2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS) | 979-8-3503-9645-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/IoTaIS56727.2022.9975984

Accuracy-Time Efficient Hyperparameter Optimization Using Actor-Critic-based Reinforcement Learning and Early Stopping in OpenAI Gym Environment

Albert Budi Christian*, Chih-Yu Lin†, Yu-Chee Tseng*, Lan-Da Van*, Wan-Hsun Hu‡, and Chia-Hsuan Yu‡
*Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan
†Department of Computer Science and Engineering, National Taiwan Ocean University, Taiwan
‡Chunghwa Telecom Laboratories, Chunghwa Telecom, Taiwan
Email: albert.c@nycu.edu.tw, lincyu@mail.ntou.edu.tw, yctseng@cs.nycu.edu.tw,
ldvan@cs.nycu.edu.tw, ringo0601@cht.com.tw, tristan@cht.com.tw,

***Abstract*—In this paper, we present accuracy-time efficient hyperparameter optimization (HPO) using advantage actor-critic (A2C)-based reinforcement learning (RL) and early stopping in OpenAI Gym environment. The A2C RL can improve the hyperparameter selection such that the resulting accuracy of machine learning (ML) algorithms including XGBoost, support vector classifier (SVC), random forest shows comparable. According to the specified accuracy of the ML algorithms, the early stopping scheme can save the computation cost. Ten standard datasets are used to valid the accuracy-time efficient HPO. Experimental results show that the presented accuracy-efficient HPO architecture can improve 0.77% accuracy on average compared with default hyperparameter for random forest. The early stopping can save 64% computation cost on average compared to without early stopping for random forest.**

***Index Terms*—Actor-Critic, Hyperparameter optimization, Reinforcement learning, Accuracy-time efficiency, early stopping**

I. INTRODUCTION

Machine learning (ML) [1] [2] is the core technology to achieve artificial intelligence (AI) era. Due to wide attention and attraction of ML, many researches built different ML frameworks such as Auto-Scikit-learn [3], Auto-Weka [4], and SMAC [5]. The ML learning and inference performance is driven by the data type and quality [6]. Specifically, these data evolve the ML model parameters with frozen hyperparameters. During the process of data training, model parameters can be modified. However, the hyperparameters must be set before training and cannot be changed during training [7]. As we know that the values of hyperparameters significantly affect the ML performance; thus, hyperparameter tuning [8] is necessary for ML training. Traditionally, the manual hyperparameter tuning is operated; however, this way costs efforts and time without clear guidelines due to the huge number of hyperparameters. Currently, the automatic hyperparameter tuning or HPO [3] is the main research trend in this field.

There exist HPO systems such as Vizier [9], RayTune [10], CHOPT [11] and Optuna [12], where most different HPO methods are supported. The gird search method [13] and random search method [14] are most well known HPO methods. The former one uses all combinations of hyperparameters to fully search the best candidate of hyperparameters, but this method is time consuming. The later one randomly selects the pre-defined hyperparameter; thus, the searching speed is faster than the grid search method, but local minimum issue exists.

In order to overcome small and low-dimensional search space, the Bayesian optimization method [15] and gradient-based method [16] are developed. However, these methods still show intensive computations. RL technique is commonly employed within HPO-related studies. In [17], Zhao *et al.* applied RL to optimize searching strategy in artificial bee colony algorithm. Zahavy *et al.* [18] focused on meta gradient RL for off-policy corrections. Liu *et al.* [19] proposed an LSTM network to decrease the high computational cost in the HPO problem. Fairee *et al.* [20] used RL based approaches for neural architecture searching.

Early stopping has been studied in many different HPO problems. Belakaria *et al.* [21] introduced an early stopping mechanism in deep neural network scenario by adopting the Bayesian optimization technique. Li *et al.* [22] focused on accelerating random search through early stopping. Muñoz *et al.* [23] introduced early stopping for the HPO of ML algorithms. Gomez *et al.* [24] applied early stopping in deep meta-RL to neural architecture search.

A few existing works use actor critic network and early stopping to solve HPO problem. In [25], Heinrich *et al.* adopted GPU-based asynchronous advantage actor critic network (GA3C) and early stopping for parallel issue in deep RL. In this paper, we are applying the A2C-based RL [26] and early stopping to perform HPO in ML algorithm.

The main contributions of this paper are described as follows.

