Group 2402 – XGBoost

Andrea Semenzato, Pietro Bernardi, Tomás Mezquita, and Mariam Chokheli
(Dated: April 3, 2024)

In this paper we describe a study of the XGBoost model in the particular case of a binary classification problem. We present a strategy to tune the model manually in order to obtain a simpler version that retained much of the predictive power of a more complex one. We studied the effect of regularization and dataset reduction. Finally, a comparison between XGBoost and other well known classification models is presented.

INTRODUCTION

In this paper we describe a study of the XGBoost[1] model in the particular case of a binary classification problem.

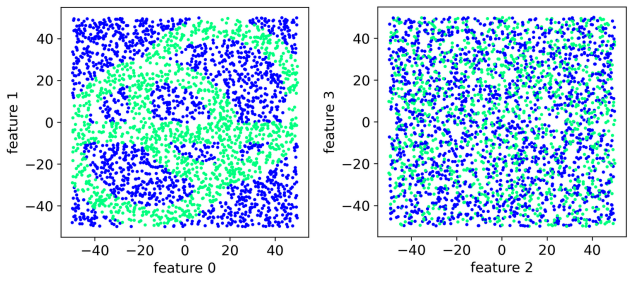

FIG. 1. Dataset with its four features plotted out.

The data is shown in figure 1. As with all machine-learning models, XGBoost offers a wide spectrum of tunable hyperparameters[2] that directly control model's complexity, regularization and boosting behaviour.

The tuning of these hyperparameters is of paramount importance and hence this study was aimed at the following goals:

- Finding a strategy to tune the model manually, leveraging on the model's hyperparameters interactions, in order to obtain a simpler tuned model that could be statistically equivalent to a more complex one.
- Checking whether regularization made an impact on the model performance and complexity.
- Repeating the above described procedure on a reduced dataset that retained only the highest importance-scoring features.
- Comparing XGBoost with other models, also by studying which one performed better when less training data was available.

In the following, we used accuracy as the scoring metric to both evaluate model performance and check that no undesired behaviours occurred (overfitting/underfitting).

METHODS

Manual tuning of XGBoost hyperparameters We partitioned the hyperparameters set into several subsets in order to group them according to their specific role in the model building procedure[3]. The sets were as follows: model complexity (`max_depth`, `min_child_weight` and γ), sampling (`subsample` and `colsample_bylevel,_bynode,_bytree`, regularization (α and λ) and boosting (`n_estimators` and `learning_rate`).

The hyperparameters of a given set were varied simultaneously in their allowed domains[4] firstly using a randomized search to constrain their range in a coarse way and secondly using an exhaustive grid search to fine tune them. In both cases, stratified cross validation[5] was used so that the model, at each iteration, could be tested on a fold that had the same structure of the test set.

Each set was considered sequentially, starting from the model complexity one and ending with the regularization one. Each time, the model was partially tuned with the best parameters searched in the previous sets.

The boosting set was dealt with differently, since the `learning_rate` was tuned first and the `n_estimators` was tuned last. In fact the latter was easily fine-tunable using XGBoost's early stopping feature[5].

Statistical equivalence of different XGBoost models In order to evaluate whether the less complex model was comparable to the more complex one, the following compatibility figure was used:

$$C = \frac{|A_1 - A_2|}{\sqrt{\sigma(A_1)^2 + \sigma(A_2)^2}} \tag{1}$$

where A_i are the two model accuracies and $\sigma(A_i)$ their respective standard deviations. A value of $C \leq 1$ was considered as an indicator of equivalence.

Impact of regularization In order to rule out the possible contributions due to other tuned hyperparameters, we used a non tuned model and only the regularization hyperparameters (including γ as it is a pseudo-regularization parameter) were varied in a manner analogous to the one described earlier. This was repeated for various values of the `learning_rate`.

Dataset reduction We considered the feature importances and we retained only the two highest scoring

Accuracy	Value (Default)	Value (Tuned)
train (mean, cv)	0.931 ± 0.009	0.929 ± 0.010
test (mean, cv)	0.928 ± 0.007	0.922 ± 0.004
train (whole)	0.999	0.990
test (whole)	0.948	0.935

TABLE I. XGB-MAN model accuracies on the full dataset (4 features) using cross validation with 5 splits and re-trained accuracies on whole training and test datasets.

