Higgs Boson Machine Learning data challenge

Jakob Krause*
(Dated: August 20, 2021)

This paper is a summary of the Higgs *Machine Learning data challenge* as the final project of the course *physics718: Programming in Physics* in the M.Sc. Physics programme at Bonn university. A boosted decision tree is trained and optimized for the AMS score.

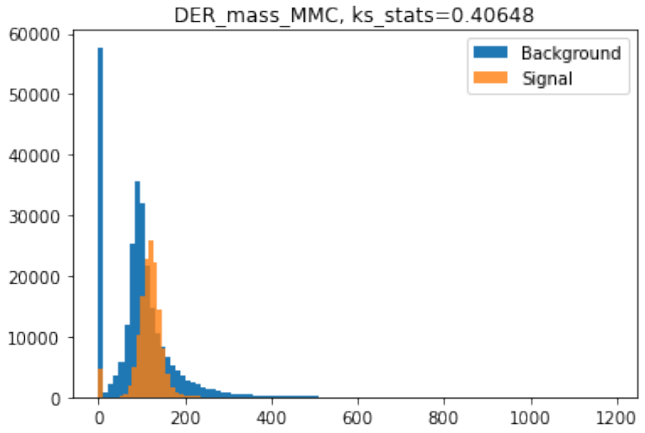
I. PHYSICS MOTIVATION

The Higgs boson is of much interest for many reasons. It is particularly of interest to study the decay of a Higgs into two *fermions*. The study of all modes of decay will increase confidence in the validity of the theory at hand and help characterize the new particle [1]. Specifically in this challenge the $H \to \tau\tau$ channel is studied. This decay is rather rare and suffers from many background contributions (for example $Z \to \tau\tau$) giving the same final state [1]. Therefore it seems a reasonable approach to look at this in terms of a classification problem, signal s or background b, depending on all sorts of detector variables. A machine learning model can then be trained and validated using Monte-Carlo simulations and in the end be applied to measured data.

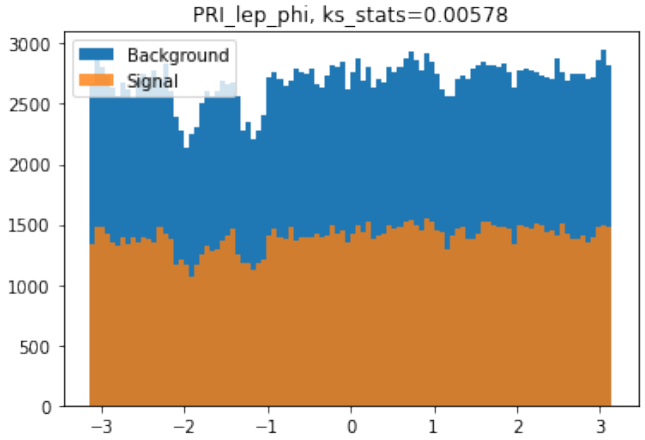
II. INSPECTION OF DATASET

First of all the dataset `train.csv` is loaded into a `pandas` [2] dataframe. There are 30 feature columns and three columns representing an event-id, event weight and event label (signal s or background b), respectively in overall 409119 rows. Upon inspecting the data one finds invalid entries in various quantities of the dataframe. These have a value of -999.0 and will be replaced by 0. Furthermore the weights and 30 input parameters as well as the event label are extracted from the dataframe in 3 arrays `Y` (label), `X` (input parameters) and `w` (weights). Of course the list containing the labels has to be mapped such that each signal event s corresponds to a 1 and each background event b corresponds to a 0. The same procedure regarding the loading and preprocessing of course applies for the test data set `test.csv` as well.

As a next step, the signal and background distributions of all 30 possible input features are analyzed using the Kolmogorov-Smirnov (KS) test which gives insight on how well the particular features can be used to discriminate between background and signal events [3]. Using the dataframe syntax of `pandas` it is easy to obtain a data frame containing only signal events and one only containing background events. Using the method `ks_2samp` of the `scipy` library [4] one can then determine the KS score for each input feature. It is then a trivial task to sort the features in ascending order. Figure 1 shows the best and worst score from the input features. Looking at the plots it becomes clear, that certainly not all input features will be useful in a fit. Also, physical intuition and an inspection by eye would agree with the result of best and worst feature regarding the discrimination between signal and background. Since the figure of merit for this classification problem scales with the sample size (see next section) it is of no use to investigate the performance of a simple machine learning model depending on the size of the training data.



(a) Estimated mass of Higgs boson candidate



(b) Azimuthal angle ϕ of lepton

FIG. 1: Best and worst Kolmogorov-Smirnov scores

* http://www.github.com/krausejm; krause@hiskp.uni-bonn.de

III. MODEL BUILDING AND FEATURE ENGINEERING

The signal and background distribution is highly unbalanced, it is thus useful to employ a more sophisticated measure of performance as usual classification problems. The score that the machine learning model should maximize is the *approximate mean significance* (AMS) and is given by [1]

$$\mathrm{AMS} = \sqrt{2\left((s+b+b_r)\ln\left[1+\frac{s}{b+b_r}\right]-s\right)}, \quad (1)$$

where s and b are the sum of weights of true positive and false positive events respectively, b_r is a regularitory term which is fixed at $b_r = 10$. The best AMS is found by cutting above a certain value in the (probability-)predictions of the model and label all events above this threshold as signal events. In detail this is done by iterating over the quantiles of the range of the prediction (probabilities) and iteratively determining the best cut. The calculation of the AMS requires the knowledge of weights of the test sample as well as a prediction and true values of the test sample. This is implemented in the method `calc_AMS` which returns the best cut value and AMS score.

