

Genetic programming automated machine learning: architecture and performance evaluation

Suraya Masrom¹, Nor Faezah Mohamad Razi¹, Masurah Mohamad¹, Abdullah Sani Abd Rahman²,
Nor Aslily Sarkam³

¹Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perak Branch, Tapah, Malaysia

²Department of Computing, Faculty of Science, Management and Computing, Universiti Teknologi Petronas, Seri Iskandar, Malaysia

³Fakulti Pengurusan Teknologi dan Teknousahawanan, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia

Article Info

Article history:

Received Aug 18, 2026

Revised Aug 28, 2026

Accepted Sep 4, 2026

Keywords:

Automated machine learning

Genetic programming

Google colab

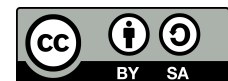
Tree-based pipeline optimization
tool

Web-based GUIs

ABSTRACT

Genetic programming-based automated machine learning (GP-AutoML) automates machine-learning pipelines through evolutionary search, but its implementation requires programming knowledge and configuration of multiple software components. This study presents a web-based platform that integrates the tree-based pipeline optimization tool (TPOT) to support GP-AutoML through a graphical modeling workflow. The platform incorporates conventional machine-learning modules, GP-AutoML configuration, result visualization, user authentication, and cloud-based storage, supported by a flask backend, scikit-learn, TPOT, and Firebase services. A comparative evaluation was conducted between the proposed web-based platform and Google colab Python scripting using five binary-classification datasets with varying numbers of records, features, and class distributions. Testing accuracy, Class 1 precision, recall, F1-score, and time-to-completion were evaluated under predefined GP parameter settings. Google colab obtained higher testing accuracy for all datasets. However, the differences were small, ranging from 0.01 to 0.06. This indicates comparable predictive performance by the web-based GUI platform. For the moderately balanced dataset, both environments achieved stronger Class 1 precision, recall, and F1-score. Although the web-based platform required a longer time-to-completion for every dataset, the evaluated runs remained practical for rapid model development. The findings demonstrate the feasibility of delivering TPOT-based GP-AutoML through an integrated web-based architecture that supports accessible visual modeling without direct programming or local software installation.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Suraya Masrom

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perak Branch

Tapah, Malaysia

Email: suray078@uitm.edu.my

1. INTRODUCTION

Genetic programming-based automated machine learning (GP-AutoML) [1] automates machine learning model development through evolutionary optimization [2]. It automates feature selection, feature engineering, model selection, and hyperparameter optimization, thereby generating candidate machine-learning pipelines for different datasets and prediction tasks [3], [4]. Compared with conventional machine learning

development, GP-AutoML can reduce manual trial-and-error and support the discovery of high-performing pipelines within complex search spaces [5]. Recent studies have further explored evolutionary approaches for constructing composite machine-learning pipelines that integrate data-processing and predictive-model components [6]. Consequently, GP-AutoML has gained attention in domains requiring adaptive, data-driven predictive analytics. Recent AutoML research further highlights the continuing development of automated pipeline-design and optimization approaches across different machine learning tasks [7]. Consequently, GP-AutoML has gained attention in domains that require adaptive and data-driven predictive analytics solutions.

Tree-based pipeline optimization tool (TPOT) [8] is an open-source Python-based AutoML framework that uses genetic programming (GP) [9] to evolve machine learning pipelines composed of pre-processing operators, feature selectors, estimators, and hyperparameter settings. Existing TPOT research has primarily concentrated for improving the Python-based framework and its evolutionary search process. For example, Le *et al.* [10] introduced a feature-set selector to improve TPOT scalability for high-dimensional biomedical data, while Ribeiro *et al.* [11] presented TPOT2, a graph-based implementation designed to provide a more modular and extensible representation of scikit-learn pipelines. More recently, Hernandez *et al.* [12] reviewed TPOT applications in biomedical research and highlighted its use for GP-based pipeline optimization. These studies demonstrate the development and application of TPOT through Python programming environments.

