Basic Hyperparameters Tuning Methods for Classification Algorithms

Claudia ANTAL-VAIDA
The Bucharest University of Economic Studies, Bucharest, Romania
claudia_antal@ymail.com

Considering the dynamics of the economic environment and the amount of data generated every second, the decision-making process is changing and becomes data driven, highly influencing the business strategies setup in order to keep the competitive advantage. However, without technology, data analysis would not be feasible, reason why machine learning is seen as a disruptive innovation for businesses, especially due to its capacity to convert data into actionable outcomes. Though, for a high-quality machine learning model result, algorithm selection and hyperparameters optimization play vital roles, hence became high-interest topics in the field. To achieve this, various automatic selection methods have been proposed and the aim of this paper is to compare two of them – GridSearch and RandomizedSearch - and assess their impact on the model accuracy by comparing with the results obtained when default hyperparameters were applied.
Keywords: *Hyperparameters, Tuning, GridSearch, RandomizedSearch, Classification*
DOI: 10.24818/issn14531305/25.2.2021.06

1 Introduction

Machine learning has a wide applicability in the financial field, being able to support and improve fraud prevention, credit assessment, risk management, product customization and not only. The adoption and implementation of these techniques have proven to increase efficiency by faster performing routine operations, enhance credit assessment and risk minimization by establishing the credit worthiness of a customer and predict potential future behavior based on his financial history, improve customer relationship and increase retention by offering products tailored to the customer's needs, and offer better mechanisms for fraud detections thru pattern recognition [1]. Despite the potential and predicted benefits of these technologies, a survey which targeted data professionals around the worlds outlined that only 45% of the companies were already using Machine Learning, while 21% said their companies were still exploring the technology [2]. Most probably, the main reason for the low adoption rates are the costs implied, the difficulty to justify the investments and assess the return of investment, but also due to the overall risk of failure for projects to implement emerging technologies, which is higher than in case of traditional technology projects [3].

One of the key questions in the financial area is if a customer will meet his obligations to the bank, problem which can be analyzed and solved thru classification algorithms, by predicting his potential behavior based on historical records and financial indicators. For this purpose, various machine learning algorithms, such as Linear Regression, K-Nearest Neighbor, Decision Trees, Random Forest, XGBoost or Neural Networks, were proposed and applied over time, outperforming the traditional statistical methods [4]. The software packages for machine learning available on the market have pre-defined libraries to perform predictions which can obtain good result with the default parameters, however, the key questions to answer when building a model is what algorithm to choose and how to optimize it for better results and a higher accuracy.

A Machine Learning model has hyperparameters which can take various values to customize the model architecture and control its learning process on a specific dataset, playing an important role in the accuracy of the output. Although the impact of the hyperparameter values is known, the challenge comes with setting up the best combination in order to reach the best performance of a model on a given dataset.

The hyperparameter tuning is not a new topic,

but dates back in the 90s ([5], [6], [7]), and since then, it was recognized that different combinations of hyperparameters need to be tailored to the dataset for better results [6]. Hyperparameter optimization has several important use cases, worth mentioning being the following ones [8]:

- Considerably reduces the human effort to identify the best combination of hyperparameters for the best performance;
- Improves the performance of the algorithm, by customizing it to the given dataset;
- Improves the reproducibility and facilitates comparison between the models.

As the Machine Learning usage in companies is increasing, hyperparameter optimization plays a bigger role there with a commercial substantial usage, though there are various challenges faced in real-life problems [8]:

- For large models or datasets, the functions evaluation can be very expensive;
- The complexity of the configurations and the high-dimensional space of hyperparameter's values makes it difficult to decide which are the one which should be optimized and within which ranges;
- No straightforward way of optimization for generalized performance.

2 Hyperparameters Tuning

One of the biggest challenges in the Machine Learning field is model selection and configuration, given the wide range of possibilities which are applicable. Moreover, the unavailability of a mapping between machine learning algorithms and problems to solve makes it even more difficult, reason why controlled experiments are required to assess what works best for a given dataset.

