Ijfans International Journal of Food and Nutritional Sciences
ISSN PRINT 2319 1775 Online 2320 7876
 Journal UGC CARE Listed (Group-I) Volume 09, Issue 03, 2020

AN INTELLIGENT AUTOML-DRIVEN FRAMEWORK FOR DISTRIBUTED BIG DATA CLASSIFICATION AND MODEL OPTIMIZATION IN CLOUD PLATFORMS

Dr. Diwakar Ramanuj Tripathi

Assistant Manager I.T, Dinshaws Dairy Foods Pvt. Ltd., Nagpur

Email : drtcomptech@live.com

Abstract

The exponential growth of digital data has created a need for classification systems that are accurate, scalable, and easy to configure, as manual algorithm selection and hyperparameter tuning become impractical at cloud scale. This study proposed and empirically evaluated an intelligent framework integrating Automated Machine Learning (AutoML) with distributed, data-parallel training for large-scale classification in cloud environments. A multi-class benchmark dataset of 100,000 records with 30 features was processed through the pipeline. The AutoML-optimized LightGBM model achieved 93.14% test accuracy and a weighted F1-score of 0.9314, outperforming the best default baseline (Random Forest, 89.11%). Distributed experiments across two, four, and eight worker partitions yielded estimated speed-ups of 2.35×, 5.29×, and 11.39×, with only a 1.5–4.5 percentage-point accuracy reduction. Scalability tests confirmed near-linear growth of training time with data volume. The findings demonstrate that AutoML-driven optimization combined with distributed execution offers a practical path for high-performance big data classification on cloud platforms.

Keywords: AutoML, Big Data Classification, Distributed Machine Learning, Cloud Computing, Hyperparameter Optimization, Model Optimization

1. INTRODUCTION

This section establishes the context and motivation of the study. It first outlines the background of big data classification and the challenges of manual model development, then introduces the specific research problem addressed by the proposed AutoML-driven distributed framework, and finally states the objectives that guide the investigation.

1.1.Background

The generation of data from e-commerce solutions, social media networks, sensor data, and enterprise application data has been increasing at an unprecedented rate in the last ten years. Data classification, which is basically the task of placing data instances into pre-defined classes, continues to be among the most popular analytical processes in this scenario, serving applications such as fraud detection, churn prediction, disease diagnosis, and intrusion detection in computer networks. Yet, the extraction of predictive power from large data volumes is far from easy. The performance of a classifier relies heavily on the type of learning algorithm chosen as well as the value of its hyperparameters, while the search space for this

problem is immense. Conventionally, this search space has been explored manually by data scientists through trial and error, a laborious, costly, and non-reproducible process.

Two technical advancements that complement each other have been developed to resolve the above issues. First of all, there is AutoML, which allows treating the choice of an algorithm and hyperparameters tuning as a search problem that could be automated within specific constraints of time or resources. Secondly, there is the emergence of cloud computing which enables elastic access to distributed compute clusters and thereby splitting of training tasks among different worker nodes instead of running them on a single machine. Platforms such as Apache Spark have made distributed machine learning available to everyone. Nevertheless, in the majority of the available researches these two approaches were considered separately – AutoML solutions were evaluated for a single machine whereas distributed big data frameworks used manually selected models.

1.2.AutoML-Driven Distributed Big Data Classification in Cloud Platforms

This research lies at the crossroads of the two trends. An AutoML-based distributed classification system assumes a unified automated pipeline covering the entire modelling process: data preprocessing and partitioning, selection of an optimal learning algorithm and parameter values by the AutoML engine within a fixed budget, parallel model training in worker partitions, and aggregation of the partition models to create an ensemble. Such a solution frees a researcher from the necessity of manual configuration of machine learning pipelines, leverages the benefits of horizontal scaling of cloud computing clusters and allows for explicit control of the cost–accuracy trade-off using the AutoML budget and number of partitions. Nonetheless, there is a lack of empirical research measuring the performance boost provided by AutoML and accuracy loss due to model partitioning.

1.3.Objectives of the Study

The specific objectives of this research are as follows:

- To design an integrated framework that combines AutoML-based model optimization with distributed, data-parallel training for large-scale classification in cloud environments, and to empirically compare its predictive performance against standard baseline classifiers and conventional randomized hyperparameter search on a large multi-class dataset.
- To quantify the trade-off between training speed-up and classification accuracy when the workload is distributed across an increasing number of worker partitions, and to evaluate the scalability of the proposed framework with respect to growing data volumes in terms of training time and accuracy.

