# Machine Learning project I

Anna Diena, Max Duparc, Erwan Seradour (Team AEM)

***Abstract*—This report is part of the first project of the EPFL Machine Learning course (CS-433). Its aim is to develop an algorithm able to recognise case of tau decay of a Higgs boson (mean predicted lifetime $1.56 * 10^{-22} s$) in the data of collision at LHC. Measuring the coupling of the Higgs boson to tau particles is important to verify the standard model where it's predicted that the Higgs boson is responsible for the mass of the other elementary particles.**

## I. Introduction

The aim of our project is to analyse particles collision and predict whether an event was Higgs boson or background. To achieve this goal, we apply exploratory data analysis, feature engineering and train several machine learning models seen in class on a real world dataset.

## II. Exploratory data analysis

Our train dataset is composed of 250'000 samples each described by 30 features and associated with a label either s or b, signal or background. To explore our data, we plot distribution of the features.

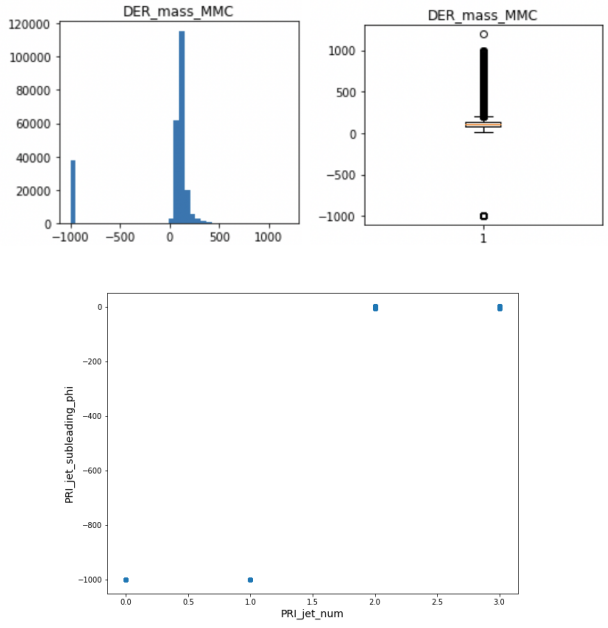


Fig. 1: Examples of a geomtric distribution and and a scatter plot showing the relation between the number of jet and missing values

From those plots, we identify several facts:

1) The $-999$ values (missing values) should be replaced by a more appropriate value.
2) We see 3 main types of data distribution. Uniform, Normal and geometric. It would be interesting to apply logarithmic augmentation for long-tailed distributions.
3) On 10 out of 11 features where there is missing values, the missing values depend from the number of jets.

We replace the missing values ($-999$) by the median of data points, we tried also to replace them with the mean but we didn't see any meaningful difference.

### A. Dependency analysis

For the sake of exhaustivity, we analyse correlation between features , but in the end, we didn't have much us out of it. Indeed, because our data set is somewhat relatively small ($7'500'000$ numbers), we can accept some redundant information, because it will not cause too much useless computing. (We wouldn't have the same discourse if we had Gbytes of data to process).

## III. Models exploration

Before doing any feature augmentation, we apply several models on the dataset to obtain a baseline and compare their effectiveness on this problem.

### A. Model evaluation

We always evaluate a model with the same setup. We randomly split the dataset (`train.cvs`) in two new datasets, train and test, each respectively with $80\%$ and $20\%$ of the samples. Empirically, we estimate that this setup gives us representative results of the true outcome. This is why we don't use cross validation.

### B. methods

In this part, all the models are trained on raw data (only the missing values are replaced by the mean of the features, except when we train logistic regression. In this case, we always normalize the features as it improve convergence very significantly).

| *methods* | *gamma* | *max iters* | *lambda* | *test acc (%)* |
|---|---|---|---|---|
| `least squares` | / | / | 0 | 0.742 |
| `ridge regression` | / | / | 1e-5 | 0.742 |
| `reg logistic regression` | 5e-6 | 500 | 0 | 0.729 |

- A first observation,there is not a major performance difference between models.
- Optimizing models with gradient descent is more difficult and takes more human attention than the closed form solution. Logistic regression takes significantly more time than the other model to converge.
- We achieve best results without regularisation in logistic regression. `least square` is equivalent to `ridge regression`.

## IV. Feature augmentation

We try several different feature augmentations : *polynomial*, sin, cos and log. Each individually improve moderately the accuracy. Now, we try to train the best model using a combination of these features.

