ChaLearn: Black box for machine learning

Moncef Biaz (UC Berkeley, CIV), Jun Jie Ng, (UC Berkeley, EECS) and Ludovic Thea (UC Berkeley, IEOR)

***Abstract*—This paper describes our implementation of a black box for binary classification problems under time-budget constraints. The purpose of this black box is to remove the human of the loop when building predictive models. This implementation uses the starter kit from the ChaLearn AutoML competition. For each dataset, we automatically choose among 3 different tuned classifiers to predict on a test set using normalized balanced accuracy metric. Our system gives similar results than the winners of the AutoML contests on some datasets, but still has room for improvement for other datasets.**

***Keywords*—Auto-Machine Learning, predictive modeling, black box, ChaLearn.**

I. Introduction

Our project was to design a machine learning black box capable of performing all model selection and hyper-parameter tuning without any human intervention. This problem is important for companies to make data-driven decisions more automatic. Some companies who do not even have data scientists could still leverage on their data using that kind of black box. Our initial intent was to enter the ChaLearn AutoML challenge. However, the competition was already at a too advanced stage for us when we started this project. Since we could not enter the competition for the first rounds, we chose to build our own black box by following as closely as possible the constraints of the competition, but without entering it.

The data sources are already provided in the competition so we were restricted to using them. There are currently 5 open rounds (numbered 0 to 4) for the competition, each corresponding to different levels of problems and difficulty. For this project, we focused on the Phase 1 round, which corresponds to **binary classification using dense data**.

II. Challenges

Given a dataset of instances X_{train} and the associated labels y_{train}, we want to predict the labels $\hat{y}_{test}$ associated with the dataset X_{test}.

Doing so usually requires different steps: exploring the data, preprocessing it, engineering or selecting the features, choosing a model, tuning its hyperparameters, and validate it on a validation set. Our goal is to design a black box that goes through all these steps without any human intervention. The difficulty of the problem is that there are many different possibilities for feature selection and parameter tuning. And testing all the possibilities take a lot of time.

Although we did not enter the competition, we still tried to build a black box that would had been ready to enter the competition if the previous rounds were still opened. In particular, the competition includes a **time budget constraint for each dataset**. Within this time budget limit, the black box has to load the data, preprocess it, select the features, select a model and tune the hyperparameters. This constraint makes this problem even more challenging than it already is.

Because we could not enter the competition, we could not compare our predictions using the test set that was given to us. Therefore, we split the training sets of the competition into a smaller training set and a new test set, so that we could assess the quality of our final predictions.

III. About the data

The data for Phase 1 only contains features that are numerical. The **metric** used for assessing the quality of the prediction is the **normalized balanced accuracy**. There are no missing values for all five datasets.

TABLE I. Statistics of the data for Phase 1

Name	Christine	Jasmine	Madeline	Philippine	Sylvine
Time budget	20 min	20 min	20 min	20 min	20 min
Number of features	1636	144	259	308	20
Dataset size	5418	2984	3140	5832	5124

IV. Building the black box

To build the black box, we used the starter kit that was provided by ChaLearn. This gave us insights on how to actually build the black box, although we recoded some of the modules from scratch. The starter kit provided a nice pipeline, but the actual data handling was simple - the missing values were systematically replaced by 0's. There was not handling of categorical variables. The first 1,000 features were selected, regardless of their significance. Depending on the type of data (sparse or dense), a specific classifier or regressor was chosen. The number of estimators used in those classifiers or regressors kept increasing to train better classifiers as long as there was still time budget left.

A. Data preprocessing

We modified the DataManager and DataConverter modules from the starting kit so that it the data would be preprocessed during the loading process. For the numerical features, we replaced the missing values by the mean value of each feature. We then normalize the numerical features by retrieving the mean and dividing by the standard deviation. For the categorical values, we replace the missing values by the most frequent category. The categorical features are then one-hot encoded.

B. Feature selection

Normally, feature selection should be done as part of a pipeline. However, here we did it as a preprocessing for efficiency reason. Because the dataset can contain many features (eg, the Christine dataset contains 1,636 features), and because we have a time budget constraint, we cannot have a too large feature space. Therefore, we systematically performed *Principal Component Analysis* if the number of features was greater than 100.

C. Model selection

We selected 3 different approaches for binary classification: **Discriminant Analysis**, **Neural Networks** and **Random Forests**. We thought originally that we would use those three classifiers, and see if blending the results using a majority vote would give us better results. However, we found out that ensembling the classifiers would be more expensive than we originally thought it would be. Indeed, to determine if the ensemble classifier is better than the weak classifiers, we need to perform a cross-validation on the ensemble classifier as well, so that we get the same metric to assess the quality of the solution. But performing a 10-fold cross-validation on the ensemble classifier means training 30 classifiers. Even though we had implemented the ensembling, we preferred to use our time budget to conduct a more extensive search for the hyperparameters.

D. Model tuning

Tuning the models was a tricky task because of the time budget constraint. The classical approach to tune the hyperparameters is to find the optimal set of hyperparameters using cross-validation. However, because of the time budget, we cannot conduct a thorough grid search for every model.

