Project 1: Detecting Higgs boson

Rolando Grave de Peralta Gonzalez, Maxime Rufer
{rolando.gravedeperaltagonzalez, maxime.rufer}@epfl.ch
Department of Computer Science, EPF Lausanne, Switzerland

ABSTRACT

1 INTRODUCTION

In March 2013, the Large Hadron Collider at CERN announced discovery of the Higgs boson particle. This is an elementary particle in the Standard Model of physics which explains why other particles have mass. The Higgs boson decays rapidly into other particles, this is the reason why it is rather detected by chain of products in which it decays. Scientists call this it's "decay signature". In this report we will be classifying wether a "decay signature" corresponds to that's of a Higgs boson or not.

Ajouter? Modif?

2 METHODS

2.1 Data

2.1.1 *Exploratory Data Analysis.*

Data Statistical Properties. Before creating baselines and complex models, we explore the given data. When looking at the data we first check for missing or wrongly formatted data. Which we follow by analysing basic statistical properties of our dataset and we see that we have large means, ranging from ≈ -709 up to ≈ 210. We also see that our data has some features with high variance, up to ≈ 658.

Add plots of mean/std ?

Class Distribution. When looking at our dataset we can see a big imbalance in the representation of both classes. We have over 164'333 samples that are in class -1 and 85'667 which are in class 1. Which means that class -1 corresponds to roughly 65.7% and class 1 to 34.3% of all data. This can sometimes affect performance of some models, but as we will se later it doesn't.

Other? Should we add graphs about the distributions? Boxplots? Violinplots? CorrelationPlot? Pairwise plots/hists?

2.1.2 *Data Pre-processing & Engineering.*

.

Z-score. Since we have such large variances and means we will proceed to using z-score scaling as one of the proposed pre-processing methods. This method is implemented in the *implementation.py* file as *z_normalize()*

.

Z-score.

Interaction terms.

2.1.3 *Feature extraction.*

2.2 Evaluation and Baselines

3 MODELS

3.1 Baselines

3.1.1 Static Model. A simple model that can serve as baseline is one that doesn't need any feature and simply outputs the most represented class in it's training data.

3.1.2 Bayesian Prior?

3.2 Least Squares

3.3 Logistic Regression

4 RESULTS

4.1 Environment,Implementation and Training Details

. We ran these experiments on consumer grade x86 CPUs. All machine learning tools are implemented by using numpy. This allowed us to fix randomness's seed using numpy's np.random.seed. We used the seed 42 all along the experience for to ensure reproducibility of the results.

4.2 Test Results

. In this section we present the results of our proposed models and compare them with the baselines and among them-selfs. The accuracy scores we use to compare the different models is based on Cross Validation using 5-fold validation using the same seed and thus the same folds among models. This allows better generalisation of our results rather than arbitrary taking 1 test set and 1 training set, or worse training on the public leader-board.

5 DISCUSSION

- speak about model performance why? - speak about possible improvements to the model in terms of accuracy and runtime.(For the later one could use different gradient descent methods and modifications (Using weight intertia, adaptive learning rate etc..))

6 CONCLUSION

- Conclude on results, happy? Not Happy? Why? - Discuss possibility of implementing more complex problems but no time to code them? Use libraries?