1) In this paper, we apply OpenAI Gym to build the RL agent and environment. The agent issues an action

TABLE I: Hyperparameter domain [27]

No	Domain	Search Space Range
1	numerical	$\{\Omega, \mathcal{O}\}$
2	categorical	$\{”cat_1”, ”cat_2”,”...”, ”cat_n” \}$

to tune the hyperparameters of the environment. The tuned hyperparameters are used for ML model training, validation and testing. The output the ML model is used to decide whether to enable early stopping to save the computation.

2) In our work, we explore performance using A2C network and early stopping among 10 datasets, four different search methods and three ML algorithms. To the best of our knowledge, the A2C algorithm plus early stopping haven't been used to solve HPO in ML algorithms yet.

The rest of this paper is organized as follows. Section II discusses our proposed system architecture. Performance evaluation results are shown in Section III. Finally, Section IV remarks the conclusions and future works.

II. PROPOSED ACCURACY-TIME EFFICIENT HPO SYSTEM ARCHITECTURE

In our research, we adopt the HPO's scenario and equations/notations presented in [28] as environment in the RL system. The object function is shown in Eq.(1).

$$\lambda * = \underset{\lambda \in \Lambda}{\arg\max} \mathbb{E}_{(\mathcal{D}_{train}, \mathcal{D}_{valid}) \backsim \mathcal{D}} \mathfrak{A}(\mathcal{A}_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid}) \tag{1}$$

where $\mathcal{A}$ denotes an ML algorithm with $\mathcal{N}$ hyperparameters denoted by $\mathcal{A}_\lambda$ and $\lambda \in \Lambda$. $\mathcal{D}_{train}$ denotes the training data, while $\mathcal{D}_{valid}$ denotes the validation data. The hyperparameter configuration space is denoted by $\mathbf{\Lambda} = \Lambda_1 \times \Lambda_2 \times ...\Lambda_N$ where Λ_i denotes the $i-$th hyperparameter's search space and $\times$ represents the cartesian product. In Table I [27], the hyperparameter has two types

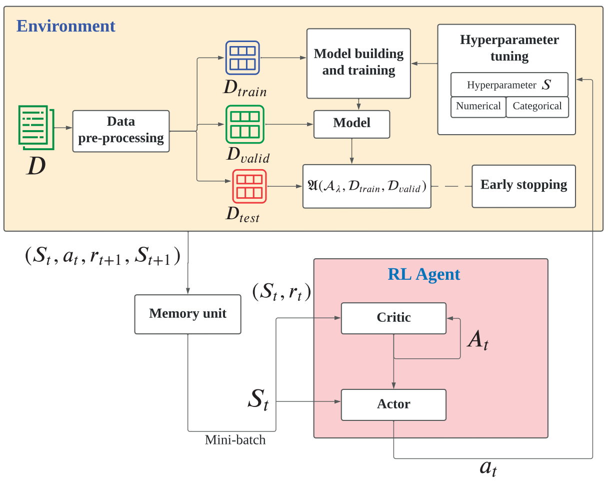


Fig. 1: Presented an accuracy-time efficient system architecture

TABLE II: Model Search Space [29] [27]

Model	Hyperparameter	Type	Search Space	Scale	Default
Random Forest	bootstrap	boolean	True, False	-	True
	criterion	category	gini, entropy, log_loss	-	gini
	min_samples_split	integer	[2, 100]	4	4
	n_estimators	integer	[100, 1200]	100	100
	max_features	category	auto, sqrt, log2	-	sqrt
	min_samples_leaf	integer	[2,100]	4	4
SVC	C	float	[0.1, 2.0]	0.1	1
	kernel	category	linear, poly, rbf	-	rbf
	gamma	category	scale, auto	-	scale
XGBoost	colsample_bylevel	float	[0.5,1.0]	0.1	0.5
	colsample_bytree	float	[0.5,1.0]	0.1	0.5
	gamma	float	[0.05,1.0]	0.01	0.05
	learning_rate	float	[0.01,0.1]	0.01	0.01
	max_depth	integer	[3,25]	1	3
	min_child_weight	integer	[1,9]	2	1
	n_estimators	integer	[100,1200]	10	100
	reg_alpha	float	[0.1,1.0]	0.1	0.1
	reg_lambda	float	[0.01,1.0]	0.01	0.01
	subsample	float	[0.5,1.0]	0.1	0.5

of domain search spaces, numeric value (integer or float) and category. Herein, the hyperparameter is classified into two categories, the numeric type including integer and float, and the category type including the binary and category data. For the numeric type, the total action space is 3^n and the category type depends on the n possible value in the certain hyperparameter.