Therefore, we implemented a search for the hyperparameters that would fit into a time-budget constraint by defining cycles of hyperparameter tuning. A new cycle is launched if and only if there is still some time budget left. For each cycle, we would test the three different types of model for a certain set of hyperparameters, and do a 10-fold cross-validation. We keep in memory the best classifier built so far based on the cross-validation scores. The hyperparameters used during the cycles are picked differently depending on the type of model:

- For Random Forests, the tuning parameter is the *number of trees*, which grows at every cycle.

- For Discriminant Analysis, we compare at each cycle the best classifier between LDA and QDA. The hyperparameter for LDA is the *number of components*, whereas the hyperparameter for QDA is the *regularization parameter*. For LDA and QDA, the value of hyperparameter that is tested is picked at random and is different at each cycle. This approach is sometimes more efficient than trials on a grid [1], and fits better our time budget constraints.

- For Neural Networks, we used several hyperparameters that we tune successively: *learning rate*, *batch size*, *activation function* (sigmoid or tanh). We tuned those hyperparameters one by one and in this order. We acknowledge that the error function is not a convex function in the hyperparameters. Therefore, we have no guarantee that this heuristic gives us the optimal set of hyperparameters. We chose to adopt this strategy nonetheless because of the time-budget constraint.

TABLE II. RESULTS FOR PHASE 1 (BAC: NORMALIZED BALANCED ACCURACY)

Name	Christine	Jasmine	Madeline	Philippine	Sylvine
BAC Neural Networks	.1780	.3503	.0896	.2566	.0321
BAC Random Forest	.4071	.5908	.4124	.4331	.8695
BAC Discriminant Analysis	.4397	.5596	.1831	.4263	.8148
BAC of best on test set	.4659	.6043	.3680	.4805	.8637

TABLE III. RESULTS FOR PHASE 1 OF THE WINNERS OF THE AUTOML CHALLENGE

Position	Christine	Jasmine	Madeline	Philippine	Sylvine
1st	..4964	.6160	.8119	.6959	.8989
2nd	.4820	.6198	.6446	.5232	.8903
3rd	.4772	.6198	.6446	.5232	.8903
4th	.4293	.5627	.5274	.5111	.8895

V. RESULTS

Table III presents the normalized balanced accuracies (BAC) for cross validation for each type of model, and the BAC of the predictions on the test set using the best model.

As we can see, the Neural Networks have a rather poor cross-validation BAC. This can be explained by the fact that the neural nets did not have enough training epochs. Indeed, we have limited the number of epochs to 5 in order not to consume all our time budget on training one neural net. Relaxing the limitation on the maximum number of epochs should give better results, but it would not allow us to test as many different hyperparameters for each model.

We can also notice that the cross-validation BAC is generally better when using Random Forests. The difference with the Discriminant Analysis approach can sometimes be quite significant (see madeline dataset). This result illustrates the power of ensemble methods.

We compared our results to the top scores of the Phase 1 ChaLearn competition, (Table III) which was already closed by the time we started this project. We performed reasonably well on the datasets Christine, Jasmine and Sylvine, with similar accuracies on a test set, but rather poorly on the other datasets. More specifically, for Christine, we fell short of the 3rd place by 0.01 points. For Jasmine, we did worse than the 3rd place by 0.016 points. However, we need to keep in mind that the test sets are different, as we constructed our own test set from the original training set in order to get a measure of the quality of our results.

VI. CONCLUSION

We have built a black box for binary classification under time constraints. Using the ChaLearn competition as a starting point, we have designed a pipeline that goes from loading and preprocessing the data to predicting the labels using a tuned model. Our results are comparable to the scores of the Phase 1 ChaLearn competition winners for the dataset Christine, but performs poorly on the Philippine and Madeline datasets.

The first key take-away of this project is that Neural Networks may be not adapted to problems with strong time-budget constraints. The second key-take away is that ensembling methods can be still be efficient in this context.

Possibilities of improvements for the existing system include testing other classifiers (such as SVMs, or logistic regression) or using other types of data preprocessing depending on the type of the data that can be explored during a meta-learning phase prior to the preprocessing step. The next steps also include extending the scope of the problem by considering sparse data and multi-label classifications or regressions.

Appendix A
Metric for evaluation

Balanced accuracy is the average of class-wise accuracy for classication problems— and the average of sensitivity (true positive rate) and specicity (true negative rate) for binary classication. For binary classication problems, the class-wise accuracy is the fraction of correct class predictions when q_i is thresholded at 0.5, for each class. For multi-label problems, the class-wise accuracy is averaged over all classes.For multi-class problems, the predictions are binarized by selecting the class with maximum prediction value $argmax_l\ q_{il}$ before computing the class-wise accuracy

We normalize the metric as follows;

$$|BAC| = (BAC - R)/(1 - R)$$

where R is the expected value of BAC for random predictions (*i.e.*, R = 0.5 for binary classification and R = (1/C) for C-class problems. [2]

References

[1] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. The Journal of Machine Learning Research, 13(1), 281-305.

[2] Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Ho, T. K., ... & Viegas, E. AutoML Challenge 2015: Design and First Results.