Despite the growing adoption of machine learning technologies, many existing rapid machine learning platforms such as Rapid Miner [13], WEKA [14], and KNIME [15], primarily support conventional machine learning workflows and provide limited direct integration of GP-AutoML. Implementing GP-AutoML in these environments may require external scripting, manual library configuration, or advanced programming knowledge. Accessibility to GP-AutoML is therefore not merely a user-interface concern. It is also an informatics issue involving software-environment configuration, computational-resource management, and the deployment of generated machine learning pipelines. These requirements can limit the practical adoption of evolutionary AutoML by users who need intelligent modeling capabilities but do not have the expertise or infrastructure to configure and execute Python-based workflows. More specifically, within the rapid modeling platforms considered in this study, TPOT-based GP pipeline search is not natively exposed through a web-based guided configuration workflow. This creates a technical gap between Python-based GP-AutoML libraries and accessible web-based intelligent modeling systems.

To address this gap, this study develops and describes a web-based platform that integrates GP-AutoML using the Python TPOT framework. The platform incorporates graphical model configuration, server-side execution, result visualization, user authentication, and cloud-based storage. The study further evaluates the platform's predictive accuracy and execution time against a Google colab Python-scripting environment using five classification datasets of different sizes. Thus, the study contributes an integrated web-based architecture for implementing TPOT-based GP-AutoML and a comparative evaluation of its predictive and computational performance.

2. METHOD

2.1. System architecture

Figure 1 presents the architecture and operational workflow of the proposed web-based GP-AutoML platform. The front-end layer was developed using HTML, JavaScript, and CSS. Through the front-end interfaces, users select the target variable, upload a tabular dataset, specify the GP-AutoML settings, and initiate the modelling process without directly writing Python scripts.

The flask backend acts as the communication layer between the graphical interface and the machine learning services [16]. User-defined configuration settings and uploaded data are transmitted from the front end to the flask application. The backend invokes scikit-learn for conventional machine learning functions and TPOT for GP-based pipeline search. During GP-AutoML execution, TPOT generates and evaluates candidate pipelines. The best pipeline and its performance results are returned to the flask backend before being displayed through the result-visualization module. Firebase authentication [17] is used to manage user sign-in and access to the platform. The architecture separates the presentation, application, machine learning, and cloud-service layers, to allow the GP-AutoML engine to be updated independently from the graphical interface. To reduce risks associated with dataset upload, the platform should restrict accepted file formats, and limit access to authenticated users.

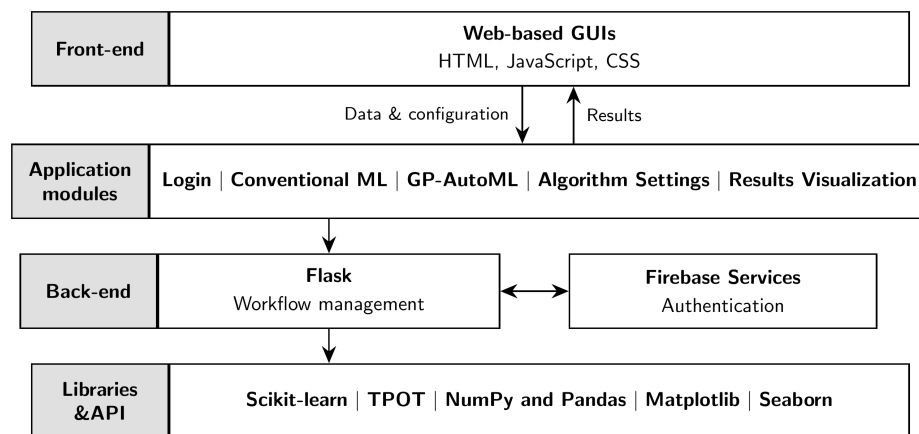


Figure 1. System architecture

2.2. Genetic programming-based automated machine learning execution workflow

Figure 2 illustrates the GP-AutoML execution workflow of the proposed web-based platform. The user-defined TPOT parameters include the population size, number of generations, crossover rate, mutation rate, folds number for cross-validation, and random seed. TPOT represents each candidate solution as a tree-based machine learning pipeline that may consist of pre-processing operators, feature-selection methods, classifiers, and hyperparameter configurations [4], [18]. The GP search begins by generating an initial population of candidate pipelines. Each candidate pipeline is evaluated using a two-objective fitness function. The first objective is to maximize cross-validated predictive accuracy, whereas the second objective is to minimize pipeline complexity, measured by the total number of operators in the pipeline. The GP search evolves a population of candidate machine-learning pipelines. Population size determines the number of candidate solutions evaluated in each generation, while crossover and mutation generate new pipeline configurations for further exploration [11]. TPOT applies the non-dominated sorting genetic algorithm II (NSGA-II) to select non-dominated pipelines that provide a balance between high predictive performance and low structural complexity [12], [19]. The selected pipelines then generate new offspring through one-point crossover. When an offspring is selected for mutation, TPOT randomly applies one of its standard mutation operators (point, insert, or shrink). The cycle for evaluation, selection and variation is repeated until the predefined number of generations is reached. Finally, the best pipeline from the Pareto front is selected and presented through the graphical interface.