Considering A2C RL and early stopping, we present our proposed accuracy-time efficient HPO system architecture as shown in Fig. 1. In Fig. 1, there are three main modules including the RL agent module, environment module, and memory module. Herein, we adopt the OpenAI Gym [30] toolkit which is a Python-based environment to build and train RL agents. In the RL agent module, we adopt the A2C during the training phase. The testing data is used to infer the results. Herein, the early stopping scheme is used to save the computation according to the test result accuracy. The outputs of the environment including states (a list of hyperparameter values), action (hyperparameter tuning), and reward (accuracy of the ML model) feed into the memory unit module. The agent module interacts with environment module iteratively until the performance meets the early stopping criterion. In our system architecture, we implement Cartesian product algorithm [31] to generate the list of our actions. The list of action a from the agent is processed based on the hyperparameter data type. The overall operation steps in Fig. 1 are described as follows.

Step 1: Sample (s_t, a_t) using the policy $\pi\theta$ from the actor network.

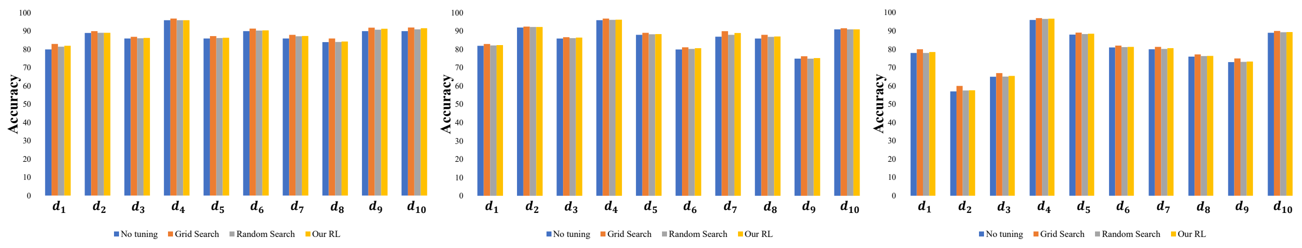
Step 2: Train an ML model with the selected hyperparameters on the training data $\mathcal{D}_{train}$.

Step 3: Verify the ML model on the validation data $\mathcal{D}_{val}$.

Step 4: Test the ML model to obtain the accuracy on the testing data $\mathcal{D}_{test}$ as a reward signal.

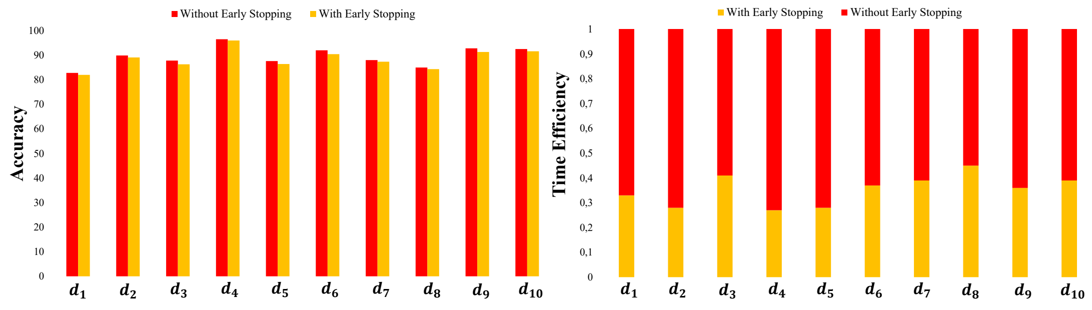
Step 5: If the accuracy meets the early stopping criterion, the overal processes will be terminated. If no, go to Step 6.

Step 6: Evaluate the advantage function A_t in the critic network [32].



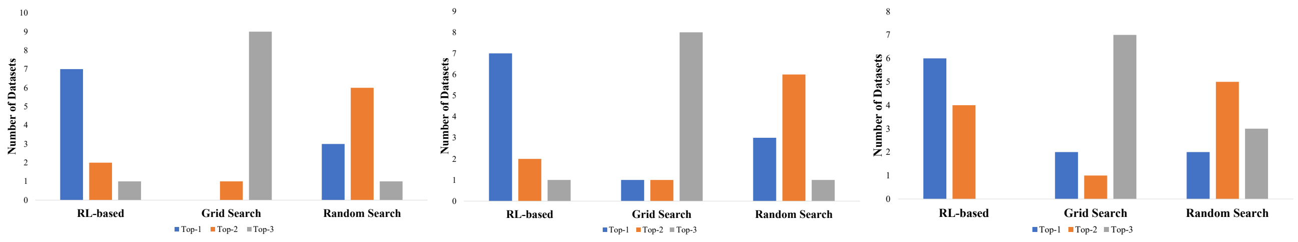
(a) Random Forest (b) XGBoost (c) SVC

Fig. 2: HPO performance for 10 classification datasets using baseline, grid search, random search, and A2C-based RL on three algorithms.