A Machine Learning algorithm have 2 types of variables: hyperparameters and parameters. If model parameters represent properties of the training data learnt by the model during the training process, required for making predictions, model hyperparameters dictate the behavior of the model during the training time and are configured before model training even begins [9].

Given a supervised machine learning problem such as predicting if a customer will pay his credit or not, a researcher may build a model in a manual and iterative way: first selects one or more algorithm either based on experience, trial-and-error or based on the literature recommendations, second, determines the values of the hyperparameters, followed by the training, testing and assessment of the models. If the model does not provide satisfactory results, the researcher can manually adjust the hyperparameters and repeat the training and testing steps until achieving the expected results. Even though this task can be hardly achieved manually, thru exploring various combinations and comparing the output, there are techniques which can support this task in an automated manner, and this is referred in the domain literature as hyperparameter tuning or optimization. There are libraries in Python which can be called to perform such a task and they result in a single set of well-performing hyperparameters which can be used for the model configuration [8].

There are various techniques which can achieve this task, the most basic ones being Grid Search and Random Search, while more advanced techniques such as Bayesian Optimization and Evolutionary Optimization are also available. This paper focuses on the Grid Search and Random Search techniques and aims to assess their impact on classification algorithms, by comparing the outputs when default hyperparameters are considered with the ones obtained after hyperparameters tuning is performed.

Grid search is a basic hyperparameter tuning method which builds models for the cartesian product of the values provided, evaluates them and selects the architecture which results in the best performance. Although this method is used for automatic tuning, the efficiency of the algorithm rapidly decreases when the range of hyperparameters being tuned increases, resulting in expensive processing costs.

Random search, initially presented by Bergstra and Bengio in 2012 [10], behaves in a similar manner as Grid search, assessing the models with different combinations of values, but instead of performing an exhaustive

search, it rather requires a statistical distribution for each hyperparameter which are randomly sampled. Comparing with Grid Search, this method is more efficient in a high-dimensional space as not all hyperparameters are equally important to optimize [11].

Grid Search performs well for combinations that are known to perform well, while Random Search helps to discover hyperparameters combinations that are not intuitive. The later is known to be faster than the first one as it does not test the full range of possibilities and it does also reduce the chances of model overfitting to the training data [10].

Considering that the identification of the right hyper-parameters is very important in the success of a model (average change being of 46% [12]), the purpose of this paper is to assess the results of 6 widely used algorithms when default hyperparameters are applied and identify ways to improve the outcome by tuning the hyperparameters.

3 Experimental results

In this paper we will apply the Grid Search and Random Search techniques for hyper-parameters tuning on the top machine learning algorithms which are used for classification problems. The classification algorithms considered are Logistic Regression, K Nearest Neighbours, Decision Trees, Random Forest, Neural Network and XGBoost.

The dataset used for this research was posted by I-Cheng Yeh in UCI Machine Learning Data repository [13], and contains 30,000 observations and 25 columns (one identifier, 23 dependent variable and a dependent variable), described in **Table 1**.

Table 1. Dataset description

Attribute	**Description**	**Type**
ID	Unique identifier of the client	Categorical (Nominal)
LIMIT_BAL	Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.	Numeric
SEX	Genger (1=male; 2=female)	Binary
EDUCATION	Education (1= graduate school; 2 = university; 3=high school; 4=others; 5=unknown)	Categorical
MARRIAGE	Marital status (1=married; 2=single; 3=other)	Categorical
AGE	Age	Numeric
PAY_1	The repayment status in September 2005 (-1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.)	Categorical
PAY_2	The repayment status in August 2005	Categorical
PAY_3	The repayment status in July 2005	Categorical
PAY_4	The repayment status in June 2005	Categorical
PAY_5	The repayment status in May 2005	Categorical
PAY_6	The repayment status in April 2005	Categorical
BILL_AMT1	Amount of bill statement in September, 2005	Continuous
BILL_AMT2	Amount of bill statement in August, 2005	Continuous
BILL_AMT3	Amount of bill statement in July, 2005	Continuous
BILL_AMT4	Amount of bill statement in June, 2005	Continuous
BILL_AMT5	Amount of bill statement in May, 2005	Continuous
BILL_AMT6	Amount of bill statement in April, 2005	Continuous

