Project 1 report for Machine Learning (CS-433)

Narek Alvandian*, Elia Fantini*, Anastasia Filippova*

School of Computer and Communication Sciences, EPFL, Switzerland
**Equal contribution, authors sorted alphabetically*

***Abstract*—In this paper we share our results on applying different machine learning techniques for the task of binary classification of the Large Hadron Collider (LHC) data. The overall goal was to get a maximum possible classification accuracy for differentiating between two types of signal: the "background" signal, and a "Higgs boson" signal. We used multiple preprocessing techniques with several optimization models, comparing and explaining the derived results. Our work resulted into $\tilde{1}0\%$ accuracy improvement over the baseline model.**

I. Introduction

Given tabular data of sensor recordings from LHC at CERN an AiCrowd competition participants were expected to solve a binary classification task. Two classes that needed separation were "signal" and "background", representing the presence of the Boson Higgs, or the absence of it respectively. Our team took part in this competition, and reached a final accuracy of 83%, which is a significant improvement over the baseline solution, which was only able to get accuracy of 72%.

We did an exloratory data analysis(EDA), used different preprocessing techniques, inferenced multiple optimization algorithms, implemented hyper-parameter search and used cross-validation to reach the stated result.

In further sections we will describe above mentioned steps in detail, explaining the reasoning behind each decision and comparing the according effects.

II. Methods

A. Exploratory Data Analysis and Preprocessing

We want to be familiar with the data we are working with - this is what we achieved in our EDA. We get insights about out data, which later will be used for meaningful preprocessing. We measure the "usefulness" of each of our preprocessing steps by comparing the resulting accuracy of a fixed estimator before and after the preprocessing. Details and results of the evaluation can be seen in subsection II-C.

First, we need to see how our features are distributed. We do this to get an understanding of each feature's distribution. This will be useful for handling outliers, missing values, or potentially "uninformative" features.

As can be seen on the Figure **??**, there are certain features, which have a majority of their value set to -999. This means that the sensor was not able to provide measurements. For handling this particular scenario we decided to introduce a new binary feature, which will indicate whether the particular measurement was successful or not. We also substitute the -999 value with the mean value of according feature.

We can also notice that one of the variables takes only 4 distinct values. We interpret this as a "categorical" value, and decide to encode it in a one-hot manner (O). This way we remove one column, and add 4 binary columns, each representing whether the measurement takes one of 4 category values or not.

One more option is the correlation analysis (C). Generally speaking - if two features are highly correlated, they might give our model the same information, so it can make sense to get rid of all pairwise correlated features except one. Correlation matrix can be seen on Figure **??**. Indeed, there are some highly correlated features, so we might only leave one of them for our training dataset.

Another step of our preprocessing is scaling the data (S). We can see on Figure **??** that some of our feature values are not exceeding 10 in absolute value, while others can be orders of magnitudes greater. We scale our data to remove this difference, so that our model doesn't think of features with greater values as of more important ones. We look at two types of scaling: min-max scaling, and normalizing.

Taking logarithm (L), of the features might prove useful as well. Logarithm reduces the range of a variable so the differences are preserved while the scale of the difference is not so dramatic as it would've been for skewed distributions. This operation makes our feature to be distributed (almost) normally.

Polynomial features. Since we are using linear models - we only can model linear relationships. If our target is not linear in some feature, using polynomial features might help us to restore the right dependency. We use the squares of our features (SQ) and cross features of the first order (CR).

And finally, it might be useful to add a bias term, so it can capture other the dependencies, that our normal features do not capture.

B. Optimization algorithms

In our experiments we used three types of models, which differ by the loss function they minimize and their approach of finding the solution.

First type - are the models which find the analytical solution of the Mean Squared Error (MSE): Least Squares Optimizer (LSO) and Ridge Regression (RR).

Second type - are the models that find the minimum of the MSE iteratively using the gradient descent: Gradient Descent (GD), Stochastic Gradient Descent (SGD) and Batch Gradient Descent (BGD).

TABLE I: Values of accuracy for different preprocessing steps

S	M	S,B	O,S	O,S,B	S,L	S,L,B	O,S,L	O,S,B,L	O,S,B,L,SQ	O,S,B,L,SQ,C	O,S,B,L,CR	O,S,B,L,CR,C
0.728	0.690	0.726	0.733	0.755	0.70	0.726	0.722	0.757	0.816	0.816	**0.8323**	0.8322

TABLE II: Values of accuracy for different optimization algorithms

LSO	RR	GD	SGD	BGD	LR	LRR
0.80	0.806	0.697	0.695	0.721	**0.812**	0.808

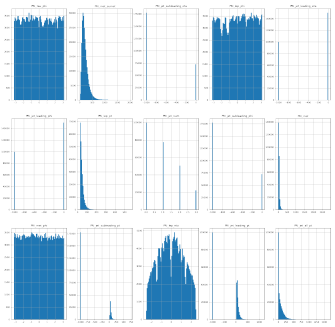

Fig. 1: Distribution of as subsample of features from the dataset

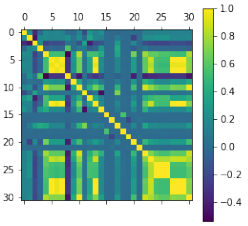

Fig. 2: Pairwise correlation matrix for all features of the dataset

Third type - are the models that find the solution to the iteratively find the solution to the Logistic Loss - the negative Log Likelihood of the observed data. Logistic regression (LR), and logistic regression with regularization (LRR).

C. Performance evaluation

In this section we describe the evaluation for both preprocessing and optimization algorithms.

For both preprocessing evaluation and model evaluation we used K-fold cross validation with 20% data being in the test set and K = 4 folds.

For preprocessing "usefulness" evaluation we chose the folowing algorithm. We select a fixed model (Logistic Regression in our case) and evaluate it just on a scaled data, getting the baseline accuracy of 72%. After that we only applied one preprocessing per step, and evaluate the model to see the change in accuracy. If the metric increased, then we consider this preprocessing step to be useful and we apply the next step on top of the current one, if not - we didn't use this step for our further experiments dataset. Results can be seen in I. After making our decision on which preprocessing steps are useful, we apply all of them to both training and testing (submission) sets. We want to notice that we understand that the more precise way would be to study effects of all possible feature combinations, yet for this project it would've been an overkill with too much of a computational time.

For model selection, we get our preprocessed dataset and use it to train and inference all models with a hyper-parameter search and K-fold(K=4) cross-validation for each of the algorithms. Results can be seen in Table II.

III. Final setup

As can be seen on Table I and Table II, our best performance was obtained with the following preprocessing: One hot encoding, standartizing, adding a bias term, applying a logarithm, and taking cross products of our features. The best performing model is the logistic regression.

IV. Conclusion

In our paper we showed the effect of different preprocessing techniques, and different models on the accuracy of classification of the Large Hadron Collider (LHC) data. By finding the optimal setup we improved the baseline result by more than 10% in target metric.