Higgs Boson Machine Learning Competition - EPFML Project1 2019

Maxence Jouve, Rayane Laraki, Pierre Schutz
EPF Lausanne, Switzerland

***Abstract*—The discovery of the Higgs boson announced July 4, 2012, led to many prestigious awards, including a Nobel prize. ATLAS, a particle physics experiment taking place at CERN, observed a signal of the Higgs boson decaying into two tau particles. Nevertheless, the signal is small and in a noisy background. This report details our work on the Higgs boson Machine Learning Challenge, where the goal is to use machine learning methods to improve the discovery significance of the ATLAS experiment. Using simulated data characterizing events detected by ATLAS, the task was to classify events into "tau tau decay of a Higgs boson" vs "background".**

I. Introduction

Hosted on the competitive platform AIcrowd, this project aims to introduce us to applied Machine Learning tasks. Competitors are evaluated using the accuracy and the F1-score of their predictions.

The training dataset contains 250,000 examples that list 30 values calculated from a simulated ATLAS experiment (more details about the features can be found $here$. The result we need to obtain is a prediction (1) or (-1) if we detect the decay of a Higgs boson (or not). Some features values can be missing or impossible to compute and have been set to -999.

In the report, we will first explore the available models, and discuss data processing, features engineering, and parameters selection. Finally, we will present the final model and our results.

II. Model exploration

A. Initial Data Processing

First of all, before selecting the best model for our problem, we first decided to replace the missing values (-999) as well as some outliers by the median of the given feature. This will help us avoiding over-fitting the data. We chose to set the outliers at $mean \pm 3\ std$ to replace only a few ones that are very far from the distribution. We then standardize each feature. This first processing will help us testing our different models with a better dataset, and avoid spoiling our results because of outliers. We can now test the different ML techniques on our dataset to see which one performs the best. We will come back later on the data processing task to improve our performances.

B. Models available

For this task, the different models available for use are:

- Linear regression using GD
- Linear regression using SGD
- Least squares using normal equations
- Ridge regression using normal equations
- Logistic regression using gradient descent or SGD
- Regularized logistic regression using GD or SGD

We chose to implement the logistic regression and regularized logistic regression using SGD because it gave us better results.

C. Model evaluation

To select the best model between the ones presented before, we discriminate them on their performances, using the measure of accuracy on a testing set. To do so, we first split our train set into two data set with a ratio of 0.8 (80% train and 20% validation). We then perform the initial data processing on the train and test set independently. Then for the models that have a hyperparameter, we use cross-validation to determine it. Otherwise, we directly train our model on our data and compute the $loss$ and the $weights$ vector. We finally compute the test accuracy with our processed test set and the $weights$ vector which finally gives us the accuracy.

The result of this model evaluation is presented in table I. We can see that logistic regression has better performances in general than the linear models. This is not a surprise as we are facing a classification problem. Therefore, we selected the regularized logistic regression model. After working with logistic regression, we saw that our performances were not as good as expected. Also, the training time with the hyperparameters selection ($lambda$, and polynomial features $degree$) was very long. Finally, we choose to continue with the ridge regression knowing that it is less sensitive to overfitting than other linear models.

Model	**Test Accuracy**
Linear Regression GD	0.7449
Linear Regression SGD	0.74408
Least Squares (Normal Eq.)	0.7449
Ridge regression (Normal Eq.)	0.7412
Logistic Regression SGD	0.74546
Regularized Log. Regression SGD	0.74544

Table I
Models and their accuracy performance

III. Parameter Selection Processing

A. Data processing and features engineering

The full data processing (with features engineering) has been implemented in the *process_data.py* file. The goal is to clean the data and generate new features to get more information from it (without over-fitting). To do this, we decided to do perform the following operations on the data.

1) Split the dataset into 4 subsets based on *PRI_jet_num* values (0, 1, 2 or 3)
2) Following this split, delete the features that have only one value (when the feature is useless for a given PRI_jet_num value)
3) Replace the remaining -999 values by the median of their respective feature (-999 values are excluded when computing the median)
4) Replace the outliers ($mean \pm 3\ std$) values by the median of their respective feature
5) Add $\sin(x)$ and $\cos(x)$ for all angle features
6) Standardize all the obtained features a first time
7) Add polynomial features
8) Standardize all the features a second time

We noticed while reading the features description, that the *PRI_jet_num* parameter was an extremely important feature as it was determinant for the existence or not of many other features. We, therefore, decided that our model will discriminate the examples based on this feature, and build a different model for each of those values. Then, we chose to generate $\sin(x)$ and $\cos(x)$ on angle features x to get more information from them. We finally performed a polynomial expansion of our features.