PAY_AMT1	Amount paid in September, 2005	Continuous
PAY_AMT2	Amount paid in August, 2005	Continuous
PAY_AMT3	Amount paid in July, 2005	Continuous
PAY_AMT4	Amount paid in June, 2005	Continuous
PAY_AMT5	Amount paid in May, 2005	Continuous
PAY_AMT6	Amount paid in April, 2005	Continuous
default.payment.next.month	Will pay next month? (1= yes; 0= no)	Binary (Categorical)

Before running the algorithms, the dataset was split into a training set and a test set with traint_test_split function from the Python sklearn library [14] and it resulted in two datasets: a training one with 20 100 observations, representing 67% of the total number, and a testing one with 9 900, representing 33% of the initial dataset.

All the steps were performed in Phython, leveraging the set of libraries and functions available [14].

2.1. Logistic Regression

Logistic Regression, a type of linear regression, is a supervised learning classification method used to predict the probability of a target variable. The dependent variables can only result in two classes, meaning it is binary in nature [15].

For assessing the impact of the Hyperparameter optimization, the algorithm was initially running with the default hyperparameters of the LogisticRegression function from skleran library in Python [14]. After that, the GridSearch and RandomizedSerach functions were called to identify the optimal combination of hyperparameters and 2 more runs were performed. The confusion matrixes for the 3 are presented in **Figure 1**, while the optimized hyperparameters are presented in **Table 2**.

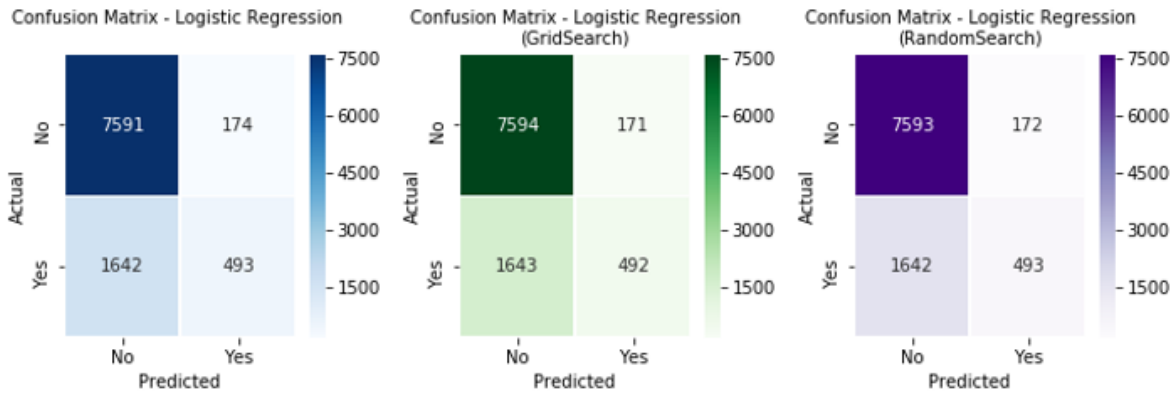

Fig. 1. Confusion Matrixes for Logistic Regression

Table 2 Optimal hyperparameters and accuracies for Logistic Regression algorithms

	Hyperparameters	**Accuracy**
Default	{C=1.0, solver='lbfgs', random_state=None, max_iter=100, penalty='l2'}	81.66%
Grid Search	{C=0.5, solver='liblinear', random_state= 0,max_iter=100, 'penalty': 'l1'}	81.68%
Randomized Search	{C=0.25, solver='sag', random_state=0, max_iter=100, penalty='l2'}	81.68%

A minor improvement was observed between the default version and the optimized ones, but when it comes to the hyperparameter combinations obtained thru the tunning methods, the

two obtained very similar results, the only difference being the categorization: the GridSearch hyperparameters correctly identified more True Negatives than the RandomizedSearch pair.

2.2. K Nearest Neighbor

K Nearest – Neighbor is a nonparametric classifier which estimated the probability of a data point to pertain to a group, based on the distance between the two [16]. It learns from the similarities between classes and once a new record is added to the model, it compares it with the nearest neighbors and adds it to the most similar class.