(a) Accuracy (b) Time efficient

Fig. 3: HPO performance of A2C-based RL for the random forest algorithm with and without early stopping.



(a) Time efficient: Random Forest (b) Time efficient: XGB (c) Time efficient: SVC

Fig. 4: Top 3 number of datasets ranked by time efficiency among three different searching methods and three ML algorithms.

Step 7: Evaluate the gradient and update the policy parameters in the actor network [32].

Step 8: Update the weights of the critic network [32]. Then go back to Step 1 until the optimal policy $\pi\theta$ is obtained.

III. Experimental Results

In this section, the experimental results are shown and discussed to validate our proposed system architecture in Fig. 1. We apply the proposed method to tune the hyperparameters in three well-known ML models including Random Forest [27], XGBoost Classifier (XGB) [33] and SVC [27]. We compared our method with the default hyperparameters (no tuning), grid search and random search methods. Table II shows the search space of each algorithm. We set up the algorithms as our environment and test on 10 different datasets, respectively. The datasets are selected from Kaggle and UCI ML repository as shown in Table III. These datasets focuses on classification problems. In this experiment, the dataset $\mathcal{D}$ is splitted into $\mathcal{D}_{train}, \mathcal{D}_{valid}$, and $\mathcal{D}_{test}$ with the following ratio 60%, 20%, 20% respectively.

As shown in Fig. 2, HPO results in terms of accuracy can be improved compared with no-tuning and random search methods or closed to that of grid search method even though grid search uses all possibility of hyperparameter range. For random forest in Fig.2(a), XGBoost in Fig.2(b), and SVC in Fig.2(c), the corresponding accuracies can be improved by 0.77%, 0.69%, and 0.60%, respectively. Next, we evaluate the accuracy performance and time efficiency considering the early stopping scheme and without early stopping scheme as a baseline. Herein, we take the random forest algorithm for demonstration. Fig. 3(a) shows even though there is an accuracy reduction around 1.25% on average, the early stopping scheme can save the computational cost around 64% as shown in Fig. 3(b).

Finally, we evaluate the four methods in terms of top 3 ranking time efficiency by counting the number of datasets as shown in Fig. 4. Obviously, the presented

TABLE III: Datasets Name

No	Dataset Name	Tasks
d_1	Default of credit card clients [34]	Classification
d_2	Date Fruit Datasets [35]	Classification
d_3	Fashion MNIST [36]	Multiclass Classification
d_4	Taiwanese Bankruptcy Prediction Dataset [37]	Classification
d_5	Portuguese Bank Marketing Dataset [38]	Classification
d_6	UCI ML Drug Review dataset [39]	Classification, Regression, Clustering
d_7	UCI Adult Census Data Dataset [40] [41]	Classification
d_8	Census Income Data Set [41]	Classification
d_9	Diabetes Health Indicators Dataset [42]	Classification
d_{10}	Climate Model Simulation Crashes Dataset [43]	Classification

A2C RL method can attain the rank 1 time efficiency among three ML algorithms and 10 datasets. From Figs. 2, 3, 4, the grid search method can achieve the best accuracy result but leads to the largest amount of time. The random search is vice versa. Generally, the presented HPO using A2C RL and early stopping scheme is able to achieve better tradeoff performance considering accuracy and time efficiency among 10 datasets and three machine learning algorithms compared with other three methods. This proves that A2C RL with early stopping can be a good solution to develop the auto ML framework.

IV. CONCLUSION AND FUTURE WORKS

In this work, the following contributions are attained. 1) Present an accuracy-time efficient HPO using A2C-based RL and early stopping. 2) Detailed evaluation and simulation results are discussed within ten datasets for three ML algorithms by four searching methods. Through the system architecture and evaluation results, the better tradeoff HPO can be achieved. In the near future, this architecture and simulation can be explored further in different ML algorithms, searching methods, early stopping criterions, and more datasets for various intelligent applications.

REFERENCES

[1] S. Mirjalili, H. Faris, and I. Aljarah, *Evolutionary machine learning techniques*. Springer, 2019.

[2] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[3] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," *Advances in neural information processing systems*, vol. 28, 2015.

[4] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "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*, 2013, pp. 847–855.

[5] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*. Springer, 2011, pp. 507–523.