Hyperparameter	Value (Default)	Value (Tuned)
max_depth	6	4
min_child_weight	1	0.5
γ	0	0
subsample	1.0	1.0
colsample_bytree	1.0	1.0
colsample_bynode	1.0	1.0
colsample_bylevel	1.0	1.0
α	0	1
λ	1	1
n_estimators	100	136
learning_rate	0.3	0.32

TABLE II. XGB-MAN model parameters on the full dataset (4 features) using cross validation with 5 splits.

ones. The very same procedures described earlier were then applied to the reduced dataset, checking whether the new model could exhibit better or equivalent performance but with less complexity.

Comparison with other models We considered other models that are commonly used in classification tasks: multilayer perceptron (MLP), support vector machine classifier (SVC), random forest (RF) and gradient boosted decision tree classifier (GBC). We also considered a blindly optimized version of XGBoost (XGB-GS) via a fully automatic hyperparameter search. We compared it with the one we manually tuned (XGB-MAN). We calculated each model's accuracy, speed of execution and memory usage. Each model was also evaluated using only a fraction of the initially available training data. This fraction was varied over several values in order to better assess each model's behaviour.

RESULTS

XGB-MAN Model - Full dataset (4 features) The results for the model's manual tuning using the full dataset are reported in tables I and II. Even if the tuned model's test accuracy is slightly lower, considering that the train and test accuracies compatibilities were as follows:

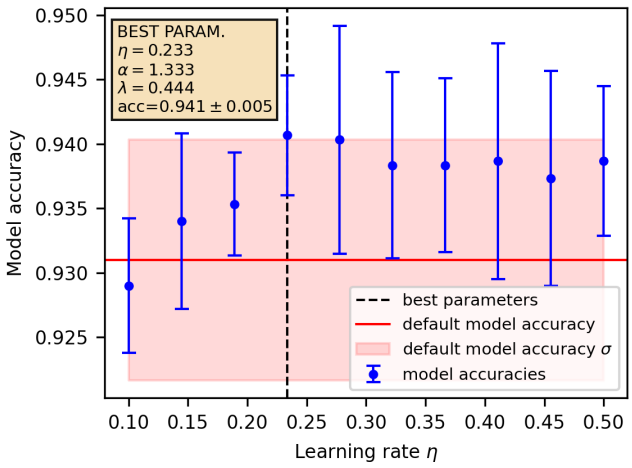

FIG. 2. XGB-MAN model accuracy vs regularization parameters α and λ for various learning_rates on the full dataset.

$$C_{train} = 0.122, \qquad C_{test} = 0.715$$

we concluded that the default and tuned model are equivalent but the tuned model is a much simpler model, being only 4 layers deep with respect to the default one.

Regularization impact was then evaluated firstly by investigating only α and λ hyperparameters and the resulting accuracies are shown in figure 2. The best reported accuracy showed a compatibility of 0.925 with respect to the default model's one.

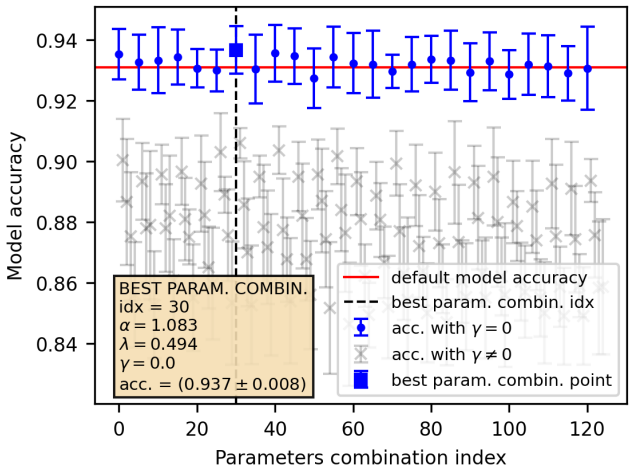

FIG. 3. XGB-MAN model accuracy vs regularization parameters α and λ and γ for various learning_rates on the full dataset.

After γ was added and varied together with α and λ, the results shown in figure 3 were obtained. The grey points in figure 3 are the ones for which $\gamma \neq 0$, so we concluded that any value of gamma greater than zero did nothing but significantly decrease the model's accuracy with respect to the default model's. The compatibility

Accuracy	Value (Default)	Value (Tuned)
train (mean, cv)	0.999 ± 0.007	0.948 ± 0.006
test (mean, cv)	0.947 ± 0.002	0.945 ± 0.006
train (whole)	0.999	0.997
test (whole)	0.955	0.967

TABLE III. XGB-MAN model accuracies on the reduced dataset (2 features) using cross validation with 5 splits and re-trained accuracies on whole reduced training and test datasets.

