Project 1: Higgs Boson Classification

Marko Lisicic
Section of Computer Science
EPFL

Lazar Milikic
Section of Computer Science
EPFL

Yurui Zhu
Section of Digital Humanities
EPFL

***Abstract*—The purpose of this project is to implement basic Machine learning models and then use them to train and later predict whether a particle with given features is classified as a Higgs boson or not. As our input data has an abundance of missing values, outliers, and features with heavy-tailed distributions, thus preprocessing and feature engineering play a crucial role in improving classification accuracies. The final result of this project yields 0.84 categorization accuracy and an F1 score of 0.757 [submission #204396].**

I. Introduction

In this project, we wish to design a Machine learning classifier that could exploit the given features representing detected signals of collision aftermath and recognize with confidence the Higgs particle in the Higgs boson problem.

II. Methods and models

We approach this problem by splitting it into two sub-tasks. First, we focus on data prepossessing and feature engineering where we exploit, extract, and emphasize the most significant information to maximize the effectiveness of our learning algorithms. The second part consists of the selection, building, and hyper-parameter tuning of models with the main goal of maximizing classification accuracy.

A. Data preprocessing

Each sample has 30 features and the label: s for the Higgs boson signal event, or b for the "background" event. The training data set consists of 250 thousand data points. Although immense in size and features, many entries do not represent a meaningful or valid value. The test set contains over 550 thousand different samples and similar distribution with training data set.

1) PRI_jet_num feature: Unlike all other features which have continuous values, the PRI_jet_num has a discrete domain. This feature represents the number of jets with values in the set $\{0, 1, 2, 3\}$ (possible larger values have been capped at 3 [1]). In addition, we know that all missing features are related to PRI_jet_num (except feature DER_mass_MMC). If PRI_jet_num equals 0, some subsequent features cannot be computed or measured at all as there is no physical sense to them [2]. Figure 1 proves this observation, where when PRI_jet_num = 0, 10 features have no meaningful value, and 1 feature (which is actually DER_mass_MMC) has 77790 out of 99913 available values, and the rest 19 columns are fully filled. The situation for PRI_jet_num = 1 is similar to when PRI_jet_num is zero. On the other hand, for PRI_jet_num = 2 and PRI_jet_num = 3, we have all fully filled features with

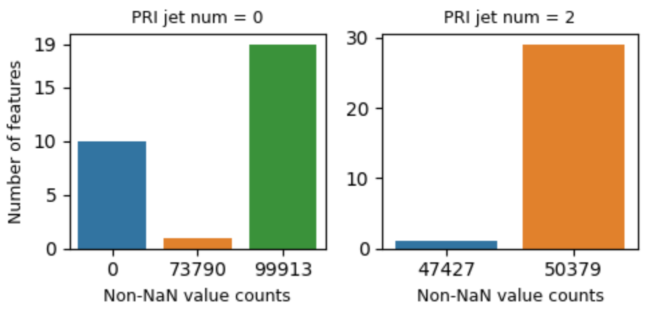

Fig. 1. Count plot of features with respect to how many values they are missing in PRI_jet_num subsets. Left when PRI_jet_num = 0 (max value count 99913). Right when PRI_jet_num = 2 (max value count 50379).

Therefore, dividing the whole dataset based on the values of PRI_jet_num leads to having either fully filled or fully empty features in each of these subsets (except for DER_mass_MMC). Thus, the dataset is split according to this feature and trained separately. Note that since for PRI_jet_num = 2 and PRI_jet_num = 3, we have the same state (the same features are fully or partially filled among two groups), we group them into one subset. When predicting a data point, the prediction model to be applied corresponds to the PRI_jet_num value of the data point.

2) Missing Value: By the definition of this dataset, the value of -999.0 is considered meaningless or non-computable. Therefore, we consider these are missing values. Moreover, more than 70% of the events lack at least one feature.

Thanks to the data splitting according to the value of PRI_jet_num, for all the subsets we form, all features, except for DER_mass_MMC, are either fully filled or have all values missing. Because these missing values are structurally absent [2], we are free to drop all these columns.

However, we still need to handle the DER_mass_MMC feature, because it has missing values in all 3 subsets we form. Because of its heavy-tailed distribution and outliers, we select to input median value to replace missing values as it is a more robust estimator. In addition, we learn that if the value of this feature is NaN, the probability of that event is background signal (-1) is approximately 92.5%. Therefore, we add a flag feature for denoting the absence.

3) Feature Engineering: Many of the features exhibit heavy-tail distributions. As heavy-tail distributed features manifest exponential properties, we can use $x \mapsto \log(x + 1)$ transformation on them under the condition of non-negative.

The log transformation can make distribution less sparse and noisy. The log transform is applied to features [0, 1, 2, 3, 5, 8, 9, 10, 12, 13, 16, 19, 21, 23, 26, 29].

Another set of features which includes [15, 18, 20, 25, 28] columns represents angles. We use the trigonometric transformation of these features to help the model better grasp some physical properties of the events, like the particle's momentum after the collision [2].

