Project 1 report: Higgs Boson Challenge

Brioschi Riccardo, D'Angeli Gabriele, Di Gennaro Federico
Department of Computer Science, EPFL, Switzerland

***Abstract*—In this report we will show how we worked on Higgs Boson Challenge, a classification problem proposed as project 1 for ML course at EPFL. We implemented our solution from scratch by only using Python's *Numpy* library.**

I. Introduction

The aim of the challenge was to predict whether a collision event can generate an Higgs Boson or not. Given the fact that scientists can not really observe this elementary particle, they discovered it by looking at its decay process through different measurements. The goal of the analysis was to find a region in the feature space in which there is a significant excess of events called signal ('s'). In Section II we describe how we processed these decay signatures, why we removed some of them and how and why we built new features starting from existing ones. Furthermore we present our models, whose parameters were chosen after a Cross-Validation/Grid-Search. Among these models, we decided to keep the best one according to the accuracy scores we obtained (r.f. Table I). In Section III we report the results and we comment them in Section IV.

II. Model and Methods

Firstly, we decided to process the data as well as we could. We truly believe that an effective feature engineering process and a proper way to handle missing values and outliers are extremely important to speed up ML algorithms and to get an high accuracy. We will now show the steps we did in order to clean data and get the models we decided to train.

A. Preprocessing

The training set contains 250000 observations with 30 features; the output column is split into 'signal' (1) and 'background' (-1). The test consists of 568238 data points with the same number of features, and we were asked to predict -1 or 1 for these data.

1) *Categorical Feature*:
We observed that one of features (*PRI Jet num*) was a categorical variable. Since, for every category, the same features may have a different relationship with the outcome, we decided to split the dataset into three subsets with respect to the value of *PRI Jet num* (0 for the first subset, 1 for the second subset and 2 or 3 for the third subset).

2) *Handle missing values*:
The dataset was full of missing values. In every subset we deleted features for which the number of missing values (labeled as -999) was larger than 50% of total number of observations. For features having a percentage of missing values in $(0, 50)$ we decided to impute missing values with the median of the column, since it is a robust estimator.

3) *Transformation of "angle features"*:
We also observed from the documentation that some of the features were angles. A reasonable choice to handle these features was to add the sin and the cos values of each angle to our features set. We later decided to drop columns containing angles since the value of the angle itself seemed to be quite useless.

4) *Log-Transform*:
After dealing with missing values, we plotted the distribution of features remained in each subset. We noticed that most of them were extremely skewed (example in Fig 1) so we decided to transform these skewed features by using a logarithmic transformation ($x_{new} = log(1 + x_{old})$).

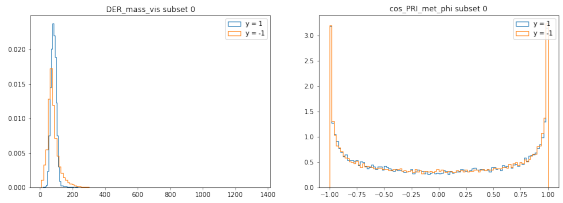

Fig. 1: *On the left a skewed distribution for subset 0, on the right a not skewed distribution.*

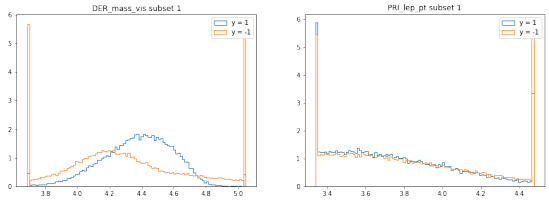

Fig. 2: *On the left a useful feature for subset 1, on the right a useless one.*

5) *Capping Outliers*:
Before doing features selection we decided to cap outliers in every remaining column by replacing them with $percentile_{0.95}$ if value $>$ $percentile_{0.95}$ or with $percentile_{0.05}$ if value $<$ $percentile_{0.05}$.

6) *Dropping useless features*:
In order to decide whether a feature was useful or not for a subset, we plotted for that subset the distribution of the given feature dividing the observations with respect to the value of the corresponding outcome ($y_i = 1$ or $y_i = -1$). Therefore, we obtained two different distributions for each analysed feature. We then compared the shape of the two distributions and decided to drop all the features for which

the two curves were very similar. In Fig. 2 we reported an example of this procedure.