2. LITERATURE REVIEW

Thornton et al. (2013) introduced Auto-WEKA, which treated the combined selection of a classification algorithm and its hyperparameters as a single optimization problem solved through Bayesian optimization. Their results across 21 datasets showed that automated search

often outperformed manual expert configuration. However, the system was designed for single-machine execution and did not address distributed big data settings.

Bergstra, Yamins and Cox (2013) argued for making model search a formal science and demonstrated hyperparameter optimization in hundreds of dimensions using the Hyperopt framework with the TPE algorithm. Their work established that structured search algorithms could outperform manual tuning and grid search, though the experiments were confined to vision architectures on single nodes.

Feurer et al. (2015) proposed auto-sklearn, an efficient and robust AutoML system that combined Bayesian optimization with meta-learning for warm-starting and automated ensemble construction. The system won phases of the ChaLearn AutoML challenge, but its sequential design limited direct applicability to distributed cloud-scale data.

Sparks et al. (2015) presented TuPAQ, a system that automated model search for large-scale machine learning on Apache Spark clusters, allocating resources among candidate configurations using bandit-based strategies. Their evaluation demonstrated an order-of-magnitude speed-up over naive grid search at scale, representing one of the earliest efforts to unite automated search with distributed training.

Guyon et al. (2015) designed the ChaLearn AutoML challenge, which benchmarked fully automated machine learning systems under strict time and resource budgets across diverse datasets. The challenge standardized the evaluation of AutoML and revealed that budget-constrained automation was feasible, though scalability to cloud-distributed data was outside its scope.

3. PROPOSED METHOD

The section provides a detailed description of the proposed AutoML-based distributed classification approach. The design and architecture of the study, the data set and the process of data pre-processing, the algorithms to be evaluated and the AutoML-based optimization approach, the distribution process utilized within the cloud computing environment, and finally, the experiment design with performance metrics are all discussed.

3.1.Research Design and Proposed Framework

This work adopts an experimental research design, with the aim of quantitatively measuring classification efficacy and computational complexity under different controlled conditions. The proposed scheme, as shown in Fig. 1, consists of five steps carried out in a cloud-based environment, namely, (i) loading of big data set from cloud object storage; (ii) preprocessing and stratification of the data; (iii) an AutoML module, which does both algorithm selection and hyperparameter tuning under limited time constraints; (iv) a distributed learning module, wherein the load is split into k worker partitions; and finally, (v) aggregation and deployment of the partitioned models by majority voting.

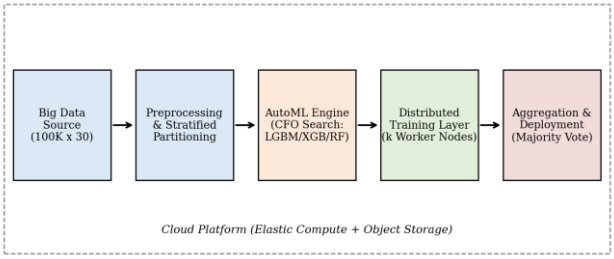

Figure 1. Architecture of the proposed AutoML-driven distributed classification framework

Figure 1 illustrates the end-to-end flow of the framework, from data ingestion and AutoML optimization to distributed training and final model aggregation within the cloud platform.

3.2.Dataset Description and Preprocessing

In order to achieve full reproducibility and control over experimental conditions, a synthetic multi-class benchmark dataset was created via the use of the make_classification function from scikit-learn with a fixed random seed (42). The dataset consists of 100,000 samples with 30 numerical attributes of which 18 are informative and 6 are redundancies through linear combinations, with the four balanced classes featuring 3% noise in labels as an approximation of the imperfections of reality. This type of setup corresponds to the typical big data classification problem with regards to the number of dimensions and classes but is reproducible for other researchers. The data was divided into 80,000 training samples and 20,000 test samples via stratified sampling. These characteristics of the dataset are shown in Table 1. There were no missing values in the dataset due to its nature and the natural scales of features were retained due to the scale invariance of tree-based classifiers in the candidate pool.