Hyperparameter	Value (Default)	Value (Tuned)
max_depth	6	4
min_child_weight	1	0.833
γ	0	0
subsample	1.0	0.722
colsample_bytree	1.0	1.0
colsample_bynode	1.0	1.0
colsample_bylevel	1.0	1.0
α	0	0
λ	1	1
n_estimators	100	105
learning_rate	0.3	0.547

TABLE IV. XGB-MAN model parameters on the reduced dataset (2 features) using cross validation with 5 splits.

between the best reported accuracy and the default one was 0.465. We thus concluded that regularization did not seem to play a significant role.

XGB-MAN - Reduced dataset (2 features) The dataset feature importances were scored as follows: $f0 = 0.489, f1 = 0.376, f2 = 0.066, f3 = 0.069$. We thus reduced the dataset retaining only features f0 and f1.

Repeating what has been described above, we obtained the results shown in table III and IV. The train and test compatibilities this time were evaluated as:

$$C_{train} = 0.073, \qquad C_{test} = 0.354$$

So, again, the tuned less complex model is equivalent to the more complex one. Comparing the obtained accuracies with those of the full-dataset trained model, we calculated an accuracy improvement due to reduction given by $\Delta = 0.017 \pm 0.011$ which, given the uncertainty, did not appear to be relevant.

With respect to regularization we observed a behaviour that was identical to the one already reported earlier in the case of the full dataset.

XGB vs Other Models The results of the comparisons are collected in tables V, VI and VII.

The results are also shown in figure 4. These showed how XGB models significantly outperformed the others with respect to speed, while keeping a high accuracy scoring. It was also worth noticing that memory usage was less pronounced when compared to that of other boosting models.

Model	Avg. Accuracy
MLP	0.9433 ± 0.0061
SVM	0.5280 ± 0.0000
RF	0.9334 ± 0.0031
GBC	0.9508 ± 0.0021
XGB-GS	0.931 ± 0.009
XGB-MAN	0.929 ± 0.0010

TABLE V. Average accuracies of the compared models.

Model	Avg. Train. Time (s)
MLP	3.0581 ± 0.2568
SVM	0.1806 ± 0.0295
RF	0.8013 ± 0.0188
GBC	3.6953 ± 0.0721
XGB-GS	0.0546 ± 0.0069
XGB-MAN	0.0371 ± 0.0002

TABLE VI. Average training times of the compared models.

Model	Avg. Memory Usage (MB)
MLP	0.600 ± 0.025
SVM	0.160 ± 0.002
RF	0.405 ± 0.013
GBC	0.558 ± 0.005
XGB-GS	0.288 ± 0.024
XGB-MAN	0.210 ± 0.0020

TABLE VII. Average memory usage of the compared models.

We then investigated the impact of varying the size of the training set on the performance of the above mentioned models.

The results are shown in figure 5.

Analyzing the training accuracy reveals that it surpasses the test accuracy and it is usually close to 1 (i.e. the XGBoost and GradientBoost). The difference between training and test accuracies was at its largest at very small training set fractions, signalling poor performance of the models in this region.

Fluctuations in training accuracy can be attributed to the trade-off between model complexity and overfitting.

We found that test accuracy increases with training set size and tends to approach training accuracy. This stresses the importance of sufficient training data for achieving robust model performance.

To compare the efficacy of different methods, we consider their mean test accuracy across various training sizes, as shown in table VIII.