For assessing its accuracy, the algorithm was initially running with the default hyperparameters of the KNeighborsClassifier function from sklearn library in Python [14], followed by two more runs based on the optimized combination of hyperparameters obtained with Grid Search and Random Search. The confusion Matrixes of the 3 models are presented in **Figure 2**, while the accuracies and the optimal parameters are presented in **Table 3**.

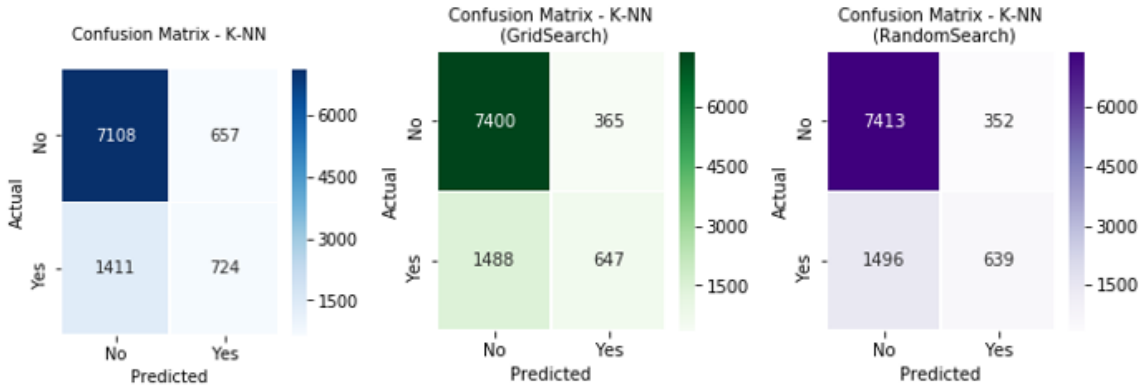

Fig. 2. Confusion Matrixes for K Nearest Neighbor

Table 3. Optimal hyperparameters and accuracies for K Nearest Neighbor algorithms

	Hyperparameters	**Accuracy**
Default	{metric='minkowski', n_neighbors=5, weights='uniform'}	79.11%
Grid Search	{metric='euclidean', n_neighbors=25, 'weights': 'distance'}	81.28%
Randomized Search	{'metric': 'minkowski', n_neighbors=29}	81.33%

If for the default parameters, the accuracy was of 79.11%, we observe an increase in accuracy when applying the optimized parameters: based on the Grid Search combination of hyperparameters, the accuracy obtained was 81.28%, while for the Randomized Search one, the accuracy was even higher, of 81.33%.

2.3. Decision Trees

Decision Trees is a non-parametric supervised machine learning algorithm where the datapoints are continuously split based on different criteria; the branches represent combination of features which lead to the resulting classes (leaf nodes) [17].

In order to assess the impact of the Hyperparameter optimization, the DecisionTreeClassifier from sklearn [14] was run with the default values, while for the next two runs, the same function was applied, but with the optimal combination obtained thru the Grid and Randomized methods. The output is presented in **Figure 3**, while the accuracies and obtained combinations of hyperparameters are capture in **Table 4**.