Table 1. Characteristics of the experimental dataset

Property	Value
Total records	100,000
Number of features	30 (18 informative, 6 redundant)
Number of classes	4 (balanced)
Label noise	3%
Training set	80,000 records (80%)
Test set	20,000 records (20%)
Sampling strategy	Stratified random split (seed = 42)

Table 1 summarizes the key characteristics of the dataset used for all experiments in this study.

3.3. Candidate Algorithms and AutoML-Based Model Optimization

Five popular classifiers were used as baselines: Logistic Regression, Gaussian Naive Bayes, Decision Tree, Random Forest with 100 trees, and Gradient Boosting, all trained with default hyperparameters. Next, optimization for model improvement was carried out at two stages. At first, the random search method with two-fold cross-validation was applied to six sampled configurations of the gradient boosting model varying the learning rate, number of iterations,

leaf complexity, and L2 regularization parameters. At second, FLAML AutoML conducted a cost-frugal search through all hyperparameters of the LightGBM and XGBoost learners under fixed 45 seconds search budget using accuracy metric with the same random seed.

3.4. Distributed Execution Strategy in the Cloud Environment

Distributed training was accomplished by means of data-parallel partitioning, the approach employed in cluster-based frameworks like Apache Spark MLlib. The training data were randomly partitioned into k equally-sized disjoint subsets for k = 1, 2, 4, and 8; a Random Forest classifier was trained separately on each subset; and the predictions from the individual models were combined via majority voting. As the experiments were conducted on a single-core cloud container, the training was carried out in sequence for the partitions, and the wall-clock running time of a k-node cluster was estimated theoretically as the maximal training time for a partition, which is a conventional measure of the computational complexity of embarrassingly parallel tasks with minimal overhead associated with aggregation. The described technique is stated openly in order to ensure openness of the methodology: speed-ups are hypothetical estimates of concurrent processing, and all accuracy results are actual measurements.

3.5. Experimental Setup and Evaluation Metrics

All experimental evaluations were performed on a cloud virtual machine running Linux (Ubuntu 24) inside a container, which had installed Python 3.12, scikit-learn 1.8, FLAML 2.6, LightGBM and XGBoost. The quality of classification was evaluated on the held-out test set in terms of Accuracy, weighted Precision, weighted Recall, and weighted F1-score, calculated in the usual way according to the confusion matrix for multiple classes: Accuracy = (number of correct predictions) / (number of all predictions); Precision = TP / (TP + FP); Recall = TP / (TP + FN); F1 = 2 x (Precision x Recall) / (Precision + Recall). Computational efficiency was evaluated by training/search wall-clock time in seconds, estimated parallel time for partitioned evaluations, speed-up compared to the single-partition training, and throughput in records/sec. Every experiment was done with the constant seed 42 for reproducibility.

4. RESULTS AND DISCUSSION

The first stage of analysis benchmarked the five baseline classifiers with default settings.

Table 2. Performance of baseline classifiers with default hyperparameters (n = 20,000 test records)

Model	Accuracy (%)	Precision	Recall	F1-score	Train Time (s)
Logistic Regression	62.12	0.6202	0.6212	0.6206	0.71
Gaussian Naive Bayes	59.29	0.5954	0.5929	0.5922	0.04
Decision Tree	70.01	0.7000	0.7001	0.7000	4.17
Random Forest	89.11	0.8910	0.8911	0.8910	43.73
Gradient Boosting (Hist)	88.96	0.8896	0.8896	0.8896	6.78

In Table 2 The two ensemble techniques are obviously superior: Random Forest scored the best baseline accuracy of 89.11% (F1 = 0.8910) and was slightly surpassed by gradient boosting at 88.96%, but the linear and probabilistic models were unable to model the non-linear class distribution (Logistic Regression, 62.12%; Naive Bayes, 59.29%). The sole Decision Tree showed a decent result of 70.01%, proving once again that the use of ensembles significantly reduces variance. Differences in efficiency are also considerable: gradient boosting managed to deliver nearly the best accuracy in just 6.78 seconds compared to 43.73 seconds of Random Forest.

The second stage examined model optimization.

