Discovering the Higgs Particle

Sinsoillier Mike Junior, Du Couédic De Kergoualer Sophie Zhuo Ran, Berezantev Mihaela
Department of Computer Science, EPFL, Switzerland

***Abstract*—This paper presents the application of several simple and well-known machine learning techniques on a real-case classification problem, namely predicting whether a given event was the result of a Higgs boson or not. The aim is of this paper is to show the complete procedure needed to obtain best possible results from very simple models. This includes data pre-processing, hyperparameters tuning, and models evaluation.**

I. Introduction

When colliding protons at high speed, Higgs boson particles may be produced. The accelerator data provided by CERN contains event signatures and the corresponding results which may be a Higgs boson or some other particle. Our goal is therefore to design a model whose for a given event's signature, will predict if it is the result of a Higgs boson or not. In this paper, we will first state the preprocessing steps, then show the different models we tried to fit on the data set and finally the evaluation of the models.

II. Models and Methods

A. Exploratory Data Analysis

To find what types of pre-processing may be useful before applying a model, we first explored the data. The available data comprises a training set with 250'000 events and a testing set with 568'238 events. Each event is described by a binary output variable and 30 features. We also noted the presence of `-999` values, representing missing or undefined values. Finally, by plotting the distribution of each feature, we observed that some of them are heavy-tailed and only one is categorical.

B. Feature Processing

Based on our exploratory data analysis, we performed some feature engineering in order to obtain better accuracy in our results.

1) Dealing with Undefined values

The data set contains missing values (expressed by `-999`) which can greatly influence the results if not handled properly. We had several choices on how to deal with those values, including dropping features having too much of them (e.g. more than half of the values are missing), or replacing them by the mean or median. The complete removal of a feature led to worse results, so we decided to simply replace the `-999` by the median of the corresponding feature. This makes more sense than the mean, as the data set contains a lot of skewed data.

2) Reduce skewness

By analysing the distribution of the features, we noticed that some of them are skewed. Skewness may bias the prediction by for example giving more importance to some values than the others, therefore we applied a log transformation in order to reduce it.

*3) Drop *-phi features*

In the feature distributions we also noted that several features had a approximately uniform distribution. They all turned out to be *-Phi features and we decided to drop them because they do not have a significant impact on the predictions result.

4) Standardization

The data set is standardized as this improves its stability.

5) Introduction of Bias

A column of ones is added to the feature matrix in order to account for the bias term.

C. Feature vectors extension

The linear models can typically be too simple to fit correctly the data. To overcome this problem, we augmented the input using polynomial degree expansion. The most appropriate polynomial degree is not necessarily the same for two different models, therefore we tried different degrees for each model.

D. Machine Learning Models

After the data was processed, we focused on designing the different models for predictions. We implemented the following ones:

- Gradient descent
- Stochastic gradient descent
- Least squares
- Ridge regression
- Logistic regression (uses gradient descent)
- Regularized logistic regression (uses gradient descent)

E. Hyperparameters Selection

The hyperparameters include:

degree	the polynomial degree expansion
max iterations	maximum number of iterations for gradient descent or stochastic gradient descent
γ	step size for gradient descent or stochastic gradient descent
λ	regularization term

Apart from ridge regression, the hyperparameters of each model were chosen by successive trials of different combinations. This approach may miss some optimal parameters, but saves significant computation time. However, to have a precise idea of what is the best performance a model can achieve, we picked one of the models, **ridge regression**, and optimized it using 4-fold cross-validation to have the best possible parameters.

All the hyperparameters are reported in Table I.

Method	degree	γ	max iterations	λ
GD	2	1e-2	100	-
SGD	2	1e-7	2000	-
Least Squares	1	-	-	-
Ridge	9	-	-	4e-4
Logistic	3	1.5e-7	800	-
Reg Logistic	3	1e-7	800	1e-3

TABLE I
HYPERPARAMETERS

F. accuracy measurement

To have a good estimation of how well each model performs, we used 4-fold cross-validation to produce the mean and standard deviation of the accuracy of the predictions of each model. Note that the comparison of the models is done on an accuracy basis, and not error basis because the error is not evaluated the same way for all the models.

III. Results

The results are reported in Table II and plotted in Figure 2 for a better interpretation. All models except SGD shows a high confidence for their accuracy. The high standard deviation of SGD is not a surprise since the algorithm samples single data points to compute the step's directions, and those samplings can highly vary the produced weights during the cross-validation.

Ridge regression produces the best accuracy, and obtained an accuracy of 0.819 on AIcrowd. It is followed closely by logistic regression and regularized logistic regression. It is interesting to notice that the degree of expansion goes as high as 9 before overfitting the data. Figure 1 shows the RMSE of ridge regression per degree, using for each degree the best lambda as defined by the cross-validation. We can see that the model overfits the train set as soon as it reaches degree 9.

Method	accuracy mean	accuracy std
GD	0.77	2e-4
SGD	0.68	9e-3
Least Squares	0.74	2e-4
Ridge	0.81	2e-4
Logistic	0.80	3e-4
Reg Logistic	0.80	7e-4

TABLE II
MODELS ACCURACY

IV. Discussion

The ridge regression produces the best results, this is probably thanks to the tuning of its hyperparameters using cross-validation, which tests all possible combinations of degrees and regularization term. Theoretically, the model supposing to produce the most accurate predictions is the regularized logistic regression, which is particularly appropriate for classification task as it is the case here. However, this method uses gradient descent and would require too much computational if we want to test all combinations of hyperparameters. Therefore, ridge regression represents a good tradeoff.

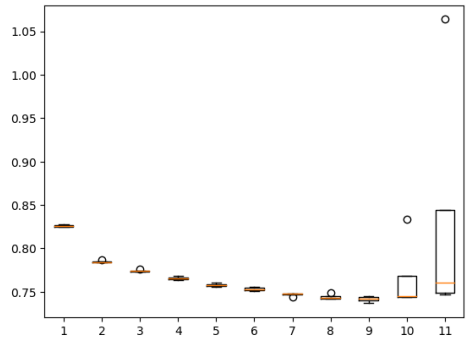

Fig. 1. RMSE per degree for ridge regression

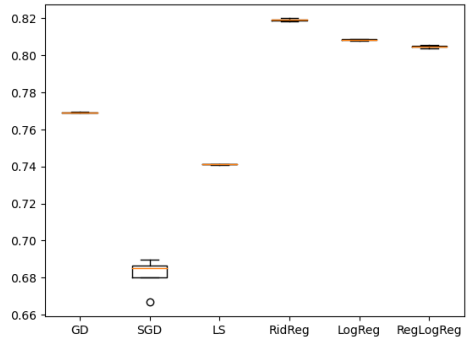

Fig. 2. means and standard deviations of the accuracy per model

Some more approaches could have been taken in order to improve our results and the relevance of our comparisons, including using MAE loss, choosing best hyperparameters by using cross-validation on all the models, or using SGD with larger batch size. Concerning the data, some more pre-processing could have been done, including handling of outliers or splitting the data on the categories to train the models for each feature.

V. Conclusion

To conclude, it is not difficult to obtain a *not-too-bad* model. Our work highlights the additional work needed to improve the models and to obtain a better accuracy and stability. In this intention, data pre-processing, fine-tuning of hyperparameters, and finely tracking the error and accuracy of the models all play a crucial role.