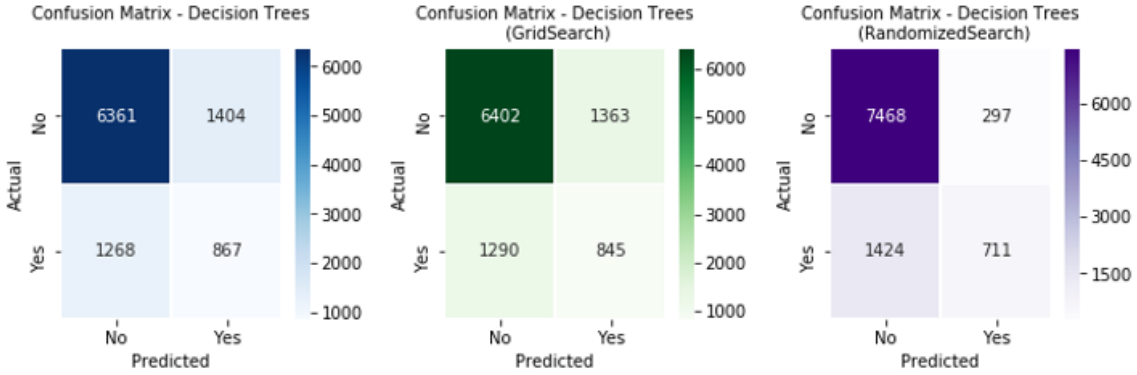

Fig. 3. Confusion Matrixes for Decision Trees

Table 4. Optimal hyperparameters and accuracies for Decision Trees algorithms

	Hyperparameters	**Accuracy**
Default	{criterion='gini', splitter='best', max_depth=None, min_samples_split=2, max_features=None}	73.09%
Grid Search	{criterion='entropy', max_features= 'log2', splitter= 'best'}	73.20%
Randomized Search	{criterion='entropy', splitter= 'best', min_samples_split= 12, max_depth= 2}	82.62%

If the Grid Search hyperparameter combination did not improve considerably the output, the results obtained with Randomize Search brought an improvement of ~9.5pp compared to previously obtained results. In this case, the later optimization method had a considerable impact on the accuracy of the model, increasing the number of True Negative identified cases by more than 1 000.

2.4. Random forest

Random forest is a widely used technique for classification problems which was initially proposed by Breiman in 2001 [18]. It is an ensemble classifier based on bootstrap followed by aggregation which uses a combination of decision trees trained in parallel for several times and its output is built upon the majority decisions of the trees. For our research, we used the RandomForestClassifier from sklearn [14] and the output of the three runs are presented in **Figure 4**, while the hyperparameter's values are consolidated in **Table 5**.

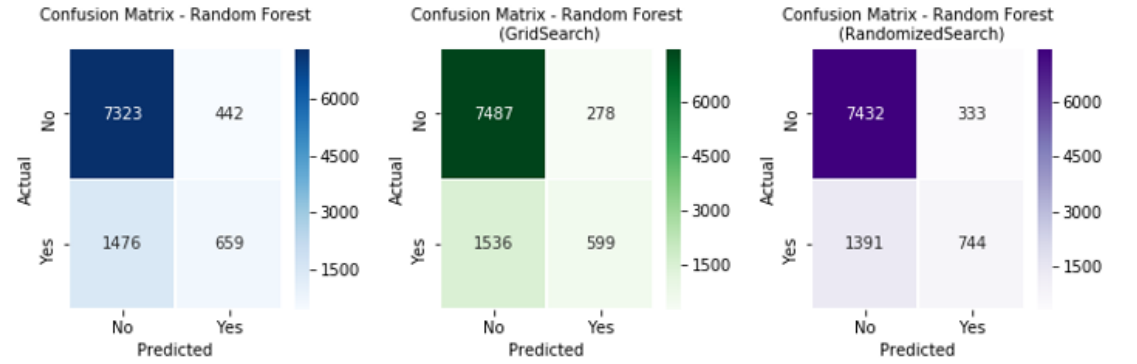

Fig. 4. Confusion Matrixes for Random Forest

Table 5. Optimal hyperparameters and accuracies for Random Forest algorithms

	Hyperparameters	**Accuracy**
Default	{n_estimators= 100, max_depth=None}	80.63%
Grid Search	{n_estimators= 40, max_depth=5}	81.68%
Randomized Search	{n_estimators= 410, max_depth=7}	82.59%

The accuracies from **Table 5** prove that better combinations than the defaults can be applied to improve the models. Based on the combination obtained thru Grid Search, the accuracy of the model increased by almost 1pp, while thru the pair obtained thru RandomizedSearch, the accuracy was higher with almost 2pp. Though, the second combination better predicted the number of True Negatives, meaning that the algorithm correctly identified the cases when a customer did not actually pay his debts. In such a case, getting an accurate prediction for the True Negative would minimize the risk of loss.