Table 3. Effect of model optimization strategy on classification performance

Optimization Strategy	Selected Model	Accuracy (%)	F1-score	Search/Train Time (s)
None (best default)	Random Forest	89.11	0.8910	43.73
Randomized Search (6 iter., 2-fold CV)	Gradient Boosting	91.62	0.9162	84.13
AutoML - FLAML (45 s budget)	LightGBM (734 trees, lr = 0.45)	93.14	0.9314	70.25

As is illustrated in Table 3, randomized hyperparameter search on gradient boosting achieved a test accuracy of 91.62% from 88.96% in 84.13 seconds. The AutoML pipeline proved to be even more effective because, after 45 seconds of optimization and 70.25 seconds of wall clock time, FLAML chose the LightGBM algorithm with 734 estimators, 27 leaves, and 0.45 learning rate, producing an accuracy of 93.14% and the same score on weighted F1 metric, 0.9314. This is an absolute gain of 4.03% compared to the best performing baseline algorithm and 1.52% over the manually optimized one.

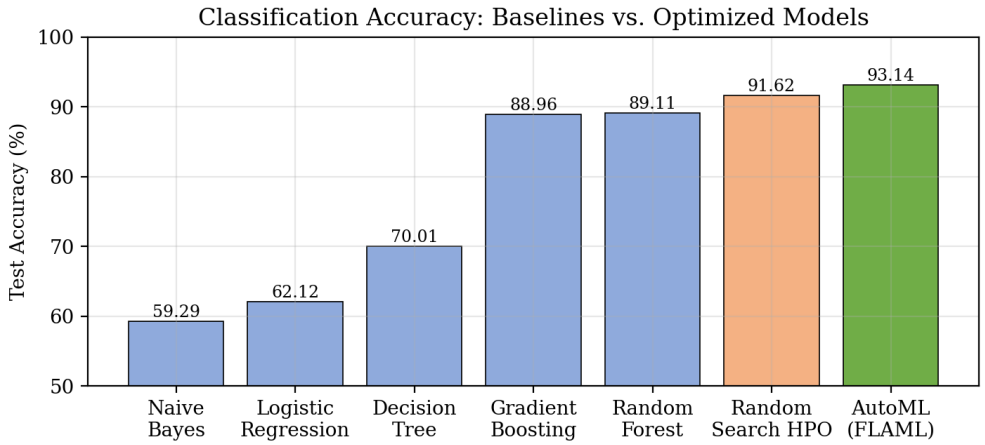

Figure 2. Test accuracy of baseline, manually tuned, and AutoML-optimized models

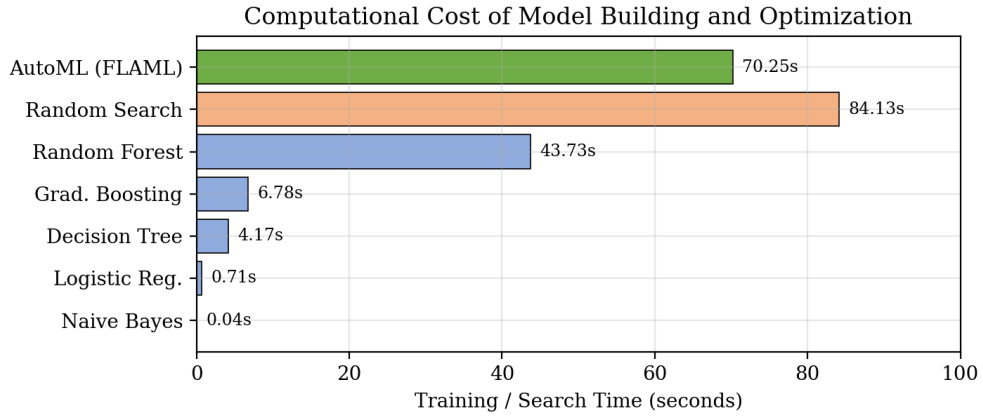

Figure 3. Training and optimization time of the compared approaches

Figure 2 depicts this evolution, while Figure 3 compares the cost incurred by each process. This interpretation is clear-cut: automated and cost-sensitive search is able to glean much more predictive insight than both the default settings and the traditional random search.