### A. Least squares

First, we apply least squares with feature augmentation. We are able to achieve a maximum test accuracy of 0.81%.

We don't overfit, but we identify another problem. We have a big numerical accuracy problem for big order polynomial. The solutions are not stable (ref figure 2 ). To solve this, we explore two options : introducing a regulariser or using regularised logistic regression.

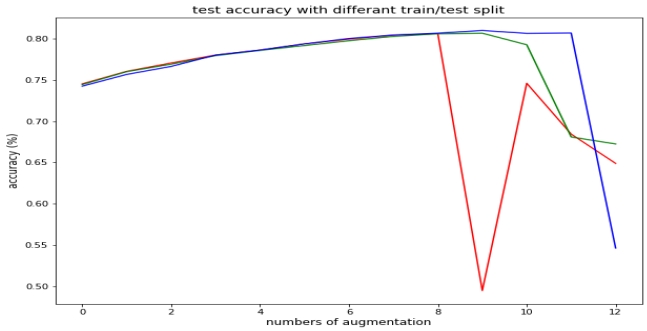


Fig. 2: we train a least square model several time with different train, test dataset

### B. ridge regression

Introducing a regulariser helps to fight ill conditioning. Thus, we may expect better results. We use grid search to find the optimal choice of augmentations and $\lambda$. We find that the best $\lambda$ is $1.29e-4$ and the best set of augmentations is *log, sin, cos and polynomial expansions with degree from* 2 *to* 11. The best accuracy is 0.815.

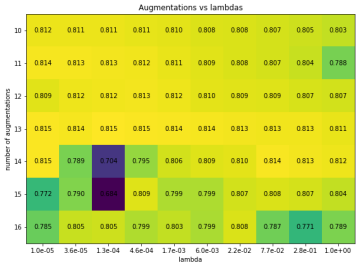


Fig. 3: test accuracy according to the number of augmentations and lambda

### C. regularized logistic regression

We try to find the best set of parameters with logistic regression. To make the different tests, We use a fix number of iterations (300) and normalize the features. We search using two grid search first for the best set of augmentations, then for the best combinations of $\lambda$ and $\gamma$. We find as best accuracy 0.75 (ref fig 4), it is significantly small than 0.815 the best accuracy found using ridge regression. Therefore, we decided to drop this idea.

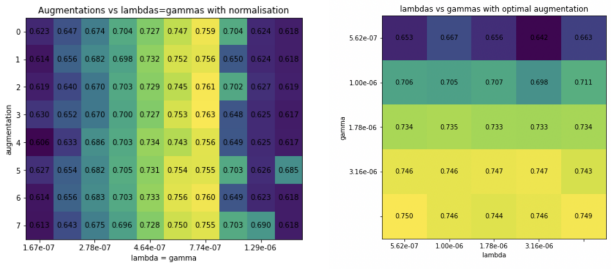


Fig. 4: test and train accuracy according to $\lambda$ and $\gamma$ on `rlr`

## V. Results and discussion

We therefore found the best result with $\lambda = 4.64 * 10^{-3}$ and with *log, sin, cos and polynomial expansions with degree from* 2 *to* 11 as augmentation, giving us, `test accuracy` of 0.815. Thus, we have the following choice

| *name* | *max accuracy* | *computation time* | *big number accuracy* |
|---|---|---|---|
| `ridge` | 0.815 | 5s | No |
| `rlr` | 0.75 | 45s | Yes |

We thus choose `ridge regression`, because its only downsize is *number accuracy* but this is something we can control.

Using this method, we obtain when uploading on AIcrowd an accuracy of 0.82, which is close from the result of our test.

It is important to note that if we used normalisation using `rlr`, we are not using it with `ridge regression`. This is because in this case, this is giving us worst result than without.

## VI. Improvement ideas

If one thing is certain, it is that we were lacking time for this project. Indeed, we could have improved our ML method by:

1) Studying the context of the data more thoughtfully. For instance, in the data there is only one categorical feature,$PRI\_jet\_num$ which represent the number of jets. Since the definition of several features depend on the value of this categorical feature, we could have explored more thoroughly the possibility of using four different model, one for each value of $PRI\_jet\_num$ and eliminate the not defined features. For example, if $PRI\_jet\_num < 1$, then many other values are not defined and thus equal to $-999$.
2) Try to study in far broader details the inner working of `rlr` and to find its shortcoming, to be able to reach 0.8 accuracy with our model.