2.5. Neural Network

Neural Networks or Artificial Neural Networks is a learning system inspired by human biology and the way the neurons of the human brain function together. It uses a network function to understand and translate the input into the desired output [19] and proved to perform well in identifying complex patterns and make predictions for new records fed to the mode([20], [4], [21]). For the current research, the neural_network from sklearn was used in Phython [14] and the results of the analysis are summarized in **Figure 5,** presenting the confusion matrixes obtained for the three runs, and **Table 6,** which summarizes the hyperparameters used.

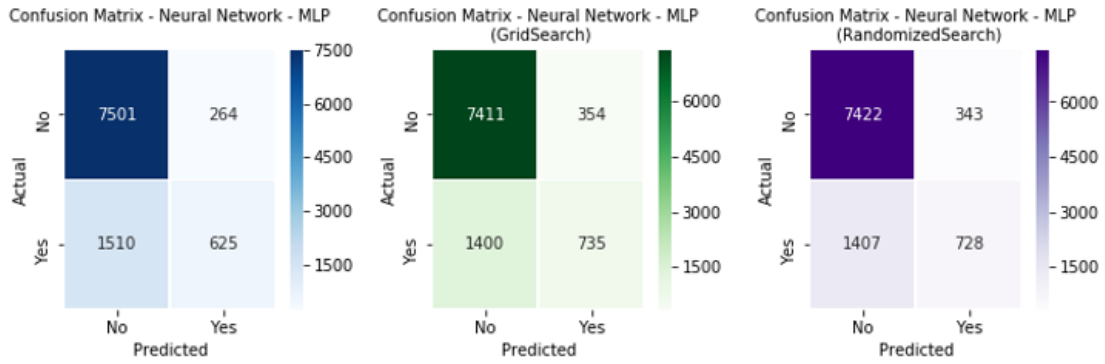

Fig. 5. Confusion Matrixes for Neural Network – MLP

Table 6. Optimal hyperparameters and accuracies for Neural Network - MLP algorithms

	Hyperparameters	**Accuracy**
Default	{activation= 'relu', hidden_layer_sizes= '100', solver='adam'}	82.08%
Grid Search	{activation= 'tanh', hidden_layer_sizes= (20, 2), solver='adam'}	82.28%
Randomized Search	{activation= 'relu'}	82.32%

As presented in **Table 6**, with the default parameters, the algorithm obtained an accuracy of 82.08%. For the optimized algorithms based on Grid Search and Random Search, we noticed a slightly increased accuracy, of 82.28% and 82.32% from 82.08%.

2.6. XGBoost

XGBoost stands for extreme gradient boosting and is an implementation of gradient

boosted decision trees designed for speed and performance, introduced in 2016 by Tianqi Chen [22]. This method scales beyond billions of examples using less resources than the existing systems, focusing on computational speed and model performance. For the current paper, we used the xgboost library in Phyton [23]. **Figure 6** highlights the confusion matrixes obtained for this algorithm, while **Table 7** presents the pairs of hyperparameters applied.

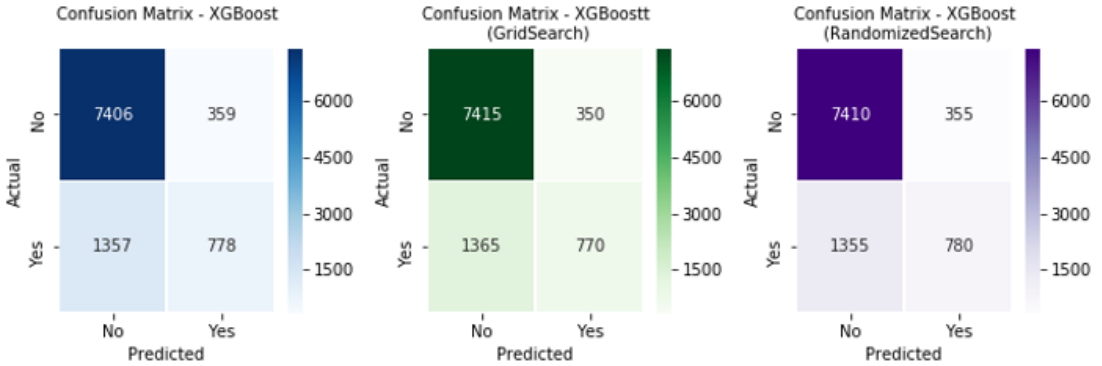

Fig. 6. Confusion Matrixes for XGBoost

Table 7. Optimal hyperparameters and accuracies for XGBoost algorithms

	Hyperparameters	**Accuracy**
Default	{booster= 'gbtree', n_estimators= '100', learning_rate= '0.1', gamma= 0}	82.67%
Grid Search	{booster='gbtree', n_estimators= 10}	82.68%
Randomized Search	{booster='gbtree',n_estimators= 380, learning_rate= 0.2, gamma= 15}	82.73%

