-of-range value of -999.000 for missing features in the set, as specified in its description.[1]. To compensate for these missing values, we opted for the simple strategy of interpolating by the median (a more robust measure than the mean) : we assigned the median of all non-missing values present in the column to the faulty features.

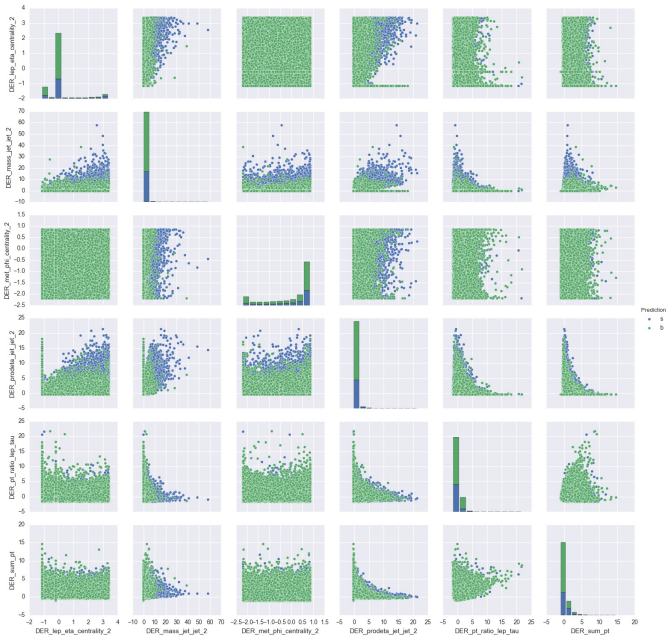

Fig. 1. Pairplot of selected DER features.

### A. Feature selection

The way we proceeded in the exploratory phase of our analysis, was to leverage the work accomplished by CERN physicists, Which is why we mostly pruned the DER features to improve our predictions. The results can be seen in 1. We found that DER_mass_jet_jet and DER_prodeta_jet_jet (2nd and 4th rows in 1) were the features which demonstrated the clearest separation between our signal of interest and the background noise, visualized as the distinction between blue and green points. This is why we decided to make the model more complex with respect to these two features, and therefore added quadratic terms for each. Finally, both DER_mass_jet_jet and DER_prodeta_jet_jet present a bimodal distribution. Adding the quadratic terms allows then a clearer separation on one side.

## III. Models and Methods

We first put in evidence why we choose to work with logistic regression as opposed with the other regression methods. First, we know that a simpler approach like least squares regression is not really made for classification. We fixed $\gamma$ and $\lambda$ to 0.1 for the plots in Fig.2. Such selection is just representative and the results are robust. In fact, many predicted values for least squares regression lie outside the range 0-1 (we used a simple truncation heuristic for mapping them to 0-1). For assesing logistic regression as our preferred choice we plot the precision score ([6] common metric for classification) for held out data in our test set. We can see in Fig. 2 that logistic regression unambiguosly beats the other methods as we vary train-test proportion (sampled at random).

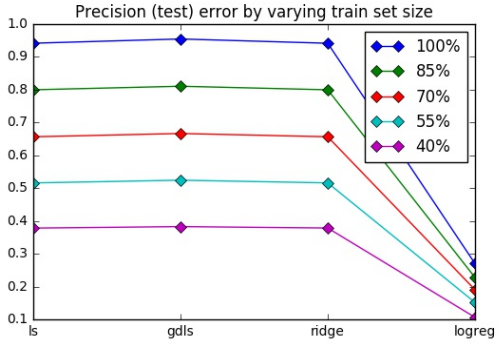


Fig. 2. Empirical comparison of logistic regression against the other methods for this task.

### A. Model Selection

Beyond the methods which we were requested to implement for the project, we ultimately chose **l2-Regularized Logistic Regression** to produce our final prediction, as it demonstrated the best results on the test set.

### B. Algorithms

Because Logistic Regression with l2-penalty is a convex problem, we used Netwon's method. In addition, we implemented k-fold cross validation to select an appropriate $\lambda$ value to control our model's complexity. The metric introduced for selecting the best $\lambda$ was the mean test error (across folds) multiplied by variance, and scaled by fold size. Minimizing such quantity yields the 'smallest' loss, but also accounts for its stability (e.g. variance and sample size). We discuss it further in section IV.
One of the other main point while testing the model fit, was choosing an appropriate step-size parameter($\gamma$), for which we used backtracking line search (Chapter 9 of [2]) to select automatically such ($\gamma$) at each iteration. Backtracking line-search proved to be efficient enough for our purpose.

