CS433 Machine Learning Project 1

Francesco Bardi, Samuel Edler von Baussnern, Zeya Yin

I. INTRODUCTION

This project's goal was to tackle the Kaggle Higgs Boson challenge as part of the *CS433 Machine Learning* course at *EPFL*. This challenge is to distinguish between two events: one being noise the other being the detection of a Higgs Boson. We tried multiple linear models (constrained by the exercise), feature selection methods and more complex multi-model design methods. The highest score on Kaggle's hidden data set was over 0.82.

II. DATA ANALYSIS

The data set consist of 30 features and 250000 observations, all but one variables are numeric the other one being categorical, referred to as jet. A binary class label is also given. Roughly half of the features are based on the other half and show a high correlation. The variables show different value-ranges, standardisation is thus highly required. The data set is unbalanced, with a 65.7% being noise. This ratio varies for each subset defined by its jet.

III. OUTLINE

Our simple model consists of two steps:

a) Pre-Processing & Feature Engineering b) Fitting the model

We outline these steps in more detail below.

Our final model however consists of multiple simple-models, each with it's own set of parameters and weights. It wraps the above mentioned two steps around a subset selection based on the categorical value `PRE_jet_num`.

IV. PRE-PROCESSING & FEATURE ENGINEERING

This steps consists of both basic data handling as well as feature engineering. When ever applicable the transformation is *learned* using the *training set* and the same transformation is then applied on the *test* and *prediction set*, we denote this with the term *[...] is stored*. As some of these steps are intertwined we present them in the order of application.

A. Removal of redundant features

Redundant features, identified by the absolute value of the pair-wise correlation which exceed a certain threshold, are being removed. The list of these features is stored.

B. Removal of features with too many undefined values

Features where the amount of undefined values is too large are removed. The list of these features is stored.

C. Dummy encoding of categorical values

We only identified one categorical feature `PRE_jet_num`. A dummy encoding was added to the feature matrix. The old column was removed. This applies only to the single-model case (see V-B).

D. Feature Expansion

In order to overcome the linearity of the given classifiers we add multiple transformations of the base features to the feature matrix:

Bias Column of constant value 1,
Polynomial $\phi(x) = x^i$, $\forall i$, $1 < i \leq 6$,
Combinatory Pairwise difference and product of the features: $\phi(x) = x_i * x_j$, $\forall i, j$, $i \neq j$ and $\phi(\mathrm{x}_i, x_j) = \|\mathrm{x}_i - x_j\|$, $\forall i, j$, $i \neq j$,
tanh $\phi(x) = tanh(x)$,
log $\phi(x) = log(x + 1000)$ The term 1000 stems from the given *undefined-value* which is -999,

E. Handling undefined values

As some features are undefined for some jets, we replace those values with 0. This comes from the data set description provided from CERN [1]. These values are ignored in the standardisation step.

F. Standardisation

All features (except the categorical ones, and the bias feature) are standardised to have a mean of 0 and a standard deviation of 1. For the computation of the mean the undefined values where ignored. This transformation is stored.

G. Oversampling

Given that the data is not balanced we oversample the minority class. However our implementation as turned out to yield worse results. This was therefore not applied in the final model.

H. Merging of categories

The second attack on the balance was to merge samples which take either 2 or 3 as their `PRE_jet_num`, as size of these subsets is quite small compared to the other two subsets.

I. Feature selection based on Lasso

Given that Lasso converges to a sparse weight matrix we selected the weights with an absolute value larger than a certain threshold. We used the Lasso Shooting Algorithm[2]. This selection was saved. See appendix 1 for more information.

V. Model design

After the upper mentioned steps a linear classifier is fitted to the data.

A. Hyperparameter search

The choice of the right parameters is of paramount importance, for example the regularization parameter λ_{LASSO} is sensitive to data perturbation; the common practice is to perform cross validation to gain an unbiased estimation of the performance however due to time constraints we leave this as a future exercise. The hyperparameter search was done over the train-test-split, and the model with the highest test-score was selected as the final model.

