Machine Learning project 1
-Finding Higgs boson-

Rayan Daod Nathoo, Kopiga Rasiah, and Yann Meier

(Dated: October 28, 2019)

This project aims to explore different methods of data processing and data modelling applied to a popular classifying problem through a trial-and-error approach to obtain the most accurate model.

I. INTRODUCTION

Finding the Higgs boson using data from the CERN is a very popular classifying machine learning challenge. We are given 250'000 data with 30 different features from detectors and sensors which track the energy and momentum of colliding particles in the accelerator. The objective is to determine whether each data sample (with 30 dimensions) indicates that the Higgs boson has been detected or not. A model based on the train data-set is built and then tested on a test set of 568'238 data without the result of the prediction. The online platform AIcrowd assesses the accuracy of the model on this data-set.

II. DATA PROCESSING

A. Pre-processing

An important part of the problem consist of processing the data before building the model. Three different methods were used to process the data:

1. standardization, which consists of reducing all features to the same order of magnitude by subtracting the mean over all data of this feature and dividing by its standard deviation;
2. identifying groups of data according to a specific criterion and build one model per group of data;
3. removing unnecessary features;
4. clipping outliers, replacing outliers above or below a given threshold (expressed in terms of the standard deviation of this feature) by the threshold value.

Two different grouping methods are used. The first one involves identifying the distribution of the missing data that have been replaced by -999. Each data-point is classified into one of the 6 groups according to which feature is missing. The second technique looks at the PRI_jet_num feature which takes the values 0, 1, 2 or 3 and it is noted that other features of the category PRI_jet are correlated to the value of PRI_jet_num. In particular, the distribution of missing data is closely related to the value of this feature. The data is split into four groups and then each group can be split into two subgroups according to the value of the feature DER_mass_MMC, resulting in 8 groups. The feature used to distinguish the group is then removed. One of the drawback of this grouping method is that the allocation of the data is very unequal : the smallest entity only has 1'477 data while the two largest entities include 70'000 data each. Therefore it is expected that the test set also follows the same distribution of the train data-set to avoid building a biased model (see table I).

TABLE I. Frequency of data per group in [%].

Group	0	1	2	3	4	5	6	7
Train set	29.5	10.4	28.0	3.02	19.0	1.18	8.27	0.591
Test set	29.6	10.4	27.8	3.03	19.0	1.19	8.37	0.570

Unnecessary features can be removed from the data set when they do not contribute in making the prediction of the model more accurate. We observed for example that the azimuth angle ϕ [1], which appears in different features, can be removed without affecting the accuracy of the model. On the other hand, outliers can greatly affect the model, more than *common* data do. A way to restrict the effect of outliers is to replace every value which is outside of the interval $\mu_{\text{feat.}} \pm \alpha \cdot \sigma_{\text{feat.}}$ by the value of the bound of this interval closest to the former value of the feature (i.e. clipping outliers). All these methods are coded in the preprocessing.py script.

B. Feature expansion

After pre-processing the data, additional features can be added to increase the complexity of the model. Since the data have a relatively low dimension, numerous ways of expanding the features can be implemented. The first method consists of building a polynomial basis with a given feature:

$$\tilde{\mathbf{x}}^{(n)} = \begin{bmatrix} x_{(n)}^1 & x_{(n)}^2 & \dots & x_{(n)}^d \end{bmatrix}$$

In addition to expanding each feature to some degree, features can also be combined with each others to a certain degree (e.g. for degree two, one gets the following new features : x_1^2, x_1x_2, x_1x_3, etc.). Finally, a whole set of non linear functions can be added to expand the features, such as $\sin x_n$, $\cos x_n$, $\tan x_n$, $\ln x_n$, $\sqrt{x_n}$, etc. These methods are coded in the feature_engineering.py script.

III. MODEL SELECTION

Two methods are used to validate the model. First, an internal check is performed by splitting the data in a

80%-20% ratio within each group. The model is trained on the 80% and then tested on the remaining 20%. These tests generally give a good estimate of the accuracy. The other is to submit the prediction on the test set on the online platform to measure the accuracy. This two ways allows to *empirically* find the best combination.