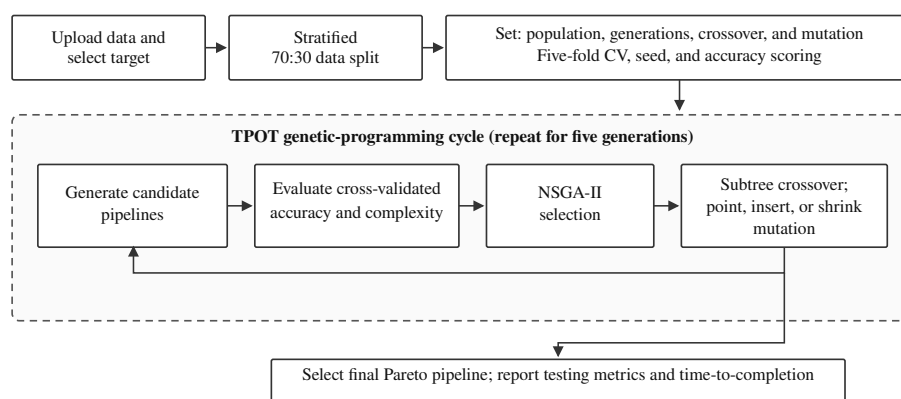


Figure 2. GP-AutoML execution workflow implemented through the proposed TPOT-based web platform

2.3. Datasets for evaluation

The datasets used in this study comprise both small and large size datasets with different rows and features. While dataset size alone is not the sole determinant of machine learning performance, variations in

dataset scale provide valuable insight into the GP-AutoML performances across different execution contexts. An overview of the datasets employed in the evaluation is provided in Table 1.

Table 1. Characteristics and sources of the evaluation datasets

Dataset	Records	Features	Target variable	Class distribution	Source
ESG	722	13	ESG level (1/0)	Class 1: 62.47%; Class 0: 37.53%	Kaggle
Student performance	2,391	14	Student-grade (1/0)	Class 1: 15.72%; Class 0: 84.28%	Kaggle
Bank marketing	4,119	20	Term-deposit subscription (1/0)	Class 1: 10.95%; Class 0: 89.05%	UCI
Telco customer churn	7,043	20	Customer churn (1/0)	Class 1: 26.54%; Class 0: 73.46%	Kaggle
Diabetes	101,766	50	Readmission (1/0)	Class 1: 11.16%; Class 0: 88.84%	Kaggle

Note: Class 1 denotes the outcome of interest and Class 0 denotes the alternative outcome.

Previous studies indicate that dataset scale and feature dimensionality can affect GP-AutoML performance. As the number of records increases, each candidate pipeline requires more time for repeated training and evaluation. Feature dimensionality may further increase the search cost and introduce irrelevant variables that affect predictive performance [20], [21]. Accordingly, five publicly available binary-classification datasets were selected to provide variation in dataset scale, feature dimensionality, and class distribution, as summarized in Table 1. The datasets are arranged in ascending order of record count, ranging from 722 to 101,766 records and from 13 to 50 features. The selection intentionally includes four datasets with class distributions skewed towards Class 0 namely Student Performance, Bank Marketing, Telco Customer Churn, and Diabetes. In these datasets, Class 1, which denotes the outcome of interest, represents between 10.95% and 26.54% of observations. ESG was included as a contrasting dataset with a moderate class distribution in which Class 1 is the majority (62.47%). Therefore, accuracy is reported together with Class 1 precision, recall, and F1-score to assess positive-class detection under different class-distribution conditions.

2.4. Experimental configuration and parameter settings

All experiments used a stratified 7:30 training-test split and 5-fold cross-validation on the training partition. The stratified sampling procedure is particularly important for datasets with imbalanced class distributions [22]. The held-out test partition was used to report the final predictive accuracy. Each configuration was executed independently three times because TPOT uses stochastic evolutionary operations. Table 2 that lists the experimental configuration.

