# EPFL Machine Learning CS-433 - Project 1

Jérémy Baffou
*SC-MA1*
*jeremy.baffou@epfl.ch*

Bruno Rodriguez
*MA-MA3*
*bruno.rodriguezcarrillo@epfl.ch*

Léon Delachaux
*EL-MA1*
*leon.delachaux@epfl.ch*

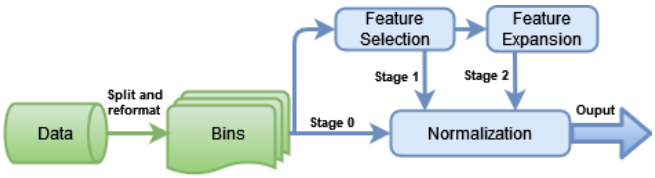
***Abstract*—The Higgs Boson competition aims at developing machine learning pipeline and models to accurately predict the presence of a Higgs boson in the data collected by the LHC from CERN. We developped six different methods based on the course CS-433 of EPFL and developped a small data processing pipeline to achieve this task. By using cross-validation, we suceeded to select a model which achieves an accuracy of 79 per cent.**

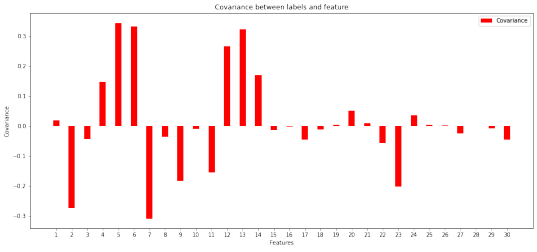
## I. Introduction

We will present you briefly in this report our work for the competition Higgs Boson. We will describe our data processing pipeline, the models we have implemented and the results we had.

## II. Data preprocessing

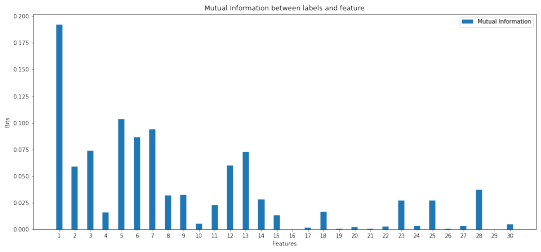
Working on raw data can lead to multiple problems, from overfitting due to outliers to non convergence. Thus we need to understand our data in order to fit the models correctly. We performed an exploratory data analysis of our data set and faced a first problem. The -999 value is present in the data set in multiple features and data points, but it is a proxy to say that the data is undefined. By studying more these values we saw that, except for the first feature, they were splitting the dataset in multiple parts. This is on this idea that we based our data processing pipeline that is described in the figure . As the number of jets is determinant in which features are defined, then there is probably a proper structure for every possible value of the "PRI-jet-num" feature. Before spliting the data set, we first proceeded to mean imputation on the first feature, because we don't want to handle meaningless values of -999. Which means that we replace every undefined data points -999 by the mean of the defined ones. Then we split our data in three distinct partitions based on the number of jets ("PRI-jet-num"). One bin for the case with zero jet, one bin for the case with a single jet and one last bin for cases with multiple jets. All the next steps, from feature engineering to model training and label prediction, will be made bin wise, i.e. each bin will be treated independently of each other. For feature engineering, we have three possible stages (0,1 or 2). Each of the stages can be specified in our script run.py, and will produce different results. The first stage (stage 0) will only take the raw data bins and not apply any processing. The stage 1 will first perform feature selection based on Pearson correlation and mutual information. We computed these two quantities between each feature and the labels. The use of mutual information is a nice complement to the Pearson correlation because it allows us to capture some non linear dependencies. It is computed from the frequencies of the data, which is an approximation of the probability distribution. We found some matching results between the two quantities which motivated the selection of some particular features for each data bin. We also paid attention not to keep some too much correlated feature as it would reduce the explanatory power of the model. Finally, stage 2 will perform some feature expansion. It will add some polynomial version of some features and add some joint-products. If a feature's degree improves the accuracy we add it to our feature list. Same thing for the joint-products. We computed every product of two different features and then took the 6 products of pairs that gave us the best improvement in accuracy. The results of any stage are then normalized by the mean and standard deviation of the training set. The data is now ready to be used for model fitting.




## III. Models and Methods

We have our class set $\mathcal{C} = \{-1, 1\}$. We implemented gradient and stochastic gradient descent (GD, SGD, respectively), least squares (LS), logistic regression (LR), regularized logistic regression (R-LR) and Ridge regression (RR).




