Class Project 1 Report

Angerand Grégoire, Goullet Boris, Grondier Julien

***Abstract*—In this project, we were given a dataset consisting of LHC measurements (and derived data) of particle collisions. Our task was to apply different machine learning methods to this dataset to train a model that would predict whether or not those measurements correspond to a Higg's Boson or not.**

After trying different methods, we were able to attain more than 80% prediction precision using 4 different ridge regression models, each with separate grid-searched hyperparameters.

I. Exploratory Data Analysis

First, we started off this project by opening up the dataset in a spreadsheet software.

We quickly realized that a number of columns were sometimes undefined (-999). However, we also noted that, apart from DER_mass_MMC, which features were defined was completely determined by the value column 22, PRI_jet_num, which, incidentally, is the only categorical column.

We also plotted the distribution of each feature, relative to whether the event was categorized as a HB. These plots can be found in data/images/. They show that some features have a normal distribution (which is good for linear regression), but others seem to have a reciprocal or parabolic distribution.

Feature processing

Since we wanted versions of features with normal distributions, we decided to construct artificial features from existing ones and add them as feature columns. As such, our prepare_data method adds inverse, log, and polynomial versions of the individual feature columns when possible.

II. Models applied

We started off by applying a simple gradient descent with linear regression on our training set. However, this gave us mediocre results (around 75%).

So we started using ridge regression (with MSE). This quickly gave us much better results, even with arbitrary λ values: 80.4% for $\lambda = 0.001$ and polynomial features up to degree 7.

We later tried using regularized logistic regression with GD, however the results were on par with unpenalized linear regression, so we abandoned that idea.

PRI_jet_num	λ	degrees	error rate
0	$1 * 10^{-5}$	1	0.168
1	$2.5 * 10^{-5}$	7	0.214
2	$1 * 10^{-5}$	2	0.201
3	$1 * 10^{-4}$	7	0.209

Table I
LOWEST ERROR RATE λ AND DEGREE HYPERPARAMETERS FOR EACH SUBSET

Bucketing and hyperparameters

We implemented a grid search method with 4-fold cross-validation to find the best value of λ and of how many degrees of polynomial features we wanted; however that didn't really improve our results.

Instead, we decided to separate the dataset into 4 subsets, according to PRI_jet_num. Then we trained a different ridge regression model for each "bucket", which could ignore features that were undefined for that category, and specialise more.

We went even further by gridsearching hyperparameters per bucket: this gave us 4 different λ, degree pairs, the "optimal" ones for each bucket (see table I). This let us reach 80.7% precision rate on the Kaggle leaderboard.

III. Discussion

A. *Logistic Regression*

While we were unable to reach a meaningful result with regularized logistic regression, we believe this is not because it is not the right model for this dataset (on the contrary, this dataset looks like a perfect fit to apply logistic regression on), but rather because, as we were all beginners in ML and Data Science in general, our implementation of logistic regression may be faulty.

B. Bucketing

While our separation of the dataset into 4 different subsets gave us better results in this case, it also meant each model had less data to be trained on. Since the original dataset was large enough, this was not a problem in this case; however, with a smaller dataset, this might have caused our models to massively overfit, especially when adding multiple polynomial features (we have upwards of 200 features when we build polynomial features up to degree 7!). We tried using different bucketing strategies, notably sorting the samples depending on their defined values. This creates many categories with too few elements to be able to train a model efficiently.

C. Loss functions

We used the failure rate on our test set as the error metric for our gridsearch. We found that RMSE gave us results that were unrepresentative of the model real performance.

IV. Summary

A simple linear regression quickly gave us a reasonable score of 0.76. Basic feature engineering brought us to 0.79. Upgrading to ridge regression allow us to reach 0.84. Any attempt at further progress was met with failure except for giving each bucket its own set of hyperparameters.