Machine Learning - Project 1

Montavon Justine, Rizk Kevin, Poulain- -Auzéau Louis
EPFL, Switzerland

I. Introduction

The Higgs Boson is an elementary particle which was discovered at CERN in 2013. The goal of this project is to determine from some data whether a collision generated a Higgs Boson or not. In order to accomplish this goal, we first analyzed our data set. And then we tested our data on many Machine Learning models.

II. Models and methods

A. Analysis and process of the data

The given data is composed of a train set and a test set, containing 30 features each and $250'000$ and $568'238$ observations respectively. In each of the data, a column named PRI_jet_num contains integers from 0 to 3 that indicates for 11 other columns whether or not there were missing values (if the value was 0 or 1) in the data. There are a total of $1'580'052$ and $3'588'434$ missing values respectively, each having been labeled -999.

1) Dealing with missing values:

- First method: To prevent the missing values from interfering in our experiments, we decided to replace each of those -999 by a zero value. Then, in order to see if missing values were affecting the result, we decided to create for each different column that contained missing values a new column containing only 1 and $0:0$ if a value was not missing and 1 otherwise. This was done to tale into account the effect a missing value had on the prediction. Another solution was to replace missing values with the mean of the other values, but the results were far worse so we held onto the first idea. The best accuracy we got with this method is an accuracy of 0.820 using ridge regression.
- Second method: To make more careful calculations and take more advantage of the missing values, we finally divided the data set into 4 groups, based on the value of PRI_jet_num. For each group removed the columns where there was missing values. Then, we applied our polynomial basis to each of the groups, separately, in order to get the best results for each group before reassembling. As this gave much better results, we kept this model as our final one. This gave to us our best accuracy of 0.830.

2) Data cleaning: Since the data seemed unbalanced and have long tails, we tried different methods to reduce its unbalance.

- Balancing the data: They were more data with values -1 than 1. The balancing was done by selecting a random subset of the data with -1 values, so we have an equal number of data with -1 and 1. But this method reduced the size of the train set and it didn't improve our results, hence we did not end up using this method.
- Divide according to the mean: The goal of this function was to eliminate outliers from the data. To do this, we created two groups: one that contained the rows that were too far from the mean and the other that contained the rest. Then, we performed our regression on the latter. That had the consequence to reduce too much the size of the set on which we trained and so it did not improve our model.
- Tail transformation: To reduce the weight of data with long tails, we decided to apply a logarithmic transformation $f(x) = \log(x+1)$ to the columns that had a standard deviation greater than 10. This, coupled with our other transformations (see paragraph II-A4) of the data, gave us better results so we kept it into our final model.
  Figure 1 shows firstly the presence of long tails far away of the mean and how making the log transformation reduced the unbalance of the set.
- Standardizing: We tried to standardize the features to give equal weight to all features and this helped us improve the results. So we kept it into our final model. In the end our best results were found when we standardized after applying the log transformation (to keep values positive).

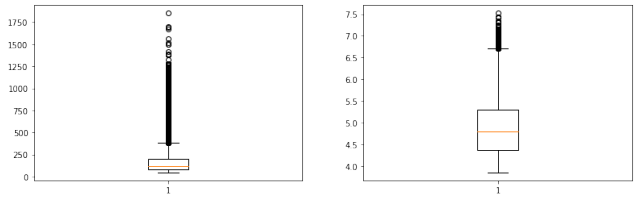

Figure 1: Box plot of a column of our training set before and after the $\log(x+1)$ transformation.

3) Feature selection: We used mostly two methods for feature selection:

- Correlation: We computed the correlation matrix associated with the data set in order to remove the features that were highly correlated (correlation outside $[-0.9, 0.9]$). It turned out that the new model was not giving better results, so we did not remove features using this method.
- Backward elimination: Noticing that after performing ridge regression, some coordinates of our minimizer w were close to 0, we tried to remove the columns associated and to run again with the new model obtained. Again, we did not obtain better results.

As described in paragraph II-A1, we ended up removing the columns corresponding to missing values in our four groups.

4) Features augmentation: We mostly used a model based on a polynomial basis with specific degrees that were found to perform better after running several tests and cross validation. More specifically, we constructed a new model containing the initial matrix multiplied element wise by itself, up to the degree we entered and then we added the 3 columns corresponding to missing values:

$$\Phi(\text{Data}, \text{degree} = d) = \begin{pmatrix} \mathbb{1} & \text{Data} & \dots & \text{Data}^d & \text{Missing} \end{pmatrix}$$

We tried to add features of order 2, which means introducing a new column obtained by multiplying two columns of the initial data (after removing the -999), but the results were not outperforming our previous results and the simulation was too long to determine which columns to multiply or to perform every possible multiplication.

