Getting over 84% in finding the Higgs boson

Baetu Ciprian, Volodymyr Lyublinets, Musuroi Doru

***Abstract*—Facing the challenge to detect the appearance or not of a Higgs boson in a collision of protons, we perform an analysis of the provided features and point out useful observation about the values that are present and the ones that are missing, then we propose a featurization pipeline on the result of which we employ different models as baseline evaluation and one model based on a simple neural network implemented from scratch that performs the best.**

I. Introduction

Throughout the month of the competition we've made gradual progress towards reaching a satisfying result. We start with no featurization and basic models as linear regression, ridge regression and logistic regression. These models prove to have a rather small accuracy on a cross-validation evaluation (see table with results - TODO!!). We then turn our heads towards feature augmentation and add a polynomial basis of degree 2 for each feature. As expected, the accuracy increases as presented in table (TODO!!). Then we dive deeper into data analysis and we try to understand the features and how to combine them. This results in the final feature augmentation pipeline presented later in the report. The results can be seen in the table (TODO!!).

(Insert here argumentation of why we chose the NEURAL NETWORK (Having hit a blocking point related to the feature augmentation, we focus on the model and work on the neural netwrok implementation)).

In most models, in special for neural networks, we have to fight issues like overfitting, underfitting or large training times, that prevents us from effectively using the chosen models. For overfitting we choose to use a regularization parameter and we analyze it's effectiveness later on in the paper. We tackle the large training times by changing the learning method from gradient descent to stochastic gradient descent which proves to be a great improvement.

II. Data analysis

Before employing any model, we proceed to do data imputation and cleaning procedures.

To start with, the provided dataset has a problem with missing values. After basic exploratory analysis, we see that missing values are split into three groups and either the entire group is missing or the entire group is present. These column groups are:

First group

DER_deltaeta_jet_jet, DER_mass_jet_jet, DER_prodeta_jet_jet, DER_lep_eta_centrality, PRI_jet_subleading_pt, PRI_jet_subleading_eta, PRI_jet_subleading_phi

Second group

PRI_jet_leading_pt, PRI_jet_leading_eta, PRI_jet_leading_phi

Column 0

DER_mass_MMC

After reading the documentation **ADD HERE LINK TO BIBLIOGRAPHY**[2], we find that these values are not missing at random, but rather undefined when PRI_jet_num ¡= 1 for the first group of columns, undefined when PRI_jet_num = 0 for the second group of columns and undefined when the event is too far from the expected topology for DER_mass_MMC. Column 0 (DER_mass_MMC) is particularly interesting - if we look at b/s ratio for cases when it is undefined, we see that around 95% have the 'b' value. Thus, this is a very important signal in itself and we should not just discard it.

We consider several strategies for dealing with missing values:

- Replacing them with maximum of the column + 1
- Replacing them with the mean of the column (only defined values)
- Replacing them with the median of the column (only defined values
- Variations of class-based imputation

III. Feature augmentation

Throughout the competition we've tried many variants of the new features and chose the ones that have shown to provide the biggest benefits to our submission model. In the beginning, the process of deciding whether a new set of features was good involved doing a simple cross validation, but when result crossed the 0.84 mark, we started using 5-fold cross-validation to battle variance. The final list of used features is the following:

Squares of the original features values

We do it for linear regression, where adding them shows an improvement from 0.74 to 0.77. This is quite expected, as adding polynomial features increases the representational power of linear models. We then try adding them to a 1 hidden layer NN and the result goes up from X to Y. Neural network never multiples the data with itself, so it's reasonable that squared features are useful for it too. Adding higher power features does not lead to additional improvements for our neural network.

Cosines of pairwise angle differences
: Applying sin(x) or cos(x) only makes sense for angles. We also discovered that doing it for pairwise differences is quite helpful.

Converting PRI_jet_num column using one-hot encoding
: This is the only categorical feature in the dataset with only 4 options, so it makes sense to do one-hot for it

RBF features for columns of identical type
: For columns X and Y that are G(0, 1), we add column with $exp(-||X - Y||^2)$. RBF features, being typically used to extend Support Vector Machines with good results, we considered worth trying to add them to our neural network. This resulted in a minor improvement. Identical type meaning both columns represent mass, centrality, etc.

As a note to the rather concise feature augmentation pipeline, using a neural network allows us to pus less effort into this process compared to using a linear model. In this reason, neural networks are known to have ability to fit complex data shapes as a result of non-linearity, thus sparing us from searching for features that can allow simple regressions to achieve this behaviour.