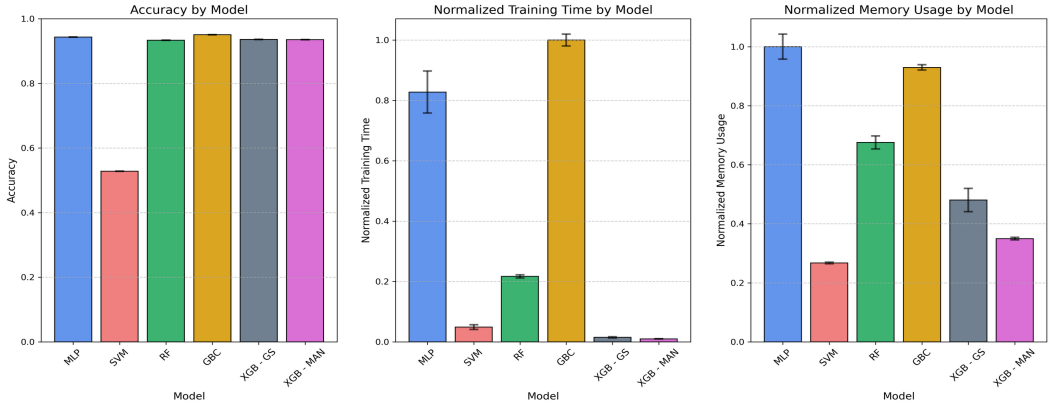

FIG. 4. Model comparison results. Starting from the left, panel (a) shows the accuracies, panel (b) shows the normalized training times and panel (c) shows the normalized memory usage.

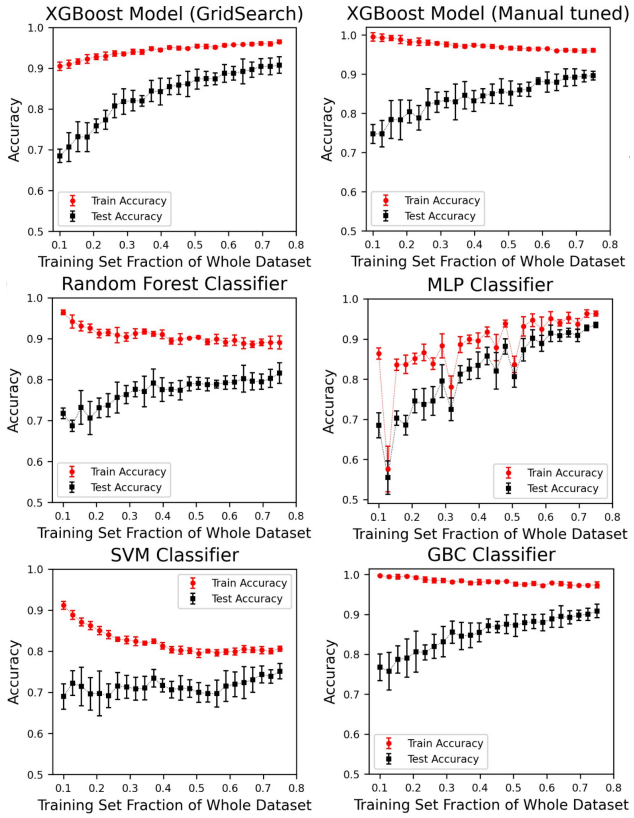

FIG. 5. Training and test accuracies for different training set fraction sizes and models. Train accuracy is shown with red circles while test accuracies with black squares.

Model	Avg. Test Acc.
MLP	0.816 ± 0.095
SVM	0.714 ± 0.016
RF	0.770 ± 0.033
GBC	0.852 ± 0.043
XGB-GS	0.833 ± 0.065
XGB-MAN	0.840 ± 0.043

TABLE VIII. Mean test accuracy of each model across training sizes.

CONCLUSIONS

We found out that a simpler XGBoost model with a depth of only 4 is equivalent to one of depth ≥ 6 when operating on the dataset with all of the four features. Slightly better but not statistically significant improvements in accuracy occurred when considering a reduced dataset (only two features), with a mean increase of 0.017 ± 0.011. Regularization did not make a significant contribution in this case. We demonstrated that training dataset size is of crucial importance in this case and that boosting models exhibit more reliable performance (less variance) at very small training sizes with respect to the others. Out of these, XGB models appeared as the most desirable given their speed and memory utilization.

Group members' contributions Andrea: model comparisons, Pietro: manual tuning and regularization, Tomás: variations of training set, Mariam: manual tuning.

[1] Chen, T. & Guestrin, C. Xgboost: A scalable tree boosting system. *Proceedings Of The 22nd Acm Sigkdd International Conference On Knowledge Discovery And Data Mining*. pp. 785-794 (2016)
[2] XGBoost Documentation. xgboost developers (2024).
[3] Jain, A. (2024, January 7). Mastering XGBoost Parameters Tuning. Analytics Vidhya.
[4] XGBoost Parameters — xgboost 2.0.3 documentation.
[5] Wade, C. and Glynn, K. (2020) Hands-On Gradient Boosting with XGBoost and Scikit-learn. 1st edn. Birmingham: Packt Publishing, Limited.