4) Outliers: As regression models are often not robust to outliers (especially with MLE loss), we remove them by the inter-quartile range (IQR) method using $[2, 98]$ percentile.

5) Feature augmentation: We use polynomial expansions to augment features to help linear classifiers handle non-linear decision boundaries better. Additionally, we rely on the assumption that the presence of the Higgs boson depends on the interaction of multiple parameters. Thus, for the given features and a *degree*, we expand each features up to the given degree, and then integrate products, sums, and transform $(\mathbf{x}_1, \mathbf{x}_2) \mapsto \mathbf{x}_1^2\mathbf{x}_2$ for each pair of features.

6) Standardization: To help to learn and prevent putting emphasis on features with larger values, we apply standardization to all features of the model.

B. Model selection and evaluation

We build and test 6 main models: Gradient Descent (GD), Stochastic Gradient Descent (SGD), least squares, ridge regression, logistic regression and Reg Logistic regression.

Each of these models has a polynomial *degree* as hyperparameter, while ridge regression and Reg Logistic Regression also use regularization factor λ. To find a good bias-variance trade-off and maximize the classification accuracy, we perform hyperparameter tuning with combinations of values for (*degree*, λ) pairs.

In addition, as we have noticed that models that rely on a gradient-decent algorithm had trouble converging with constant learning rate γ, we implemented the Goldstein-Price algorithm that at each step computes the most appropriate step-size. This resulted in better convergence but at a price of additional complexity $O(N^2D^2)$, where N is the number of samples, and D number of features.

III. Experiments and Discussion

In Table 1, we report the validation accuracy for each of the models. We can clearly see that the Ridge regression has performed the best.

Method	Validation accuracy
Gradient Descent	0.779 ± 0.001
Stochastic Gradient Descent	0.721 ± 0.001
Least squares	0.837 ± 0.002
Ridge regression	0.840 ± 0.003
Logistic regression	0.787 ± 0.001
Regularized logistic regression	0.789 ± 0.002

Table 1. Performance of the models

Table 2 shows the the hyperparameters that we select for our Ridge regression model. For the search for the best

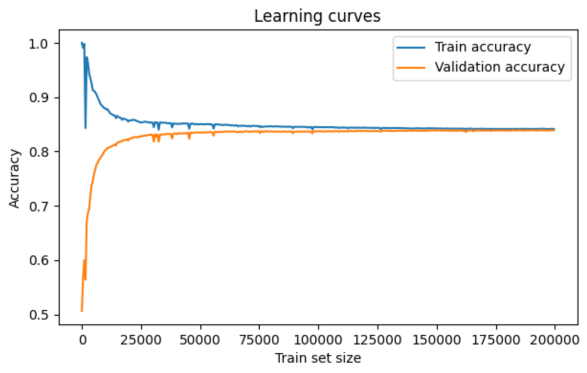

Fig. 2. Learning curves for train and validation accuracy with respect to the training set size.

parameters, we evaluate the cross-validation accuracy of all pairwise combinations for λ in the range from $[0, 10]$ spaced evenly on a log scale, and degrees between 1 and 17, selecting the best results. We report the best overall accuracy score to be 0.84 (the 5th-best accuracy on the public ranking of AIcrowd).

JET	Degree	λ	Validation accuracy
0	15	10^{-4}	0.849 ± 0.005
1	16	10^{-4}	0.816 ± 0.001
2 & 3	15	$5 \cdot 10^{-4}$	0.848 ± 0.005
Overall	-	-	0.840 ± 0.003

Table 2. Best parameters for the Ridge regression model

Additionally, in order to verify model result, we draw learning curves for training and validation accuracy with respect to the train set size. From Figure 2 we can conclude that our model achieves a good balance between bias-variance trade-offs.

Finally, we find it rather surprising that the Ridge regression model outperforms by a clear margin both logistic regressions which are specifically designed to handle classification problems. However, in order to complete the training for Logistic regression, we have to make several a couple of compromises. Standardization has to occur after feature augmentation to avoid value overflow when dealing with exponents, we cannot explore the same degree levels as it makes convergence very difficult, and due to its time consumption, we cannot fully exploit cross-validation to find the best hyperparameters for these models.

Therefore, the Ridge Regression model seems to be the ideal model for this problem. It provides an analytical solution that is computed immediately, and it utilizes regularization to avoid overfitting.

IV. Conclusion

The results of our work unequivocally demonstrate the importance of a profound understanding of the dataset and proper data preprocessing before applying any Machine learning algorithm. After thorough data analysis, handling missing values, and feature augmentation, we achieve a very respectable categorical accuracy score of 0.84 with the Ridge regression model, one of the simplest algorithms in Machine learning.

References

[1] Learning to discover: The Higgs Boson Machine Learning Challenge. (n.d.). Retrieved October 30, 2022, from https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf

[2] Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kégl, B., & Rousseau, D. (2014). The Higgs Boson Machine Learning Challenge. Proceedings of the 2014 International Conference on High-Energy Physics and Machine Learning - Volume 42, 19–55.