Machine Learning Project 1: Wait... Is this Higgs boson?

Amin Asadi Sarijalou, Ahmad Rahimi, Parham Pourmohammadi
amin.asadisarijalou@epfl.ch, ahmad.rahimi@epfl.ch, parham.pourmohammadi@epfl.ch
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

I. Introduction

In this project, we aim to use machine learning techniques in order to predict whether a proton collision event's decay signature, with a given set of features, is resulted by creation of the Higgs Boson particle in the process of proton collisions in an accelerator. Our approach is to train several classifiers on top of a dataset of collision event signatures provided to us by CERN. For this purpose, we first analyzed and cleaned the dataset and processed the features. Then we proceeded to train and evaluate several classifier models. In this report, we first describe our approach to prepare the data in several steps. Then we explain the models we used. We then describe the details of the results we obtained for each model. At the end, we state the conclusions we made by observing the results that we obtained.

II. Exploratory Data Analysis (EDA) and Data Cleaning

The dataset contains data points with 30 different features. Only one of these features (*PRI_jet_num*) is categorical and the rest have continuous values. *PRI_jet_num* takes values between 0 and 3 and specifies the number of jets, as indicated in the data sheet. For each of the values of this categorical feature, some measurement have or have not been done at all. For instance, when *PRI_jet_num* is equal to 0, *PRI_jet_leading_pt* is not defined, and therefore is not measured and it is missing from our dataset. Consequently we partition our main dataset into 4 separate sub-datasets, one for each unique value of *PRI_jet_num*. We will indicate the sub-dataset of points with *PRI_jet_num* = i as dataset i. We will train a separate model for each of these 4 sub-datasets. We removed the *PRI_jet_num* in each dataset i as it is not informative for us anymore. We continued by visualizing the features in each dataset i using histograms to observe the feature distributions, and using boxplots to detect potential outliers in the data, which we will explain in the subsequent sections.

Filling Missing Values: For feature *DER_mass_MMC*, about 26%, 9%, 5%, 6% of the values are missing in datasets 0, 1, 2, and 3, respectively. As this rates are not high, we decided to fill these missing values. Using the boxplots, we observed that the distribution of *DER_mass_MMC* is highly skewed. As a result, we filled the missing values of this feature with the median of the non-missing values, as median is a better representative for skewed distributions.

Removing Outliers: According to the boxplots we plotted for each feature, there were some excessively large values in some features. This harms the performance of least squared based algorithms as their loss function is highly sensitive to these excessive values. We took a common statistical approach for tackling the issue of outliers: we removed the values that were below $Q_1 - 1.5 * IQR$ or above $Q_3 + 1.5 * IQR$, where Q_1, Q_3, and IQR are first quartile, third quartile, and interquartile range.

III. Feature Engineering

Feature Scaling: Some features take much larger values than others. This can cause problems for many classification algorithms such as ridge regression. Ridge regression will less penalize the features that have much bigger values because these features have smaller corresponding weights, thereby treating the features unfairly. Therefore, we standardize every feature (so they will have zero mean and unit variance) to prevent this problem.

Feature Expansion: We expanded the set of original features for each dataset i by adding to our set of features 1- multiplications of each pair of original features, 2- powers of two to six of the original features, and 3- sine and cosine of the original features, thereby introducing non-linearity to our classification models. After this step, there were 279, 407, 638, and 638 features in datasets 0, 1, 2, and 3, respectively.

IV. Modelling the classifiers

The set of classifiers we used were 1-*Least Squared classifier*, 2-*Ridge Regression classifier*, 3-*Logistic Regression classifier*, and 4-*Regularized logistic regression classifier*. Although we implemented the gradient-based regression methods other than logistic regression, we did not use them as we can simply find the global minimum of the same mean squared loss function using ridge regression (normal equations). As mentioned before, we trained a separate instance of each of these models for each dataset i. These different instances differ in the values of their parameters and/or their hyper-parameters. Afterwards, at the prediction time, we will use the model trained on dataset i in order to predict the labels of the datapoint in dataset i.

A. Training, Validation and Testing

We split our data to train and (local) test sets with ratio of 0.9 and 0.1, respectively. We performed 4-fold

cross-validation on the train set to evaluate and tune the hyper-parameters of each model. We performed all the data cleaning and feature engineering once for the train set and once for the test set. The accuracies and F1-scores that we report, have been produced on the local test data set that we split from our data (10% of data).