Table 2. Experimental configuration of the TPOT-based GP-AutoML evaluation

Item	Setting
Data partition and validation	70:30 training-test split; five-fold CV on training data
TPOT and Python versions	TPOT 0.12.1; Python 3.12
GP configuration	Population size: 20; generations: 5; crossover and mutation rates: 0.5
Evaluation settings	Classification accuracy; random seed: 42; three independent runs per configuration
Web-platform hardware	Lenovo Yoga 7i notebook with 16 GB RAM

3. RESULTS AND DISCUSSION

Table 3 presents the predictive and computational performance of TPOT-based GP-AutoML on the held-out testing datasets across the two execution environments. In addition to accuracy, precision, recall, and F1-score are reported for Class 1, which represents the positive outcome defined for each dataset [23]. Reporting Class 1 metrics is important because accuracy alone may be misleading when the dataset classes are imbalanced, particularly for majority of the datasets excluding ESG. These measures indicate how effectively the selected pipeline identifies positive cases while accounting for false-positive and false-negative predictions.

Overall, Google colab obtained higher accuracy than the web-based GUI for all datasets. The accuracy differences range from 0.01 to 0.06, that shows the two environments produced broadly comparable predictive accuracy under the evaluated TPOT configuration. Although the web-based GUI provides GP-AutoML execution through graphical configuration rather than direct Python scripting, it achieved accuracy values close to those obtained in Google colab across all tested datasets. Additionally, the Class 1 metrics provide a more detailed assessment than overall accuracy. With the exception of ESG, the selected datasets are skewed towards Class 0. Therefore, accuracy alone may not adequately reflect the ability to identify the outcome of interest. Student Performance, Bank Marketing, and Diabetes produced relatively low Class 1 recall and F1-scores in

both environments. Telco Customer Churn, which has a higher Class 1 proportion than the other Class 0-skewed datasets, showed stronger positive-class detection in both environments, particularly in Google colab. In contrast, ESG has a Class 1 majority (62.47%) and a lower Class 0 proportion (37.53%). The results on ESG dataset produced higher Class 1 precision, recall, and F1-scores in both environments. This pattern suggests that class distribution may influence GP-AutoML predictive performance. However, other dataset characteristics and the specified TPOT configuration may also contribute to the observed results. Across the evaluated population sizes (10–50), predictive accuracy remained largely stable for most datasets. Increasing the population size did not produce a consistent improvement in testing accuracy under the specified five-generation TPOT configuration, whereas larger populations increased the time-to-completion because more candidate pipelines had to be evaluated. Therefore, a population size of 20 was retained for the cross-environment comparison because it provided competitive accuracy while limiting computational cost.

Table 3. Predictive and computational performance of TPOT-based GP-AutoML on the held-out testing datasets across execution environments (population size=20)

Dataset	Platform	Accuracy	Precision ₁	Recall ₁	F1-score ₁	TTC (s)
ESG	Google colab	[0.71 ± 0.08]	[0.68 ± 0.08]	[0.66 ± 0.09]	[0.67 ± 0.08]	[82 ± 0.1]
	Web-Based GUI	[0.69 ± 0.09]	[0.59 ± 0.09]	[0.62 ± 0.09]	[0.60 ± 0.09]	[90 ± 0.2]
Student performance	Google colab	[0.91 ± 0.08]	[0.48 ± 0.08]	[0.32 ± 0.09]	[0.38 ± 0.09]	[136.54 ± 0.6]
	Web-Based GUI	[0.89 ± 0.09]	[0.40 ± 0.07]	[0.29 ± 0.08]	[0.34 ± 0.08]	[291.33 ± 0.6]
Bank marketing	Google colab	[0.89 ± 0.09]	[0.49 ± 0.07]	[0.33 ± 0.08]	[0.39 ± 0.08]	[274.19 ± 0.7]
	Web-Based GUI	[0.83 ± 0.08]	[0.45 ± 0.08]	[0.31 ± 0.07]	[0.37 ± 0.08]	[345.28 ± 0.5]
Telco customer churn	Google colab	[0.81 ± 0.09]	[0.51 ± 0.09]	[0.65 ± 0.08]	[0.57 ± 0.08]	[285.55 ± 7.8]
	Web-Based GUI	[0.80 ± 0.08]	[0.49 ± 0.07]	[0.48 ± 0.08]	[0.48 ± 0.09]	[468.25 ± 6.4]
Diabetes	Google colab	[0.62 ± 0.08]	[0.28 ± 0.08]	[0.23 ± 0.08]	[0.25 ± 0.09]	[1534.78 ± 2.0]
	Web-Based GUI	[0.59 ± 0.09]	[0.21 ± 0.09]	[0.19 ± 0.09]	[0.20 ± 0.09]	[3816.97 ± 5.3]