B. Hyperparameters tuning

To have the best performances, we needed to select the optimal *degree* and *lambda* for each of our models to fit as much as possible without over-fitting. (The *params_selection.ipynb* details the steps of the hyperparameters tuning).

We performed the pipeline shown in Figure 1, again and again, modifying our data processing until obtaining our best results.

Once done, we added the *degrees* and *lambdas* to the *final_model.ipynb* and *run.py* to perform our submission.

IV. Final Model

A. Implementation

Our final model consists of 4 ridge regression models (one per *PRI_jet_num* value) and can be found in the *run.py* file. We perform the processing explained in the section above (III-A). The dataset is divided at the beginning into 4 subsets (one per *PRI_jet_num* value), then goes into the processing part. Finally, the 4 regressions are trained using the hyperparameters found in the parameter tuning part (III-B).

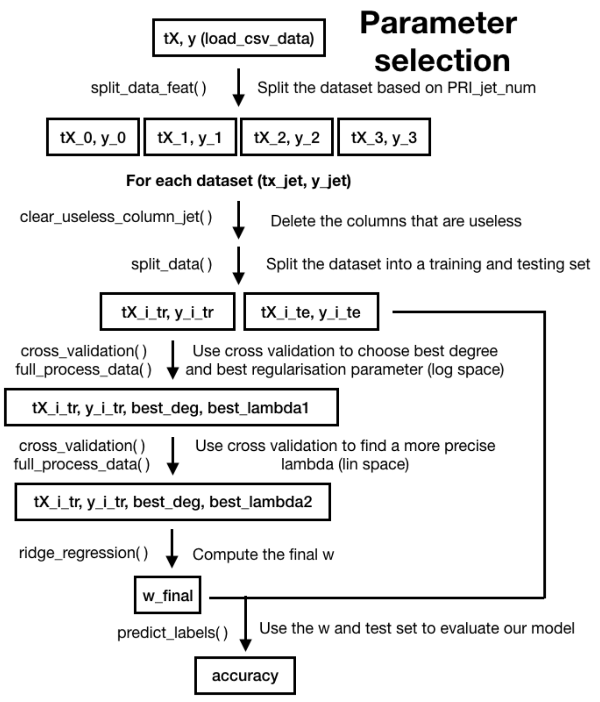
Parameter selection
tX, y (load_csv_data)
split_data_feat( )
Split the dataset based on PRI_jet_num
tX_0, y_0
tX_1, y_1
tX_2, y_2
tX_3, y_3
For each dataset (tx_jet, y_jet)
clear_useless_column_jet( )
Delete the columns that are useless
split_data( )
Split the dataset into a training and testing set
tX_i_tr, y_i_tr
tX_i_te, y_i_te
cross_validation( )
full_process_data( )
Use cross validation to choose best degree and best regularisation parameter (log space)
tX_i_tr, y_i_tr, best_deg, best_lambda1
cross_validation( )
full_process_data( )
Use cross validation to find a more precise lambda (lin space)
tX_i_tr, y_i_tr, best_deg, best_lambda2
ridge_regression( )
Compute the final w
w_final
predict_labels( )
Use the w and test set to evaluate our model
accuracy

Figure 1. Parameter selection pipeline iteration

B. Results

Using the final model previously described, we succeeded to pass the symbolic threshold of 80% accuracy on the test set. Then, to improve our results, we chose to use angle features, remove outliers, split the data using the jet_num which finally let us achieve an accuracy of 0.827 with an F1-score of 0.734.

V. Discussions

During this project, we thought about different techniques to improve our performances. These techniques have not been implemented in our project, either because of a lack of time or were out of the scope of this project.

First, from a model point of view, a neural network with the last layer of logistic regression or a random forest classifier could have done better than our simple ridge regression.

Then, for the data processing, we could have generated interaction features of different magnitudes (only on initial 30 features). This would have generated an important amount of features, so a PCA (Principal Component Analysis) could have been performed to reduce the number of features. Finally, the *PRI_jet_num* parameter could have been transformed into 4 binary features (as usually done for categorical features) instead of splitting the dataset into 4 subsets to avoid reducing the numbers of examples per ridge regression and avoid having to build 4 models.