Project report: Higgs Boson Challenge

Giovanni La Cagnina, Francesco Pettenon, Nicolo' Taroni
Department of Computer Science, EPFL, Switzerland

***Abstract*—This report aims to present the methodologies used for the Higgs boson Machine Learning Challenge. The target of the Challenge is to detect the presence of Higgs boson based on several Physics variables provided by CERN.**

I. Introduction

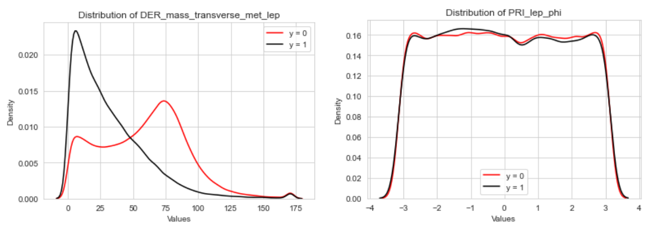
The Higgs boson Challenge tries to reproduce the process of discovering the Higgs particle exploring the potential of Machine Learning methods. The presence of the Higgs particles can be estimated only by its decay signature and by its consequences on other particles. In our analysis we exploit the measurement of decay signature to construct a model that predicts the boson presence. In order to do so, first we perform a preliminary analysis of the features, then we apply some transformation and finally we implement and compare different models. In section 2 we describe the different methodologies used to approach the Challenge. Indeed, we present the results of the Preliminary Data Analysis, the transformation applied to the features and a description of the model applied and the testing procedure. In section 3 we report and compare the results obtained with different models and transformations of the initial data, exposing the methods used for the hyperparameters tuning. Finally, in Section 4 we discuss the results obtained and how our choices have improved the accuracy of the model.

II. Methodologies

In this section we report the preliminary analysis, the feature engineering and the model design used to solve the classification problem.

A. *Preliminary Data Analysis*

- **Missing values**: The Challenge involves two dataset: a training set composed by 250000 sample points with 30 features and a test set consisting of 568238 observations. In the training set along with the features we observe a categorical variable that describes the presence of Higgs boson (s for signal, b for background). A detailed features description is reported in [1]. Through a first exploratory analysis, we detect the presence of several missing values. Based on the description of the features and on the data provided, we notice that missing values are concentrated in a few variables. This is due to the fact that some variables were recorded only when `PRI_jet_num` (the number of jet) measure is greater than 0 or 1. Therefore, we split the dataset into 4 subsets according to the jet number. This choice is similar to consider `PRI_jet_num` as categorical and allows us to use all the features for our analysis. For the remaining missing values, observed only in `DER_mass_MMC`, we replace the values with the median, a robust estimator for outliers.
- **Distribution of features and outliers**: To have a better understanding of the variables that are present in the dataset, we plot the empirical distribution for each feature given the boson classification of the train set. For some features we observe no differences between the two distributions (1). Therefore, since these features seem to have no particular effect on classification we decide to build our models without them. We observe also the presence of long tails in some distributions, possibly caused by outliers. Therefore we cap the extreme values of the features using a quantile chosen using cross validation. We also check the covariance matrix, without observing particularly highly correlated features.



(a) Example of distinct distributions (b) Example of similar distributions

Fig. 1: Comparison of feature distributions with respect to the boson classification

- **Data Transformation**: Given the observation made at the previous point, we apply the following transformation to each variable: $\log(x + 1 - \min(x))$. This transformation helps in dealing with long tails and also in introducing some non-linearity in the model.
- **Cross product and basis function**: In order to expand our data, we augment the dataset adding the cross product feature between all the variables. On top of that, we use polynomial basis function with the degree chosen in cross validation, adding also the offset term.
- **Standardization**: We observe that each variable has a different mean and variance. Then, in order to avoid ill conditioned data matrix that could lead to a bad gradient descent optimization and to instability in solving linear systems, we decide to standardize each feature.