V. Results

The best results were achieved by Ridge Regression classifier. The fact that Ridge Regression performed better than logistic regression was surprising to us, as we expected the logistic regression to outperform other models, since its loss function is, by definition, more suitable for (discrete) classification tasks. The accuracies and F1-scores reported in the tables are calculated as weighted average of the accuracies and F1-scores of the models trained on datasets 1, 2, 3, and 4.

A. Ridge Regression and Least Squared (*Best Results*)

As Least Squared is a special case of Ridge Regression method with lambda equal to zero, we discuss these models at the same time. As we can see in table II, the ridge regression model, has a higher accuracy and F1-score than the least squared method. The training accuracy and training F1-score of the least squared model was 0.837 and 0.726. Comparing these scores with table II, shows that least squared classifier over-fitted the data, and the penalty term of the ridge regression classifier successfully mitigated this problem. Furthermore, from the same table we can conclude that doing feature expansion has increased the accuracy by more than 3%. Also, adding multiplications of features is more effective than other feature expansion methods, as the accuracy drops by 1% when we do not use it. It worth's mentioning that feature expansion using sine and cosine slightly (10^{-3}) improves the F1-score. In all the experiments in table II we have fixed the lambdas for ridge regression to the values in table I.

	dataset 0	dataset 1	dataset 2	dataset 3
lambda	1.06×10^{-5}	1.14×10^{-5}	1.14×10^{-5}	10^{-5}

Table I
THE SET OF REGULARIZATION PARAMETERS (LAMBDAS) FOR EACH DATASET i, FOUND USING 4-FOLD CROSS-VALIDATION

B. Logistic Regression

We discuss Logistic Regression (LR) and it's penalized (regularized) version (RLR) in this section. We experimented these two models with several different configurations and summarized the results in table IV. We didn't use powers of the features as a feature expansion method because it made the training unstable due to an overflow caused by logistic loss function. We used a learning rate of $gamma = 10^{-3}$ and 5000 iterations for all the experiments. We also fixed the lambdas to the values in table III for the experiments with LLR. We can conclude from table IV that none of the models has good performance when we do feature expansion, i.e., high feature dimensions. We can also observe that RLR has 1% better accuracy and 1% less F1-score in comparison to LR. As we previously mentioned, the fact that logistic regression has worse performance comparing to its competitor, ridge regression, does not agree with our expectations.

Model	Feature Expansion	Accuracy	F1-score
Least Squared	mult, sin, cos, powers ≤ 6	0.644	0.552
Ridge Regression	mult, sin, cos, powers ≤ 6	**0.832**	**0.717**
Ridge Regression	mult, sin, cos, powers ≤ 5	0.831	0.717
Ridge Regression	mult, powers ≤ 5	0.831	0.716
Ridge Regression	mult, sin, cos	0.830	0.714
Ridge Regression	sin, cos, powers ≤ 6	0.823	0.702
Ridge Regression	-	0.792	0.639

Table II
PERFORMANCE COMPARISON BETWEEN LEAST SQUARED AND RIDGE REGRESSION CLASSIFERS AND ABLATION STUDY OF FEATURE EXPANSION

	dataset 0	dataset 1	dataset 2	dataset 3
lambda	10^{-5}	10^{-5}	10^{-5}	10^{-3}

Table III
THE SET OF REGULARIZATION PARAMETERS (LAMBDAS) FOR EACH DATASET i, FOR RLR

Model	Feature Expansion	Accuracy	F1-score
LR	mult, sin, cos	0.670	0.471
LR	-	0.756	0.573
RLR	mult, sin, cos	0.670	0.474
RLR	-	**0.761**	**0.562**

Table IV
PERFORMANCE COMPARISON BETWEEN LR AND RLR CLASSIFIERS AND ABLATION STUDY OF FEATURE EXPANSION

C. Performance on AICrowd

The predictions produced by our best model achieved **0.835** accuracy and **0.746** F1-score on the competition website. For the sake of producing the labels for the submission platform, once we submitted the labels produced by the model trained on the whole data (train+local test).

VI. Conclusion

In this project we successfully modeled the given data with a desirable accuracy of 83.5% using Ridge Regression model, which not only prevented overfitting and demonstrated better performance than other models, but also was much faster. We also demonstrated the importance of data cleaning and feature engineering in the results we obtain.