A. Models

The cost function used for linear models is the mean square error MSE. The ridge regression method consists of adding the term $\lambda\|\mathbf{w}\|_2^2$ with λ the regularizer. The advantage of linear models with MSE is that the exact solution $\mathbf{w}_*$ can be found directly, provided that the $D \times D$ matrix $\mathbf{x}^\top\mathbf{x} + 2N\lambda\mathbf{I}$ is invertible where D is the dimension of the data. Since in our case D is much smaller than the number of data N, it is very likely that this matrix is invertible. Moreover, a strictly positive λ improves the conditioning of the matrix. To choose the adequate value of λ, an analysis of the train and test error with a 50% - 50% split within each group is performed with a varying λ. The curves plotted in figure 1 show that the value that avoids over-fitting (low λ's) as well as under-fitting (high λ's) lies between 10^{-10} and 10^{-3}. The RMSE is here defined as the square root of the weighted average of the MSE within each group.

The feature expansion discussed earlier allows to transform linear models into non-linear models. Another way to model non-linearities is to use the logistic regression method defined in class. An additional term similarly to the ridge regression is added to constrain the norm of $\mathbf{w}$.

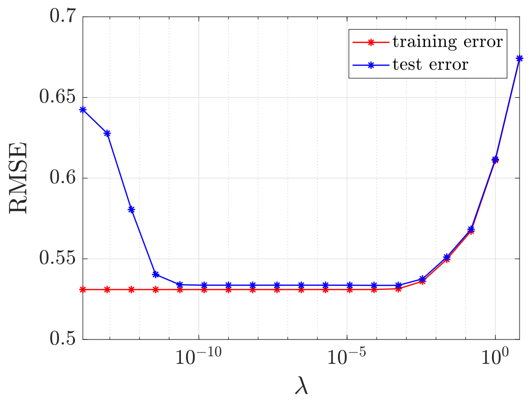

FIG. 1. RMSE of the ridge regression method for the test and train data-set with 50% - 50% split with varying λ.

B. Indirect resolution methods

As the logistic regression has no closed-form for the solution, iterative methods such as the Gradient descent (GD) or the Stochastic gradient descent (SGD) need to be implemented. The new weights vector $\mathbf{w}$ is updated by the opposite of the local gradient of the cost function multiplied by a factor γ that controls the step size. To prevent the method from diverging, it is necessary to use a decreasing γ as the number of iteration increases. The Robbins-Monroe condition gives satisfying results: $\gamma^{(t)} = \frac{1}{(t+1)^r}$ with $0.5 < r < 1$.

Moreover, the stochastic gradient descent is less sensitive to the choice of γ than the regular gradient descent. For the logistic regression, the algorithm of the gradient descent does not converge to a fixed $\mathbf{w}$ but converges towards a vector $\mathbf{w}$ with specific orientation with a norm increasing to infinity. To check how many time steps are necessary, the cosine of the angle between $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$ is plotted, a plateau at 1 indicating that the minimum number of iterations is achieved. According to our computations, 50 iterations are largely sufficient to converge.

IV. VALIDATION OF THE MODEL

As explained in section III, an internal validation, more precisely a 5-fold cross validation, provides an estimation of the accuracy of the model very close to the accuracy obtained on the test data-set (less than 0.1% difference in accuracy) as long as the model is not over-fitting the training data-set.

A. Final model

The final model consists of a combination of the aforementioned methods that gives the best performance in both the local validation test and the test on Alcrowd. The best performance obtained is **83.7%** accuracy with the ridge regression method as detailed in table II. The only non linear function which improves the performance (apart from polynomials) is the sine.

TABLE II. Model parameters giving the best performance.

Pre-processing	- $-phi-$ features removed - group splitting (8 groups) - standardization - outliers clipping at $\pm 1.5 \cdot \sigma_{feat.}$
Feature expansion	- expansion and mult. up to order 2 - adding sine of every feature - add feature column of ones
Model	- ridge regression with $\lambda = 10^{-9}$ - direct resolution by solving lin. syst.

V. CONCLUSION

In this project, different methods of processing and modelling data are explored to find the model giving the best performance according to two testing methods: internal 5-fold cross-validation and assessment of the model on the test data-set. The best result obtained through a trial-and-error approach is achieved with the ridge regression by *clipping outliers* above or under $\pm 1.5 \cdot \sigma_{feat.}$ and splitting data into 8 groups (amongst other settings). The best performance is 83.7% accuracy on the test set.

[1] ATLAS collaboration, Dataset from the atlas higgs boson machine learning challenge 2014, cern open data portal (2014).