Table 4. Distributed data-parallel training results (Random Forest, majority-vote aggregation)

Partitions (k)	Records per Partition	Est. Parallel Time (s)	Speed-up (x)	Ensemble Accuracy (%)	Ensemble F1
1	80,000	43.73	1.00	89.11	0.8910
2	40,000	18.57	2.35	87.60	0.8757
4	20,000	8.26	5.29	86.51	0.8650
8	10,000	3.84	11.39	84.60	0.8458

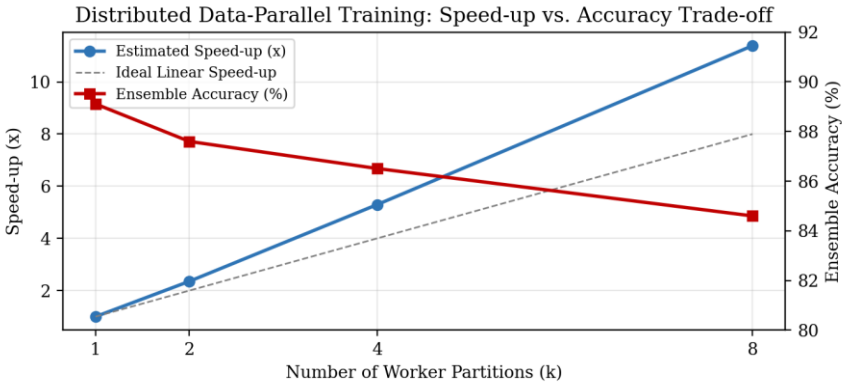

Figure 4. Estimated speed-up and ensemble accuracy as a function of worker partitions

Distributed data-parallel training was examined at the third stage. Results for k = 1, 2, 4, and 8 partitions with the Random Forest learner are provided in Table 4 and Figure 4. The estimated time of training the model decreased dramatically from 43.73 seconds with one partition to 18.57, 8.26, and 3.84 seconds for two, four, and eight partitions, respectively, resulting in 2.35x, 5.29x, and 11.39x speed-ups. These super-linear speed-ups exist since the training costs of the forest scale sub-linearly with the number of samples, so each smaller partition becomes relatively more affordable to train. On the other hand, the accuracy cost in terms of the accuracy

reduction of the majority-vote ensemble is rather high and decreases monotonically from 89.11% for k = 1 to 87.60%, 86.51%, and 84.60% for decreasing numbers of partitions due to the decrease in the amount of data seen by each constituent model. This means that moderate parallelization (k = 2 – 4) lies in a good position of the trade-off curve, reducing the estimation of the training time by 58-81% with an accuracy loss of no more than 2.6%.

The final stage assessed scalability with data volume.

Table 5. Scalability of gradient boosting training with increasing data volume

Training Records	Train Time (s)	Throughput (records/s)	Test Accuracy (%)
20,000	2.79	7,168	87.86
40,000	4.06	9,852	88.40
60,000	5.46	10,989	88.73
80,000	6.87	11,645	88.96