The chosen approach to this machine learning problem was to employ a *boosted decision tree* (BDT). For this the classifier `AdaBoostClassifier` provided by the `python`-library `scikit-learn` [5] was used. The principle of the *AdaBoost* algorithm is to fit a sequence of weak learners (such as simple decision trees) on repeatedly modified versions of the data. Results of all classifiers are then combined using a weighted sum, where the weights are determined by the errors of the respective classifiers [6].

To calculate the AMS score the weights of the test data have to be known. It was decided to split the training data into 2 sets of actual training and testing data of the same size. The AMS score can then be calculated on the test sample to check the performance. Using a BDT one will have to employ the method `predict_proba` of the `AdaBoostClassifier` after fitting to the training data to obtain a prediction sample which then in turn can be validated.

Employing a out-of-the-box working BDT with a maximum depth of 3 and 50 base estimators (simple decision trees) it proved to be advantageous to remove the 5 features with the lowest Kolmogorov-Smirnov score. This would in fact increase the AMS score, see figure 2. The starting point of improvement is thus an AMS= 1.6.

Having now established a minimal working example one should look at the *hyper-parameters* of the base estimator. For this `scikit-learn` provides the handy tool `GridSearchCV`; this will perform a fit trying to find the best hyper-parameters while optimizing the performance of the decision tree using 5-fold cross-validation to prevent overfitting. Since the performance is measured in terms of successful predictions and not the AMS score this is solely used for the base estimator which then is

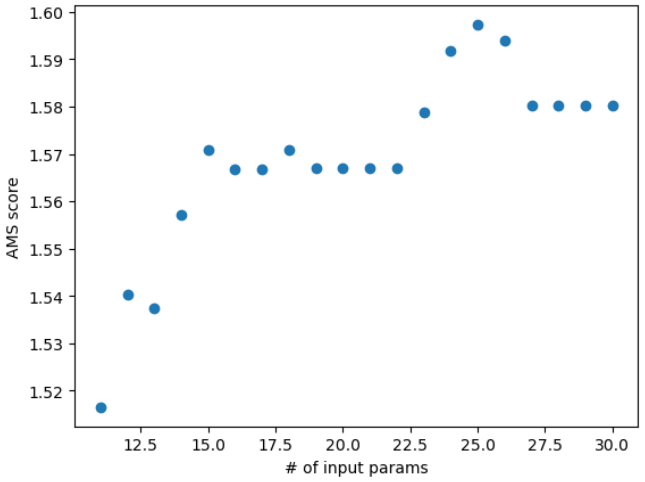


FIG. 2: AMS score depending on number of input parameters

again fed into the `AdaBoostClassifier`. The parameters available for the fit are the number of input features, the maximum depth of the decision tree, and the minimum number of samples at the nodes of the tree to allow splitting or ending of the tree, respectively. As a last tweak the number of base estimators is investigated such as to yield the highest AMS score. Those parameters are found in the described ways and saved by employing the two methods `get_best_params` and `get_best_n`. These are fed into a `AdaBoostClassifier` object one last time to cross check the intermediate results from before. An improvement in the score can be seen; it lies at AMS = 1.8.

To create the submission file an `AdaBoostClassifier` object is trained on the whole training dataset. The cut value which was obtained with half of the whole dataset from before has to be maintained since we have no other way of determining a new cut. Nevertheless the cut value should be roughly the same if we keep the hyperparameters of the classifier the same. The dataframe with the test sample is manipulated such that the same input features as before will be disregarded beforehand. The trained decision tree is then used to make a (probability) prediction of signal or background using the remaining inputs. All those events where the output exceeds the before determined cut value are kept as signal events which are then written to a `.csv` file, identified by their index which was part of the dataframe.

IV. SUMMARY

A boosted decision tree was trained in order to tackle the ATLAS Higgs machine learning challenge. The figure of merit, the AMS score, could be improved from 1.6 to 1.8 on the test sample which was the same size as the training sample after proper model building and feature engineering.

[1] Claire Adam-Bourdarios and Glen Cowan and Cécile Germain and Isabelle Guyon and Balázs Kégl and David Rousseau. The higgs boson machine learning challenge. In *Proceedings of the 2014 International Conference on High-Energy Physics and Machine Learning - Volume 42*, page 19–55. JMLR.org, 2014.

[2] pandas - Python Data Analysis Library. URL `https://pandas.pydata.org/`. last visit: August 20, 2021.

[3] Kolmogorov–Smirnov test. URL `https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test`. last visit: August 20, 2021.

[4] Numpy and Scipy Documentation. URL `https://docs.scipy.org/doc/`. last visit: August 20, 2021.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] 1.11 Ensemble methods - scikit learn 0.24.2 documentation. URL `https://scikit-learn.org/stable/index.html`. last visit: August 20, 2021.