This algorithm obtained a good accuracy even with the default parameters, proving it is an optimized library. More than that, the time required to perform the processing was considerably lower than the one required for other algorithms. When it comes to the hyperparameter tuning, the accuracy slightly enhanced based on the combinations obtained with Grid and Random Search, but the accuracy obtained

All the results of the current analysis are consolidated in **Table 8**.

Table 8. Consolidated results

Algorithm	**Accuracy (default)**	**Accuracy (GS)**	**Delta GS vs Default**	**Accuracy (RS)**	**Delta RS vs Default**
Logistic Regression	81.66%	81.68%	0.02pp	81.68%	0.02pp
K Nearest Neighbor	79.11%	81.28%	2.17pp	81.33%	2.22pp
Decision Trees	73.09%	73.20%	0.11pp	82.62%	9.53pp
Random Forest	80.63%	81.68%	1.05pp	82.59%	1.96pp
Neural Network	82.08%	82.28%	0.2pp	82.32%	0.24pp
XGBoost	82.67%	82.68%	0.01pp	82.73%	0.06pp

For all the analyzed algorithms we noticed improvements when running the algorithm with the optimized combination of hyperparameters, however, the increase is not noticeable in most of the cases. The only three algorithms where the accuracy improved by

more than 1 percentual point are K Nearest Neighbor (vs GS: 2.17pp, vs RS: 2.22pp), Decision Trees, where the optimization revealed a more effective parameter combination, increasing the accuracy by 9.53 pp when RandomizedSearch hyperparameters were used, and Random forest, where the accuracy increase by 1.96 pp with Randomized Search hyperparameters.

Although it was not the purpose of the current study, another conclusion we can draw is that XGboost outperformed the other algorithms for the given dataset, even with default parameters.

To conclude, even though these methods can improve the accuracy of the models, we observed an impact on the run time, which, in most of the cases considerably increased. These methods require high computational power to be time efficient.

4 Conclusions

Machine learning has a wide variety of applications in the banking area and the institutions can highly benefit from it, but due the novelty and also the cost of their implementation, they can be reluctant to them, and prefer instead the traditional methods which can also provide traceable outcomes. In order to increase the adoption, proving its high potential and the accuracy of the outcomes is very important. For high-quality results, model selection and hyperparameter tuning play vital roles and the current paper focuses on the assessment of two basic techniques used for hyperparameter tuning: Grid Search and Randomized Search. For this analysis, 6 machine learning algorithms were considered which were run with default hyperparameters, but also with the optimized pairs resulted from the two methods.

During the first run (based on default hyperparameters), we obtained the highest accuracies for XGBoost (82.67%) and Neural Networks (82.08%), the first proving to be very efficient from a run time perspective, as stated by Tianqi Chen, who actually proposed the method [22]. After applying the Grid Search for all the algorithms, the accuracies increased, but the delta was minimal in most of the cases. The only algorithms where the accuracy increased by more than one percentual point were K-Nearest Neighbor (2.17pp increase) and Random Forest, where a delta of 1.05 pp was observed. Based on the Randomized Search method, the results were similar, the accuracy being noticeable only in three cases: for Decision Trees, the optimized hyperparameter pair improved the accuracy by 9.53 percentual points, for K-Nearest Neighbor, it improved by 2.22pp, while for Random Forest the increase was of 1.96 pp. The delta for the remaining three algorithms was below 0.25pp.

