Report project 1: Higgs challenge

Gregoire Clement, Charles Gallay, Thomas Batschelet
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—Standard procedure done by the CERN scientists in order to detect the Higgs Boson may be significantly improved with the help of Data scientists. Using simulated data, we can increase the selection region procedure and give another perspective in the ad hoc choices of physicists. This report precisely describes a way to build a robust and reliable data model able to detect the Higgs Boson with a good performing matching score.**

I. Introduction

Given a data set of simulated events which represent the image of particles issued by proton-proton collisions. Each event is captured as a vector of fixed length that describes it's type, energy and direction for each particle so produced. The goal is to predict the presence of the Higgs Boson by finding a region separation in the feature space that allows us to classify the particles either as a background event (known processes) or a signal event, trace of a Higgs Boson (label of interest).

We want to show that with a simple, understandable and easy to implement data model, it is feasible to distinguish these two events. We claim that with enough feature engineering we can compete with more complex models.

II. Pre-Processing

A. Data exploration

The exploration of the dataset is primordial to get a first taste of the data. A simple way to do this is to plot the distribution of each features. In doing so, we discover that some features seem to have an exponential scale. Also the distribution of the PHI features (the angles) are uniformly distributed which provide very few information about the class (same distribution for both signal). Hence those features are dropped. From [1] it's known that some features are considered undefined when feature 22 (PRI_jet_num) takes some specific values.

B. Dealing with missing values

The dataset provided contains missing values (encoded as -999), corresponding to non-computed values or values without any sense. The approach taken is replacing them by the most frequent value in each feature. This has the advantage of not giving too much fake information, as well as ensuring that we replace them by possible values, in contrast with a mean strategy.

C. Normalization of data

In order to perform better, the data is normalized right after taking the log of each value, or before the $tanh$ function depending on the data model (see Figure 2). The data is shifted by its mean and divided by its standard deviation. The test set is merged with the training set, which by the law of large number helps us to find a more precise estimation of the mean and the standard deviation of the true distribution. At this step, it is important to remember the mean and standard deviation of each feature, as we need them later on to apply exactly the same transformation to the test set.

III. Feature Engineering

Based on the new X features we have, we create some additional ones by applying $\phi(x)$. This enable our model to still be linear in term of $\phi(x)$ while augmenting the richness of family functions we use to approximate the true model. The function we apply to the base features are the following:

Bias
: In order to have an unbiased estimator we add a constant equal to 1 as a new feature

Polynomial family
: Based on x we create N new features that we compute using the N *basis function* $\phi_i(x) = x^i \ \forall i \ 1 < i \leq N$. We are also doing M other features for the roots $\phi_i(x) = x^{1/i} \ \forall i \ 1 < i \leq M$

Combinatory
: From each pair of features we create a new one by multiplying them together. $\phi(x_i, x_j) = x_i * x_j \ \forall i, j \ i < j$. This helps extracting correlation between pairs of features.
We also create a new feature for the absolute difference $\phi(x_i, x_j) = |x_i - x_j| \ \forall i, j \ i < j$.

Log and log of inverse
: The log function being define in $(0, \infty)$ we shift the data into that range before applying the natural $log \ \phi(x) = log(x - min(X) + 1)$. This help us recover some normal distribution.
We also define $\phi(x) = log(\frac{1}{(x - min(X) + 1)})$ where X is the vector for the feature.

tanh
: Create a new feature that calculate $\phi(x) = tanh(x)$, this is useful as we it map x in $(-1, 1)$ and give less importance to extrema.

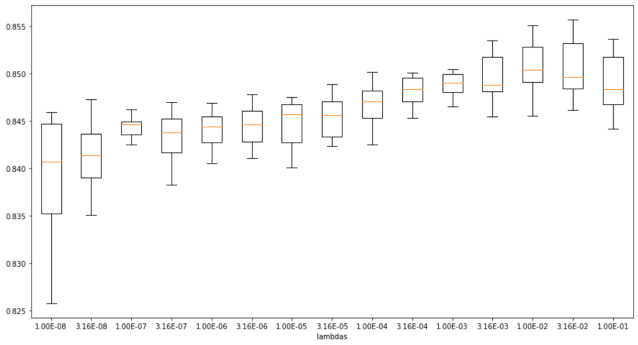

Figure 1. Accuracy of model 3 against λ

IV. Models and Methods

A. Selection

In this section, the whole pipeline is described thoroughly. Figure 2 describe graphically how from raw data we end up with a data model.