Note: accuracy, precision₁, recall₁, and F1-score₁ were calculated using the held-out testing dataset. Values are reported as mean ± standard deviation across three repeated runs. Precision₁, recall₁, and F1-score₁ refer to Class 1. TTC denotes time-to-completion.

The web-based GUI required a higher TTC than Google colab for every dataset. The difference was small for ESG (90.00 s versus 82.00 s) but became more substantial for the larger datasets. For Diabetes, Google colab required 1534.78 s (25.58 min), while the web-based GUI required 3816.97 s (63.62 min). These results show that TTC increased with the evaluated dataset scale and was consistently higher in the web-based GUI environment. However, the evaluated runs were completed within a practical time range for batch model-development and exploratory analysis tasks. Additional advantages for the novice users, the modeling involves no single programming codes or local software installation. The longer TTC in the web-based GUI environment may be associated with additional request handling, data-transfer, and application-processing steps. Distributed web and serverless architectures may introduce communication, initialization, and execution overheads [24]. In contrast, Google colab executes the Python workflow directly within its hosted notebook environment [25].

4. CONCLUSION

This study developed a web-based GP-AutoML platform that integrates TPOT pipeline optimization through GUIs. The platform enables users to do machine learning modeling including GP-AutoML without directly writing Python codes. Its performance was evaluated against a Google colab Python-scripting environment using five public binary-classification datasets with different sizes, feature dimensions, and class distributions. The results indicate that the web-based GUI achieved predictive performance comparable to direct Python scripting under the evaluated TPOT configuration. The web-based GUI consistently required a longer time-to-completion than Google colab, and this difference increased for larger datasets. Nevertheless, the evaluated runs were completed within a practical time range. Therefore, the platform is suitable for users who prioritize accessible visual configuration and are able to accommodate a longer execution period, rather than applications requiring immediate model results. This study was limited to five binary-classification datasets, a fixed five-generation TPOT configuration, and the evaluated Google colab and web-server computing resources. The TTC findings should therefore not be interpreted as a controlled hardware benchmark. Broader measures, including ROC-AUC, memory use, energy consumption, usability evaluation, and detailed implementation-feature comparisons, were outside the scope of this study. Future work should evaluate the platform using controlled equivalent computing resources, larger and more diverse datasets, additional predictive and resource-consumption measures, and class-imbalance handling techniques. Usability evaluation with

intended users and the release of reproducible implementation artefacts may further strengthen the practical value of the web-based GP-AutoML platform.

ACKNOWLEDGMENTS

We gratefully acknowledge Muhamad Aiman Mukhtar, a degree of Computer Science student at UiTM, Perak, whose contributions were important to the development of the web-based GUI system.

FUNDING INFORMATION

This project was funded by the Universiti Teknologi MARA.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Suraya Masrom		✓				✓		✓	✓			✓		
Nor Faezah Mohamad Razi	✓			✓		✓	✓			✓				
Masurah Mohamad	✓				✓		✓		✓			✓		
Abdullah Sani Abd Rahman		✓					✓			✓	✓			
Nor Aslily Sarkam		✓					✓			✓				

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal Analysis

I : **I**nterpretation

R : **R**esources

D : **D**ata Curation

O : Writing - **O**riginal Draft

E : Writing - Review & **E**dit

Vi : **V**isualization

Su : **S**upervision

P : **P**roject Administration

Fu : **F**unding Acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

DATA AVAILABILITY

The datasets used in this study are publicly available benchmark datasets obtained from Kaggle and UCI Machine Learning Repository.