7) *Standardization*:
A good practice in ML is to standardize the dataset in order to reduce overflows and ill-conditioning of the matrices. Therefore we decided to standardize our dataset before training our models.

8) *Polynomial expansion*:
As a last step, in each subset we added a column of ones (offset) to the feature matrix. We then computed a polynomial expansion up to the optimal degree chosen through Cross-Validation and Grid Search. In the end, we added the product of each pair of features in order to include interaction factors in our model.

B. Evaluation of methods

Once we finished to process the training dataset, we started evaluating some of the models we have been studying during the course. We focused on linear models and logistic regression (generalized linear model). We also added a regularization term λ where possible to prevent overfitting (since the number of remaining features was still large). The optimal value for λ was chosen using Cross Validation. We decided to not try the Least Squares with GD and SGD because the computational complexity was not that high to suggest a stochastic version of the algorithm. For the same reason, we also rejected the possibility to train the logistic regression model using SGD.
After obtaining accuracy values for every model, we chose to keep the best one according to F1-Score and accuracy. In Table I you can see the accuracy level for all models (each one trained using the best hyperparameter values, computed with a 4-fold Cross-Validation on training data). Notice that we tried to minimize the variance of the accuracy to ensure that our model could generalize well and was not subject to overfitting.

III. Results

A. Choice of hyperparameters

We used Cross-Validation and Grid Search to find the best hyperparameters (degree and λ) for each model. Note that, since each subset was different, we had to find three pairs of hyperparameters and consequently three vectors of model parameters w.

B. Analysis of the methods

As shown in Table I, we obtained the highest accuracy values using Ridge Regression and Least Squares. In both cases, we considered $degree = 7$ for each subset but different lambdas. The suspect although was that, when using Least Squares, we could suffer from overfitting and ill-conditioning problems on test data. Therefore, we decided to keep Ridge Regression as our best model.

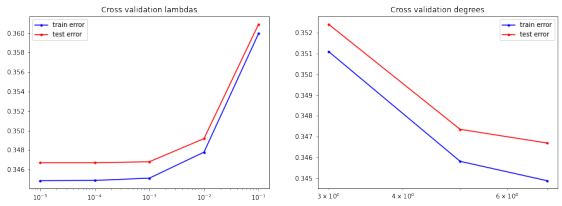

Fig. 3: *Cross-Validation on λ on the left ($degree$ fixed), on degree on the right (λ fixed). By observing the similarity between the curves and the small error values, it was clear that the chosen model did not suffer from overfitting.*

Method	Train accuracy	Test accuracy	std test
Least Squares (Normal Eq.)	0.8357	0.8350	0.0009
Ridge Regression	0.8356	0.8351	0.0009
Logistic Regression GD	0.8320	0.8319	0.0010
Reg. Logistic Regression GD	0.8320	0.8319	0.0010

Table I: *Accuracy obtained with 4-fold Cross-Validation to generalize the results. Accuracies in the Table were calculated with the best hyperparameters.*

IV. Discussion

Ridge Regression proved to be the best model for the challenge on AIcrowd. This model has also the strength of having an analytical solution

$$w_{opt} = (X^T X + \lambda' I)^{-1} X^T y$$

where $\lambda' = 2\lambda N$.
This solution can be easily computed and the regularization term allowed us to reduce and prevent overfitting and ill-conditioning problems.
We believe that the preprocessing we made has been really useful to increase our scores. Moreover, we are also sure that dividing the dataset according to the value of the categorical feature *PRI Jet num* was a key factor in improving the accuracy of the final result. Another idea that could have improved the accuracy of the model was to add square roots values for each feature (e.g $\sqrt[2]{\cdot}$, $\sqrt[3]{\cdot}$, ...) during polynomial expansion, but we decided not to increase the computational complexity of our model.
An accuracy of 0.836 and an F1-Score of 0.749 allowed us to be in the top 10% of teams in the public ranking of AIcrowd platform.

V. Conclusions

In this challenge we understood how to deal with data in order to complete a given task with good results. We also realized the importance of preprocessing on raw data: we observed on our scores how this can affect the accuracy of a ML model. We tried to be as meticulous as we could in order to interpret features in a useful way and we can also conclude that with a good step by step work every standard classification algorithm can perform well on this task.