B. *Model implementation*

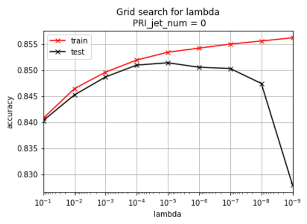
We consider the following models: Least Squares, Linear Regression with Gradient Descent, Linear Regression with Stochastic Gradient Descent, Ridge Regression, Logistic Regression with Gradient Descent, and Penalized Ridge Logistic Regression with Gradient Descent. To measure and compare performances of different models, we use the train accuracy and the test accuracy. This metric represents the percentage of correct predictions on the total number of observations. For each model considered, and for each jet subset, we fit the model and tune the hyperparameters using 4-fold cross validation and grid search. Therefore the result of the training phase, for each model, consists of four weights vectors and four set of hyperparameters. Similarly, the performance of each model is obtained as a weighted average of accuracies on the four subsets.

C. *Tuning of Hyperparameters*

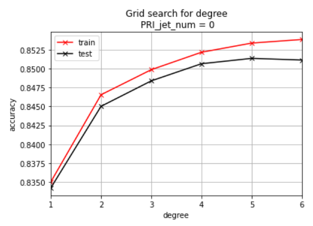
To tune the hyperparameters, we utilize the grid search algorithm, using the 4-fold cross validation performance to evaluate each point on the grid. The hyperparameters to be tuned are: the α-quantile used to cap outliers, the degree of the polynomial expansion, and, for models which include a penalization term, λ. For models allowing closed form solution we use a denser grid.

PRI_jet_num	λ	$degree$	$quantile$
0	1e-05	5	0.9250
1	1e-05	5	0.9125
2	1e-04	5	0.9500
3	1e-04	5	0.9250

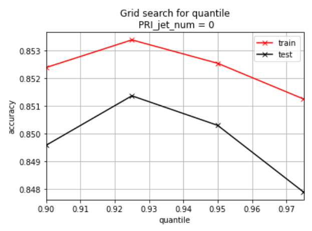
TABLE I: Hyperparameters computed for each PRI_jet_num



(a) Tuning for λ hyperparameter



(b) Tuning for $degree$ hyperparameter



(c) Tuning for $quantile$ hyperparameter

Fig. 2: Grid search for hyperparameters

III. Results

The performance of the different models considered are reported in Table II. It is possible to observe that the model that performs better is the Ridge Regression. Using this model we obtain an accuracy of 0.837 on the test set of AI-Crowd. This result is in line with the test accuracy achieved in cross validation and it leads us to think that our model is not overfitting the train set. As we can see from Table II the other models have lower test and train accuracy. This could be due to two main reasons: the closed form solution of the Ridge Regression, which allows us to train the model on a denser grid, and the penalization term, that permits to avoid overfitting while increasing the polynomial degree. To confirm this, we note that the second and third best models are Least Squares and Penalized Logistic Regression: the former does not include any penalization term while the latter, given the fact that it uses a Gradient Descent as optimization algorithm, has more difficulties in convergence. We believe that with more computational power, that allows to tune parameters with a denser grid, the Penalized Logistic Regression could reach at least the same performance as the Ridge Regression and, given its characteristic of being more robust with respect to outliers, could provide a better performance.

Model	Train Accuracy	Test Accuracy
Least Squares	0.83613	0.82617
Linear Regression (GD)	0.78122	0.78110
Linear Regression (SGD)	0.79806	0.79703
Ridge Regression	0.84563	0.83677
Logistic Regression (GD)	0.77787	0.77593
Penalized Logistic Regression (GD)	0.83166	0.82847

TABLE II: Model performances on Training and Test set computed using 4-fold cross-validation

IV. Discussion and Conclusion

As we have discussed in the `Results` section, since we increase significantly the complexity of the model, the best accuracy is obtained with the Ridge Regression in cross validation. Indeed, the regularization parameter is crucial to avoid overfitting. Additionally, the closed-form solution of the Ridge Regression problem allows to obtain a trained model really quickly and to significantly limit the computational costs. This allows us to test several different hyperparameters and, especially for the Ridge Regression, to reach a very high accuracy avoiding overfitting despite the complexity of the model. This work shows how, if developed in a consistent way, even a very simple model can perform very well in a real problem with a huge amount of data. Stated this, we also believe that, with more computational power, Penalized Logistic Regression has space for improvements and could lead to better results.

References

[1] I. Guyond, B. Kégla, and D. Rousseaua, “Claire adam-bourdariosa, glen cowanb, cécile germainc.”