After splitting on feature 22, and applying some feature engineering, We train three simple Ridge regressions independent of each other. Although the logistic regression might provide better results, ridge regression gives reasonable accuracy and is much faster.

Each model is trained with a particular degree for (root, tanh, log, log_inv), and a particular polynomial degree which are summarized in table I as well as a specific λ. To find the optimal hyper-parameter, we fix the others and look at how the accuracy improves with respect to changes on that parameter. (see Figure 1). The optimal values are displayed in table I

B. Validation

In order to improve the reliability of the model, cross validations is performed. The training set is split into 4 subsets where 75% of the data is assigned for the training set and 25% for the validation. Figure 1 shows the standard deviation of the score. Local test on the whole training set give an accuracy of 84.15% with a standard deviation of 0.11%

Model	Poly	$Rooti$, $tanh^i$, log^i, inv_log^i	λ
$F22 == 0$	9	3	10^{-3}
$F22 == 1$	11	4	$3.16 * 10^{-3}$
$F22 > 1$	12	3	10^{-2}

Table I
Optimal values found for each model

V. Results

The simplicity of ridge regression allows both a fast training computation and an easy implementation without using external libraries. Our solution involves creating a significant number of features which gradually improves our model to better predict. Even though we have a high quantity of features, we can still understand the whole process rather easily which can't always be achieved using more complex models such as neural network.

VI. Discussion

The richness of our model is by far its key strength to perform with a global accuracy above 84%. A further improvement would be to reduce the number of features, either using PCA, LASSO feature selection or similar techniques. This would help to achieve an even better level of generalization.

References

[1] "Learning to discover: the higgs boson machine learning challenge," 2014.

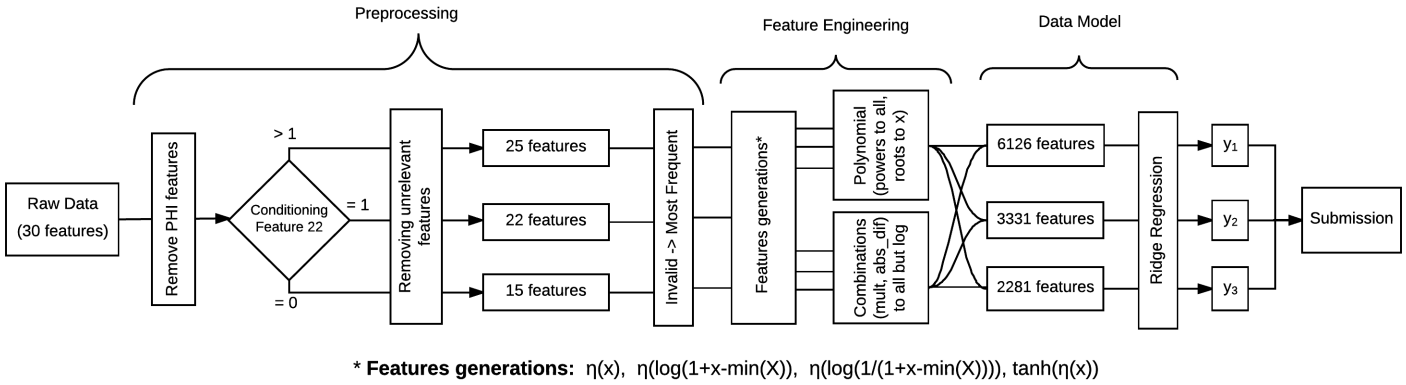

Figure 2. Pipeline of the model: $\eta(x)$ is the function which normalize input x