### C. Implementation

Two main difficulties were encountered while implementing these methods. The first was numerical overflow. This happened often while computing the sigmoid function, as the exp(.) would be applied to very large numbers during Newton's iterations. The solution we found was the simple trick of checking input sign and magnitude beforehand, yielding a numerically stable sigmoid function.[3]
The second challenge was the multiplication of large matrices. For example, multiplying large diagonal matrices for Hessian computations in Newton's method, if done naively, frequently results in memory overflows for datasets as large as ours already. We could in fact avoid the costly computing and storing for some of such matrices by using broadcasting operations available in numpy[4].

## IV. Results

In this section we compare our feature selection process (Selected_Poly in the following figures) against the raw data. These data points were obtained from successive cross-validation experiments in a 10 k-fold setting and a fixed $\lambda$ range.

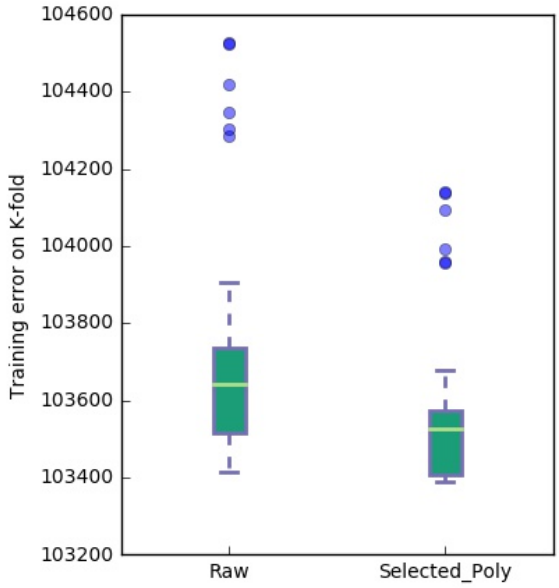


Fig. 3. Box plot for train error when we vary the feature selection.

We can observe with 3 that not only did our data selection improve the training error, as compared to the raw data, but it also made it more stable, reducing its variance.
On the other hand, the test error across folds (in 4) seems to be similar in magnitude. However, the variance is lower for our selected features.

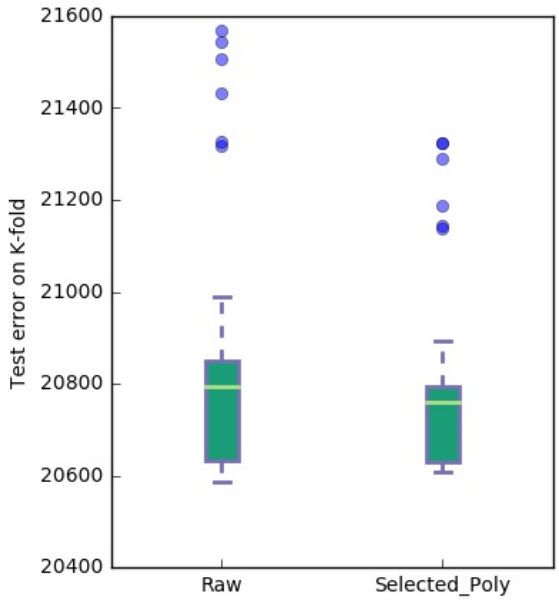


Fig. 4. Box plot for test error when we vary the feature selection..

## References

[1] *Learning to discover: the Higgs boson machine learning challenge*. Claire Adam-Gordarios, Glen Cowan, Cecile Germain, Isabelle Guryon, Balazs Kegs and David Rousseau. 2014. https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf.

[2] *Convex Optimization*. Stephen Boyd and Lieven Vandenberghe. 2004. Cambridge University Press.

[3] *Exp-normalize trick*. Tim Vieira. 2014. Graduate Descent Blog. http://timvieira.github.io/blog/post/2014/02/11/exp-normalize-trick/.

[4] *Broadcasting rules*. Numpy documentation. URL: https://docs.scipy.org/doc/numpy/user/basics.broadcasting.html.

[5] *The Elements of Statistical Learning*. Trevor Hastie, Robert Tibshirani and Jerome Friedman. 2009. Springer.

[6] *Lecture Notes on Generalized Linear Models*. Rodriguez, G. 2007. http://data.princeton.edu/wws509/notes/.