REFERENCES

- [1] F. Frank and F. Bacao, "Advanced genetic programming vs. state-of-the-art automl in imbalanced binary classification," *Emerging Science Journal*, vol. 7, pp. 1349–1363, 2023, doi: 10.28991/ESJ-2023-07-04-021.
- [2] Y. Mei, Q. Chen, A. Lensen, B. Xue, and M. Zhang, "Explainable artificial intelligence by genetic programming: A survey," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 621–641, 2022, doi: 10.1109/TEVC.2022.3225509.
- [3] M. C. Teixeira and G. L. Pappa, "Transformers as surrogate models for genetic programming in automl tasks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2025, pp. 472–480, doi: 10.1145/3712256.3726396.
- [4] J. H. Moore, P. A. Ribeiro, N. Matsumoto, and A. K. Saini, "Genetic programming as an innovation engine for automated machine learning: The tree-based pipeline optimization tool (TPOT)," in *Handbook of Evolutionary Machine Learning*, Springer, 2023, pp. 439–455, doi: 10.1007/978-981-99-3814-8_14.
- [5] Y. Omozaki, N. Masuyama, Y. Nojima, and H. Ishibuchi, "Multiobjective fuzzy genetics-based machine learning for multi-label classification," in *IEEE International Conference on Fuzzy Systems*, Glasgow, UK, pp. 1–8, 2020, doi: 10.1109/FUZZ48607.2020.9177804.
- [6] N. O. Nikitin *et al.*, "Automated evolutionary approach for the design of composite machine learning pipelines," *Future Generation Computer Systems*, vol. 127, pp. 109–125, 2022, doi: 10.1016/j.future.2021.08.022.
- [7] I. Salehin *et al.*, "AutoML: A systematic review on automated machine learning with neural architecture search," *Journal of Information and Intelligence*, vol. 2, no. 1, pp. 52–81, 2024, doi: 10.1016/j.jiixd.2023.10.002.
- [8] D. Radečić, *Machine Learning Automation with TPOT: Build, validate, and deploy fully automated machine learning models with Python*, Packt Publishing Ltd, 2021.

- [9] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 147–167, 2023, doi: 10.1109/TEVC.2023.3255246.
- [10] T. T. Le, W. Fu, and J. H. Moore, "Scaling tree-based automated machine learning to biomedical big data with a feature set selector," *Bioinformatics*, vol. 36, 2020, doi: 10.1093/bioinformatics/btz470.
- [11] P. Ribeiro *et al.*, "TPOT2: A new graph-based implementation of the tree-based pipeline optimization tool for automated machine learning," in *Genetic Programming Theory and Practice XX*, Springer Nature Singapore, 2024, pp. 1–17, doi: 10.1007/978-981-99-8413-8_1.
- [12] J. G. Hernandez, A. K. Saini, A. Ghosh, and J. H. Moore, "The tree-based pipeline optimization tool: Tackling biomedical research problems with genetic programming and automated machine learning," *Patterns*, vol. 6, no. 7, pp. 1–12, 2025, doi: 10.1016/j.patter.2025.101314.
- [13] M. F. Mustapha, A. N. I. Zulkifli, O. Kairan, N. Zizi, N. N. Yahya, and N. M. Mohamad, "The prediction of student's academic performance using rapidminer," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 32, no. 1, pp. 363–371, 2023, doi: 10.11591/ijeecs.v32.i1.pp363-371.
- [14] Y. Dou and W. Meng, "Comparative analysis of weka-based classification algorithms on medical diagnosis datasets," *Technology and Health Care*, vol. 31, no. 1_suppl, pp. 397–408, 2023, doi: 10.3233/THC-236034.
- [15] F. Acito, *Predictive analytics with KNIME*, Springer Nature Switzerland, 2023.
- [16] L. Albesher and R. Alfayez, "An observational study on flask web framework questions on stack overflow (so)," *IET Software*, vol. 2024, no. 1, pp. 1–15, 2024, doi: 10.1049/sfw2/1905538.
- [17] Himanshu, M. Kapoor, M. S. Shekhawat, S. Vohra, and O. Sachdeva, "Enhancing user security and experience: Implementing firebase and google authentication in android development for the news app," in *Applications of Artificial Intelligence in 5G and Internet of Things*, CRC Press, 2025, pp. 177–181, doi: 10.1201/9781003532521-32.
- [18] B. G. Bernardo, F. Nauber, L. Marques, J. Alexandre, and S. J. Fong, "TPOT automated machine learning approach for multiple diagnostic classification of lung radiography and feature extraction," in *Computerized Systems for Diagnosis and Treatment of COVID-19*, Springer, 2023, pp. 117–135.
- [19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.
- [20] R. T. Potla, "Scalable machine learning algorithms for big data analytics: Challenges and opportunities," *Journal of Artificial Intelligence & Research*, vol. 2, no. 2, pp. 124–141, 2022.
- [21] F. Zogaj, J. P. Cambronero, M. C. Rinard, and J. Cito, "Doing more with less: characterizing dataset downsampling for automl," in *Proceedings of the VLDB Endowment*, vol. 14, no. 11, pp. 2059–2072, 2021, doi: 10.14778/3476249.3476262.
- [22] S. Szeghalmy and A. Fazekas, "A comparative study of the use of stratified cross-validation and distribution-balanced stratified cross-validation in imbalanced learning," *Sensors*, vol. 23, no. 4, p. 2333, 2023, doi: 10.3390/s23042333.
- [23] S. Demir, H. Selvitopi, and Z. Selvitopi, "An early and accurate diagnosis and detection of the coronary heart disease using deep learning and machine learning algorithms," *Journal of Big Data*, vol. 12, p. 228, 2025, doi: 10.1186/s40537-025-01283-7.
- [24] K. Linkalapalli, R. K. Kanneganti, M. Gouri, G. K. Sangam, N. Patil, and K. Chaganti, "Serverless and nosql databases in the cloud: A systematic review of security and management techniques," *International Journal on Computational Modelling Applications*, vol. 2, no. 4, pp. 41–52, 2025, doi: 10.63503/j.ijcma.2025.216.
- [25] V. Miranda and C. Jimenez-Orozco, "Data mining for materials science: A jupyter-based tutorial for students and instructors," *Journal of Chemical Education*, vol. 103, no. 5, pp. 2737–2742, 2026, doi: 10.1021/acs.jchemed.5c01585.