[6] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[7] M. Claesen and B. De Moor, "Hyperparameter search in machine learning," *arXiv preprint arXiv:1502.02127*, 2015.

[8] P. Probst, A.-L. Boulesteix, and B. Bischl, "Tunability: importance of hyperparameters of machine learning algorithms," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1934–1965, 2019.

[9] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1487–1495.

[10] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[11] J. Kim, M. Kim, H. Park, E. Kusdavletov, D. Lee, A. Kim, J.-H. Kim, J.-W. Ha, and N. Sung, "Chopt: Automated hyperparameter optimization framework for cloud-based machine learning platforms," *arXiv preprint arXiv:1810.03527*, 2018.

[12] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[13] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[14] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[15] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[16] S. Putatunda and K. Rama, "A modified bayesian optimization based hyper-parameter tuning approach for extreme gradient boosting," in *2019 Fifteenth International Conference on Information Processing (ICINPRO)*. IEEE, 2019, pp. 1–6.

[17] H. Zhao and C. Zhang, "A decomposition-based many-objective artificial bee colony algorithm with reinforcement learning," *Applied Soft Computing*, vol. 86, p. 105879, 2020.

[18] T. Zahavy, Z. Xu, V. Veeriah, M. Hessel, J. Oh, H. P. van Hasselt, D. Silver, and S. Singh, "A self-tuning actor-critic algorithm," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 913–20 924, 2020.

[19] X. Liu, J. Wu, and S. Chen, "Efficient hyperparameters optimization through model-based reinforcement learning and meta-learning," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2020, pp. 1036–1041.

[20] S. Fairee, C. Khompatraporn, S. Prom-on, and B. Sirinaovakul, "Combinatorial artificial bee colony optimization with reinforcement learning updating for travelling salesman problem," in *2019 16th international conference on electrical engineering/electronics, computer, telecommunications and information technology (ecti-con)*. IEEE, 2019, pp. 93–96.

[21] S. Belakaria, R. Sheth, J. R. Doppa, and N. Fusi, "Bayesian optimization over iterative learners with structured responses: A budget-aware planning approach," *arXiv preprint arXiv:2206.12708*, 2022.

[22] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[23] Á. L. Muñoz Castañeda, N. DeCastro-García, and D. Escudero García, "Rhoaso: An early stop hyper-parameter optimization algorithm," *Mathematics*, vol. 9, no. 18, p. 2334, 2021.

[24] J. Gomez Robles and J. Vanschoren, "Learning to reinforcement learn for neural architecture search," *arXiv e-prints*, pp. arXiv–1911, 2019.

[25] G. Heinrich and I. Frosio, "Metaoptimization on a distributed system for deep reinforcement learning," in *2019 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*. IEEE, 2019, pp. 19–30.

[26] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[28] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated machine learning*. Springer, Cham, 2019, pp. 3–33.

[29] W. Jia, C. SenPeng, and C. XiuYun, "Rpr-bp: A deep reinforcement learning method for automatic hyperparameter optimization," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[30] G. B. et al., "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[31] O. Agesen, "The cartesian product algorithm," *European Conference on Object-Oriented Programming*, 1995.

[32] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[33] 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*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[34] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert systems with applications*, vol. 36, no. 2, pp. 2473–2480, 2009.

[35] M. Koklu, R. Kursun, Y. S. Taspinar, and I. Cinar, "Classification of date fruits into genetic varieties using image analysis," *Mathematical Problems in Engineering*, vol. 2021, 2021.

[36] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[37] D. Liang, C.-C. Lu, C.-F. Tsai, and G.-A. Shih, "Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study," *European Journal of Operational Research*, vol. 252, no. 2, pp. 561–572, 2016.

[38] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decision Support Systems*, vol. 62, pp. 22–31, 2014.

[39] F. Gräßer, S. Kallumadi, H. Malberg, and S. Zaunseder, "Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning," in *Proceedings of the 2018 International Conference on Digital Health*, 2018, pp. 121–125.

[40] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[41] R. Kohavi *et al.*, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." in *Kdd*, vol. 96, 1996, pp. 202–207.

[42] Z. Xie, O. Nikolayeva, J. Luo, and D. Li, "Peer reviewed: building risk prediction models for type 2 diabetes using machine learning techniques," *Preventing chronic disease*, vol. 16, 2019.

[43] D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, and Y. Zhang, "Failure analysis of parameter-induced simulation crashes in climate models," *Geoscientific Model Development*, vol. 6, no. 4, pp. 1157–1171, 2013.