B. Multi-Model approach

In order to tackle the categorical feature and the undefined values for each jet we chose to split the data into subsets according to their jet-value. For each subset a different model was trained using the above mentioned pipeline. This results in at most four different sub-models.

C. Choice of linear classifier

We tried multiple linear classifiers, see table I, and chose *Ridge regression* as our linear classifier. These models were trained with the single-model approach without feature expansion, with the removal of highly correlated features, and merging of categories. See `src/functions/implementations.py` in the code for more information on the chosen parameters.

Given the primarily test results, problem description, easy of implementation and run-time speed, which allowed us to explore other ranges in the pipeline as well, we chose Ridge Regression as our base model.

D. Scoring

Given that our model's performance was going to be measured on its accuracy we selected the accuracy as well. We would like to stress that in the general case of an unbalanced data set a different score, e.g. F1, might be more informative.

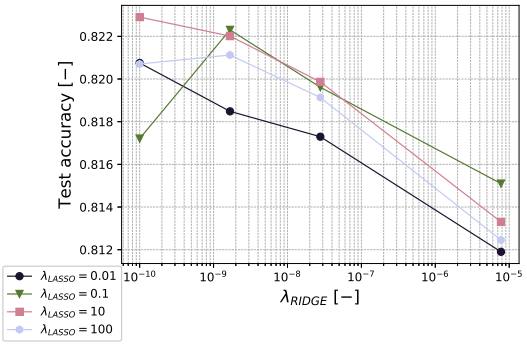

Fig. 1. The test accuracy given by different λ_{LASSO} with fixed λ_{RIDGE}

Model	Train accuracy	Test accuracy	λ	γ
Least squares GD	0.74	0.74	\	1e-03
Least squares SGD	0.73	0.73	\	1e-03
Least squares	0.74	0.73	\	\
Ridge regression	0.71	0.71	1.0	\
Logistic regression	0.74	0.74	\	1e-06
Regularized logistic regression	0.74	0.74	1.0	1e-06

TABLE I

Mandatory function results for primary model.

Method	Train acc.	Test acc.	Feature No	λ_{LASSO}	λ_{RIDGE}
Single model	0.80	0.80	304	10.0	10e-9
Multi model	0.82	0.83	529	10.0	10e-9

TABLE II

Optimal results for final two models

VI. Results

We refer to table II for the results and main parameters of the best models.

VII. Discussion & Conclusion

As can be seen in figure 2 the derived features, and especially the added features through expansion have a huge influence on the model, this gives rise to future steps of expanding the feature engineering step.

Although the model performance surprisingly well we would like to point out that even though the final model as an average of 529 features it does not overfit in general. This could either mean that the proposed feature expansion steps are not expansive enough or that the chosen model is not an appropriate model for the given data. We estimate that a Gradient Boosting Algorithm would perform much better, and still be interpretable. Which which in this case is probably a highly valuable target of the model, since we do not only want to get good predictions but only understand the underlying physical relationships.

VIII. Reference

References

[1] Cecile Germainc Isabelle Guyond Balazs Kegl David Rouseau Claire Adam-Bourdariosa, Glen Cowanb. Learning to discover: the higgs boson machine learning challenge.

[2] Kevin P. Murphy. *Machine learning : a probabilistic perspective*.

Appendix

Lasso

Among the various methods to perform feature selection l_1 regularization techniques have the advantage that the selection process is part of the learning algorithm since they can converge to a sparse weight matrix. The most appropriate learning algorithm for classification problems is the logistic regression due to its robustness to leverage points. However the large number of features generated with the expansion process make the regularized logistic computationally very expensive, moreover due to its simplicity in the implementation the LASSO regression was preferred. The LASSO equation can be solved by means of the coordinate descent

method, which solves a one dimensional minimization problem at each iteration. The Lasso shooting algorithm is very appealing since it relies on an analytic expression for the minimization step(see Algorithm 1).