BIOGRAPHIES OF AUTHORS



Suraya Masrom     is the member of Machine Learning and Interactive Visualization (MaLIV) Research Group at Universiti Teknologi MARA (UiTM) Perak Branch. She received her Ph.D. in Information Technology and Quantitative Science from UiTM in 2015. Currently, she is an associate professor in Universiti Teknologi MARA. Her research fields are meta-heuristics search approach, machine learning, and educational technology. She can be contacted at email: suray078@uitm.edu.my.



Nor Faezah Mohamad Razi     is a senior lecturer in Universiti Teknologi MARA (UiTM). She is the Resource Person for Fundamentals of Research. She received her B.Sc. in Statistics and M.Sc. in Applied Statistics from UiTM in 2009 and 2012, respectively. Currently she pursues her Ph.D. study in decision science area. Her research interests in structural equation modeling (SEM), data envelopment analysis, and educational technology. She can be contacted at email: norfa122@uitm.edu.my.



Masurah Mohamad     is currently a senior lecturer at Universiti Teknologi MARA Perak Branch, Tapah Campus (UiTM Perak). She received her B.Sc. (Hons.) in Computer Science majoring in Software Engineering from Universiti Teknologi Malaysia (UTM) in 2004, and M.Sc. in Computer Science also from Universiti Teknologi Malaysia (UTM) in 2006, respectively. She received her Ph.D. in Computer Science field also from UTM in May 2021. She can be contacted at email: masur480@uitm.edu.my.



Abdullah Sani Abd Rahman     obtained his first degree in Informatique majoring in Industrial Systems from the University of La Rochelle, France, in 1995. He received a master's degree from Universiti Putra Malaysia in Computer Science, with specialization in distributed computing. Currently, he is a lecturer at the Universiti Teknologi PETRONAS, Malaysia. He can be contacted at email: sani.arahman@utp.edu.my.



Nor Aslily Sarkam     is a senior lecturer at the Universiti Teknikal Malaysia Melaka (UTeM). She obtained her Ph.D. in Management majoring in Statistics from Tohoku University, Japan, in 2018. She also earned both her master's degree and bachelor's degree in Statistics from Universiti Teknologi MARA (UiTM), Malaysia. Her research focuses on artificial intelligence, machine learning, structural equation modeling (SEM), data analytics, and predictive modeling. She can be contacted at email: aslily@utem.edu.my