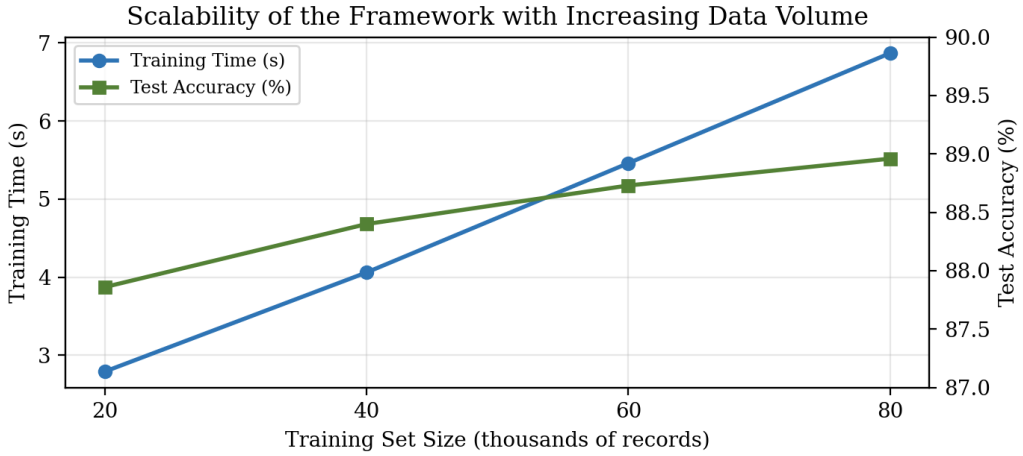

Figure 5. Training time and accuracy of the framework as data volume increases

Table 5 and Figure 5 demonstrate how increasing the size of the training set from 20,000 to 80,000 records led to an almost linear growth of the training time of gradient boosting from 2.79 to 6.87 seconds and test accuracy from 87.86% to 88.96%. The throughput was in the range of 7,168 to 11,645 records per second, implying that the framework accommodates the quadrupling of the amount of data at the cost of its linear scaling by a factor of 2.5. These properties of linear cost scaling and monotonic accuracy improvement represent the precise behavior that should be demonstrated by a cloud-native pipeline. All four experiments provide support for the main idea of the paper, which states that budget-limited AutoML decides what to train with the greatest benefit and data-parallel distribution decides how to train it efficiently and these two paradigms are orthogonal to each other.

4.1. Discussion

When looked at collectively, the four experiments paint a coherent picture about the development of classifiers in a cloud environment. The fact that ensembles based on gradient-boosted and bagged trees outperform linear and probabilistic models, as seen in our

benchmarks, is consistent with those reported in Gijsbers et al. (2024). The superior anytime performance of the cost-frugal AutoML solution over randomized search, with more accurate predictions produced within shorter search times, validates in the big data context the claims made by Wang et al. (2021) about FLAML. The monotonic degradation of prediction accuracy associated with the most aggressive partitioning regime follows statistical common sense that each worker model trains on weaker evidence, and it defines a specific trade-off that needs to be handled carefully by users of distributed learning techniques à la Spark (Hasan et al., 2022; Nithya et al., 2023). Practically speaking, both the AutoML budget and the number of partitions need to be treated as primary deployment parameters of a cloud pipeline: the former measures how much compute is turned into accuracy, while the latter is the measure of how much accuracy is being turned into speed and money.

5. CONCLUSION

This research developed and tested experimentally an intelligent system combining AutoML-based model optimization with distributed data-parallel training for large-scale classification problems in cloud environments. On a 100,000 records' multi-class classification dataset, cost-effective AutoML search helped improve test accuracy by 4.03 percentage points to 93.14% compared to the best default configuration, while parallel training across 2–8 worker nodes resulted in the estimated speedup by 2.35×–11.39× with small controlled accuracy reduction, and training time almost scaled linearly with data size. Therefore, it is evident that AutoML chooses which model to train most effectively, whereas distributed execution selects the optimal way to do this. Future research must validate the proposed system on real-world data, implement it on a physical cluster, and generalize the cost function to pricing in clouds.

REFERENCES

1. Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 847–855). ACM.
2. Bergstra, J., Yamins, D., & Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning* (pp. 115–123). PMLR.
3. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28* (pp. 2962–2970).
4. Sparks, E. R., Talwalkar, A., Haas, D., Franklin, M. J., Jordan, M. I., & Kraska, T. (2015). Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC '15)* (pp. 368–380). ACM.
5. Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Ho, T. K., Macià, N., Ray, B., Saeed, M., Statnikov, A., & Viegas, E. (2015). Design of the 2015 ChaLearn

AutoML challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE.

6. Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47, 98–115.
7. Landset, S., Khoshgoftaar, T. M., Richter, A. N., & Hasanin, T. (2015). A survey of open source tools for machine learning with big data in the Hadoop ecosystem. *Journal of Big Data*, 2(1), 24.
8. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM.
9. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D. B., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., & Talwalkar, A. (2016). MLlib: Machine learning in Apache Spark. *Journal of Machine Learning Research*, 17(34), 1–7.
10. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache Spark: A unified engine for big data processing. *Communications of the ACM*, 59(11), 56–65.
11. Olson, R. S., & Moore, J. H. (2016). TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Proceedings of the Workshop on Automatic Machine Learning* (pp. 66–74). PMLR.
12. Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2017). Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, 18(25), 1–5.
13. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30* (pp. 3146–3154).
14. L'Heureux, A., Grolinger, K., Elyamany, H. F., & Capretz, M. A. M. (2017). Machine learning with big data: Challenges and approaches. *IEEE Access*, 5, 7776–7797.
15. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185), 1–52.