Data: $\mathbf{X}$, $\mathbf{y}$
Result: optimal weights $\mathbf{w}$
-Initialize w with RIDGE regression
while $convergence$ **do**
  for *j in 1,...,D* **do**
    $a_j = 2\sum_{i=1}^{n} x_{ij}^2$;
    $c_j = 2\sum_{i=1}^{n} x_{ij}(y_i - \mathbf{w}^T\mathbf{x_i} + w_j x_{ij})$;
    $w_j = soft(\frac{c_j}{a_j}, \frac{\lambda_{LASSO}}{a_j})$;
  end
end

Algorithm 1: Lasso shooting algorithm

Feature importance

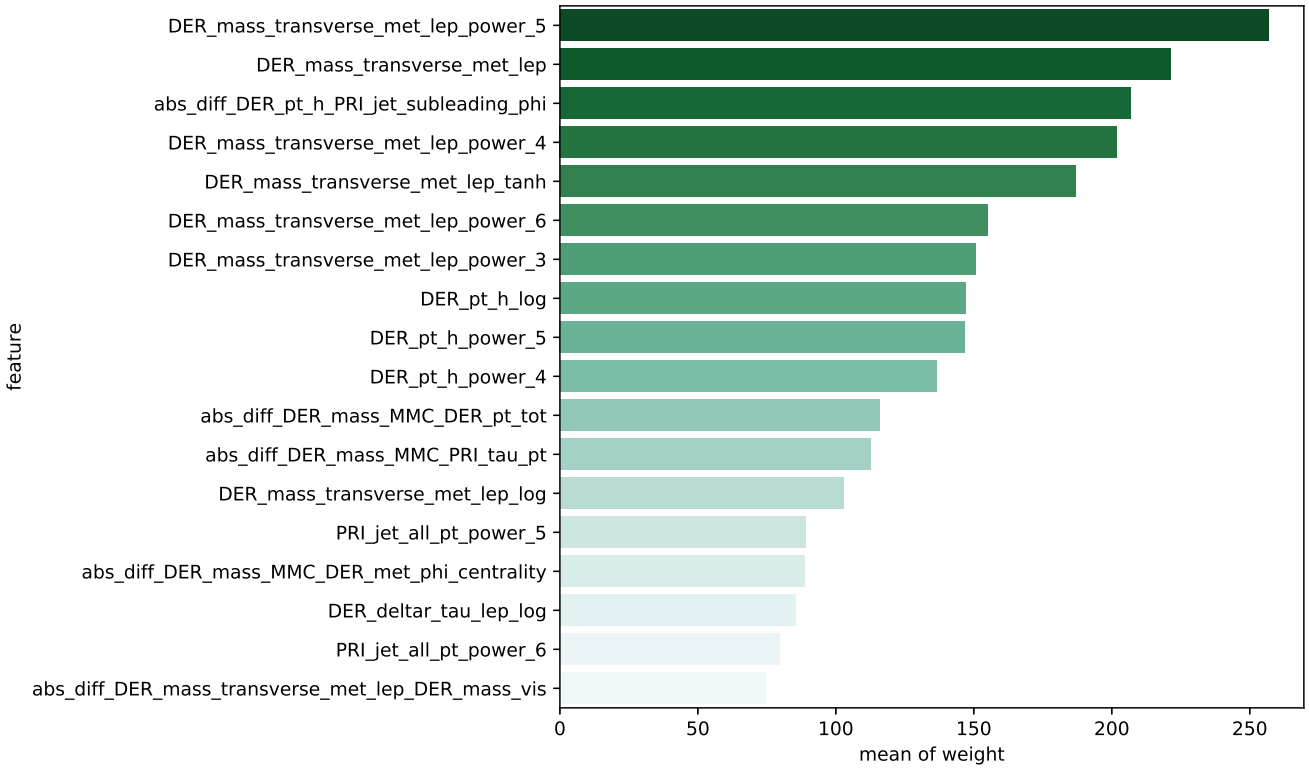

Fig. 2. Average absolute weight of the features which determine 50% of the total weights. The given labels follow this naming schema: <feature >_<function> for univariate and <combinator >_<feature a>_<feature b> for bivariate feature expansion.