Since the following idea was implemented for all of our functions, we describe our procedure using the Ridge Regression. After having processed our data set, we vary the number of k-folds used in cross-validation, that is, we first divide the full data set $\mathcal{S}$ into 10 equal subsets and aggregate 9 of them in $\mathcal{S}_{train}$ and the last one will form $\mathcal{S}_{test}$. This corresponds to k-folds = 10. Following this, we compute the optimal weights for different values of the hyper parameters (if any). Such a procedure is repeated for diverse attributions of the 10 subsets of the data. The range of values for $\lambda$, $\gamma$ was determined experimentally by running several tests.

Regarding the gradient descent methods (both for linear and logistic regression), the step size $\alpha_k$ is crucial for the method to not explode due to gigantic values in the gradient.

We measure the accuracy of our models by simply multiplying the optimal weights by each of the test data points, that is $\mathbf{p} = \mathbf{X}_{train}\mathbf{w}_{optimal}$

$$\mathcal{C}_{i,prediction} = \begin{cases} -1 & , p_i < 0 \\ 1 & , p_i \geq 0 \end{cases}$$

where $\mathcal{C}_{i,prediction}$ denotes the predicted class of the $i$-th element of $\mathbf{p}$ and $\mathbf{X}_{train}$ is the matrix containing all test data. Afterwards, we compute an average over all points that are classified correctly as follows:

$$\text{accuracy} = \frac{1}{|\mathcal{S}_{test}|} \sum_{i=0}^{|\mathcal{S}_{test}|} \mathbb{1}\{\mathcal{C}_{i,test} = \mathcal{C}_{i,prediction}\} \cdot 100\%$$

where $\mathcal{C}_{i,test}$ is the true classification of $i$-th data point in the test set. For logistic regression and its regularized version, we take another threshold and labels, $\mathcal{C}_{i,prediction} = 0$ if $p_i < 0.5$; it is simply because logistic regression values lie between 0 and 1.

The submission systems also used a metric called F1 score which is another measure of accuracy based on recall (how much of the data we declared as signal) and precision (ratio of true signal over declared signal among this data).

For Ridge regression we tested how our accuracy function by varying $\lambda$ and number of k-folds in cross validation. Likewise, in gradient methods we tested our accuracy by modifying the step-size $\alpha_k$; the batch size in SGD was set to 1 for all simulations. More specifically, once we select a value for $\alpha_k$, we keep it during a computation of $\mathbf{w}_{optimal}$. This means that $\alpha_k$ is not changed at each iteration $k$ of the same code running. We determine the range of value for $\alpha_k$ as we did for RR.

The table below shows the accuracy and the F1 score for each method and the following one shows the chosen parameters for each method.

| | Stage 0 | Stage 1 | Stage 2 |
|---|---|---|---|
| GD | 0.730 / 0.612 | 0.732 / 0.617 | 0.723 / 0.616 |
| SGD | 0.731 / 0.611 | 0.728 / 0.616 | 0.723 / 0.621 |
| LR | 0.752 / 0.616 | 0.735 / 0.618 | 0.732 / 0.627 |
| R-LR | 0.734 / 0.615 | 0.735 / 0.618 | 0.732 / 0.627 |
| LS | 0.760 / 0.602 | 0.759 / 0.601 | 0.790 / 0.666 |
| RR | 0.760 / 0.602 | 0.759 / 0.601 | 0.753 / 0.572 |

| Parameters / Methods | Lambda | Gamma |
|---|---|---|
| GD | - | 0.0001 |
| SGD | - | 0.0001 |
| LR | - | 0.000001 |
| R-LR | 1e-6 | 0.000001 |
| LS | - | - |
| RR | 1e-8 | - |

## IV. Results and Discussion

The results show that the highest accuracy is obtained when we use Least Squares. Which is quite surprising and shows that in certain situation, even with simple models we are able to make some good results, provided a good data pre-processing.

A quite interesting results is also that for some models such as Gradient Descent or Logistic Regression, the accuracy can reduce as we increase the stages, i.e. do extra data processing. But the F1 score is in general always increasing. Which means that we even though we are miss classifying some data points, we manage to improve the recall and precision trade-off.

This project highlights perfectly the importance of the data. The performance gain is obtained by working better with the data rather than with more complex models. Thus for further improvements we should look into this direction. For example we could have a more careful feature selection based on non-arbitrary threshold and more complex feature dependence analysis. Right now, the mutual information is computed with bins frequencies as proxies for probabilities. This suffers from many things such as the number of bins chosen. Maybe a more robust implementation should be investigated.Another improvement would be to have a finer feature engineering step with some new features based on the physics knowledge. We could probably craft some new features based on some angles or momentum but our limited comprehension of the field did not allow us to achieve such a goal.