Since the results were not satisfactory, we tried to standardize the matrix but again it was not improving the results, so we dropped out this possibility.

B. Loss function for least squares and ridge regression

When we started the project, we noticed that the MSE/RMSE error for all the methods using least squares was not the most suited loss function for our classification task. Indeed we were more concerned to know how many data points predicted were false. So we created a function which computes the accuracy of our results by calculating the ratio of falsely predicted labels. We used this function to have a better understanding about how each models behaves and which ones of our changes or methods was performing better or which hyper-parameter was the most suitable.

C. Our implementations

We implemented the six used ML methods with different processed data. We observed several things.

- Least square: First, we found that the results of least squares and ridge regression using normal equations were more convenient than least square using gradient or stochastic gradient as by using normal equations we directly go to the optimum. So we did most of our testing with ridge regression and least squares using normal equations and we experimented a lot by splitting the data or choosing different degrees to expand our features or changing the lambda for the ridge regression. This helped us find a good weight giving us an accuracy of about 0.830. Figure 2 shows an example where we found of about 0.820 on AIcrowd. We split the training data into a training and test set and we did a grid search on the degrees and lambdas using ridge regression, and we plot the smallest error for each degree.
- Logistic regression: As this is a classification problem, logistic regression or regularized logistic regression is normally a more suitable task, but there is no closed form solution for the optimum of the loss. Hence we decided to use gradient descent to try to find the minimum, and as the loss function is convex, and so the optimum is unique, for a good choice of gamma we should converge to this global minimum. We did a lot of experiments by trying different lambdas, and different degrees, we had the best accuracy close to 0.805 which is still good. But it was worse than using ridge regression or least square.

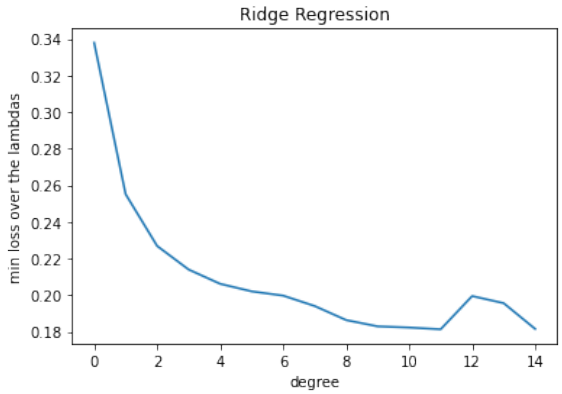

Figure 2: Example of one of the plots of the error for different degrees using Ridge Regression

III. Results

The process of the best result we obtained is the following. First we split the data into a training and testing set, then we divided the data into 4 groups, each of them corresponding to a particular value of PRI_jet_num. In order to be able to assemble again, we keep in memory the indexes. Next, we "clean" the data by applying a logarithmic transformation to the columns that have a long tail and then standardizing. Then, for each group, we perform a ridge regression. To do this, we use a grid search as cross validation to find the optimal degree and λ using the error on the testing set. The data is modified, for a given degree, according to the procedure described in paragraph II-A4. When the regression is over, we make a prediction for each group. Then we re-assemble the predictions into one prediction.

For the groups $(0, 1, 2, 3)$, we obtained as hyper-parameters $\Lambda = (1.46 \cdot 10^{-4}, 2.56 \cdot 10^{-12}, 1.33 \cdot 10^{-6}, 5.18 \cdot 10^{-7})$ and $d = (13, 16, 14, 11)$.

We performed an accuracy of 0.83 with $F1$-score of 0.744 on AIcrowd.

IV. Conclusion

Although we were expecting better results using a logistic regression for this classification problem, we got the best results using a ridge regression method. One of the reason was the choice of gamma. Indeed, it could span from very low values to much higher ones. One idea to get better results could be to implement a line search at every step step of the gradient descent in order to get the best γ. But this is a costly technique. An other idea would be to perform a more careful analysis of each group, after having divided the data set in order to adapt a method for each group. For instance, the group corresponding the value 1 gave us far worse errors than the others.