In conclusion, these basic hyperparameter tuning methods have the potential to increase the accuracy of a model, but in the current case study, the delta was not noticeable. Though, Random Search gave slightly better results than Grid Search, in line with what Bergstra and Bengio published in 2012 [10]. For future studies, we will take a deeper look into the hyperparameters of the algorithms which are highly impacting the accuracy of the model and we will investigate other tuning methods in order to identify one which outperforms the ones already studied.

References

[1] R. Chuprina, "Machine Learning in Finance: Benefits, Use Cases and Opportunities," SPD Group, 14 January 2020. [Online]. Available: https://spd.group/machine-learning/ml-in-finance/#How_Machine_Learning_in_Finance_Changes_the_industry_Modern_Realities_and_Future_Forecasts. [Accessed 23 April 2021].

[2] B. Hayes, "Machine Learning Adoption Rates Around the World," Business Broadway, 1 February 2021. [Online]. Available: https://businessoverbroadway.com/2021/02/01/machine-learning-adoption-rates-around-the-world/. [Accessed 23 April 2021].

[3] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," Business Horizons, vol. 58, pp. 431-440, 2015.

[4] S. Lessman, B. Baesens, H.-V. Seow and L. C. Thomas, "Benchmarking state-of-

the-art classification algorithms for credit scoring: An update of research," European Journal of Operational Research, 2015.

[5] D. Michie, D. J. Spiegelhalter and C. C. Taylor, Machine learning, neural and statistical classification, River, NJ: Ellis Horwood, 1995.

[6] R. D. King, C. Feng and A. Sutherland, "Statlog: Comparison of Classification Algorithms on Large Real-World Problems," Applied Artificial Intelligence an International Journal, 1995.

[7] R. Kohavi and G. John, "Automatic Parameter Selection by Minimizing Estimated Error," in Proceedings of the Twelfth International Conference on Machine Learning, 1995.

[8] M. Feurer and F. Hutter, "Hyperparameter Optimization," in Automated Machine Learning. The Springer Series on Challenges in Machine Learning., Springer, Cham, 2019.

[9] A. Zheng, Evaluating Machine Learning Models - A Beginner's Guide to Key Concepts and Pitfalls, O'Reilly, 2015.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, pp. 281-305, 2012.

[11] J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in International Conference on Neural Information Processing Systems, 2011.

[12] C. Thornton, F. Hutter, H. Hoos and K. Leyton-Brown, "Auto WEKA: combined selection and hyperparameters optimization of classification algorithms," in Proceedings of KDD'13, 2013.

[13] I.-C. Yeh, "UCI Machine Learning Repository," 2016. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients.

[14] 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 {P}ython," Journal of Machine Learning Research, vol. 12, no. 12, pp. 2825-2830, 2011.

[15] "Machine Learning - Logistic Regression," Tutorials Point, [Online]. Available: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_logistic_regression.htm.

[16] T. Seidl, "Nearest Neighbor Classification," in Encyclopedia of Database Systems, Springer, 2009.

[17] N. S. Chauhan, "Decision Tree Algorithm, Explained," KDnuggets, [Online]. Available: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html. [Accessed 16 April 2021].

[18] L. Breiman, "Random Forests," Machine Learning, vol. 45, 2001.

[19] "What is a Neural Network?," Deep AI, [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/neural-network.

[20] A. Keramati and N. Yousefi, "A Proposed Classification of Data Mining Techniques in Credit Scoring," in International Conference on Industrial Engineering and Operations Management, Kuala Lumpur, Malaysia, 2011.

[21] S. Hamori, M. Kawai, T. Kume, Y. Murakami and C. Watanabe, "Ensemble Learning or Deep Learning? Application to Default Risk Analysis," Journal of Risk and Financial Management, 2018.

[22] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.

[23] "XGBoost," [Online]. Available: https://xgboost.readthedocs.io/en/latest/index.html.

Claudia ANTAL-VAIDA graduated the Faculty of Economic Cybernetics, Statistics and Informatics at the Bucharest Academy of Economic Studies, with bachelor's degree in Cybernetics and a master's degree in Business Analysis and Enterprise Performance Control. She is currently a PhD Student at the same University, mostly interested in business analytics, machine learning techniques, big data and business performance.