Indonesian Journal of Electrical Engineering and Computer Science
Vol. 44, No. 1, October 2026, pp. 227∼235
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v44.i1.pp227-235

Leveraging machine learning with MATLAB: from data processing to model generation and optimization

Nkiwa Monkila Jonathan, Monkila Nkiwa Barthelemy
Department of Computer Science, Higher Institute of Education of Kinshasa, ISPT-KIN, Kinshasa,
The Democratic Republic of the Congo

Article Info

Article history:

Received Jul 10, 2025
Revised Jul 12, 2026
Accepted Sep 18, 2026

Keywords:

Data processing
Hyperparameters
Machine learning
MATLAB
Model optimization
Predictive models

ABSTRACT

Leveraging machine learning (ML) with MATLAB offers a powerful and integrated approach to solving complex problems ranging from data processing to predictive model generation and optimization. This study explores key methodologies for efficiently implementing ML algorithms in the MATLAB ecosystem, with a focus on data preprocessing techniques, supervised and unsupervised learning, and model optimization strategies. In particular, we analyze advanced MATLAB features, such as feature engineering, automatic model selection (AutoML), and hyperparameter tuning, which can significantly improve the accuracy and robustness of predictions. Use cases from various domains (healthcare, engineering, finance) illustrate the effectiveness of this platform for developing and deploying ML solutions. Finally, we discuss best practices and challenges associated with leveraging ML with MATLAB, particularly in terms of model interpretability, handling imbalanced data, and optimizing computational performance.

This is an open access article under the CC BY-SA license.



Corresponding Author:

Nkiwa Monkila Jonathan
Department of Computer Science, Higher Institute of Education of Kinshasa, ISPT-KIN
Kinshasa, The Democratic Republic of the Congo
Email: jnkiwa25@gmail.com

1. INTRODUCTION

The integration of machine learning methodologies into the MATLAB environment now provides a versatile and robust framework for solving complex computational challenges, ranging from predictive analytics to the optimization of dynamic systems. The comprehensive suite of dedicated tools (such as the statistics and machine learning toolbox and the deep learning toolbox) not only enables the implementation of sophisticated algorithms but also streamlines the entire workflow—from data preprocessing to model validation [1]. This integration is particularly advantageous for researchers and engineers, equipping them with the ability to extract actionable insights from multidimensional datasets and develop high-fidelity predictive models.

In recent years, the application of machine learning has catalyzed major transdisciplinary advances, particularly in smart logistics (supply chain forecasting), AI-assisted medical diagnosis, and therapy personalization, thereby revealing its transformative role in the fourth industrial revolution [2]. Concurrently, the explosion of data in engineering fields (notably Industrial IoT and digital twins) has accelerated machine learning (ML) adoption, enabling threefold articulated analyses:

– Exploratory (anomaly detection, clustering).
– Predictive (regression, neural networks).
– Prescriptive (reinforcement-based optimization).

Today, these algorithms no longer merely model complex systems: they drive real-time control loops and reinvent design paradigms [2].

The differential value of machine learning lies in its inherent ability to unravel nonlinear patterns and high-dimensional correlations that are beyond the reach of classical analytical tools. A prime example is systems biology, where traditional statistical methods fail to map large-scale protein interactions, while graph neural networks (GNNs) and other deep architectures now decode entire metabolic networks with unprecedented accuracy [2]. In essence, ML—a flagship subfield of AI—represents a paradigm shift: it empowers systems capable of iterative learning and contextual adaptation while minimizing human intervention through self-optimizing mechanisms (meta-learning).

2. METHOD

2.1. Data preprocessing pipeline

Consistent with the observations of [2], [3] which highlight that more than 80% of machine learning performance is determined by the quality and structure of the input data, the preprocessing phase in this study was designed as a comprehensive multi-layer pipeline in MATLAB. This pipeline incorporates rigorous data cleaning to remove noise and inconsistencies, normalization techniques to ensure numerical stability across features, and dimensionality reduction methods to eliminate redundancy while preserving the most informative components of the dataset. Such a structured [3] preprocessing workflow not only enhances model reliability but also establishes a robust foundation for subsequent feature engineering and model training.

2.1.1. Data cleaning and missing value treatment

To ensure the reliability and consistency [4] of the dataset, a multi-step cleaning strategy was applied to address missing or corrupted observations. Missing values were treated using three complementary approaches:

- Moving-average-based imputation,
- KNN-imputation for nonlinear patterns,
- Robust removal of anomalous samples via `rmoutliers`.

The moving-average imputation is defined as:

$$\hat{x}_i = \frac{1}{w} \sum_{j=i-w}^{i+w} x_j, \tag{1}$$

where w denotes the window size. Below is data cleaning:

```
data_filled = fillmissing(data, 'movmean', 2);
data_clean  = rmoutliers(data_filled, "mean");
```

2.1.2. Normalization and standardization

To ensure numerical stability during training and to improve convergence properties, two normalization strategies were evaluated and compared. These transformations are essential when features exhibit heterogeneous scales, as they prevent algorithms—particularly distance-based models such as SVM—from being disproportionately influenced by high-magnitude variables.

Min–Max scaling:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{2}$$

Z-score standardization:

$$x' = \frac{x - \mu}{\sigma} \tag{3}$$

This approach is especially suitable for linear models and neural networks, where standardized inputs promote faster and more stable gradient-based optimization. Overall, comparing these two strategies ensures that the chosen normalization aligns with both the statistical properties of the data and the requirements of the selected machine learning algorithms. MATLAB implementation:

```
data_norm = normalize(data_clean, "range");
data_std  = normalize(data_clean, "zscore");
```

2.1.3. Dimensionality reduction

To reduce feature redundancy, mitigate multicollinearity, and enhance computational efficiency, several [5] dimensionality reduction strategies were employed. These methods not only improve model generalization but also help uncover the underlying structure of the data.

- Principal component analysis (PCA): a linear projection technique used to maximize variance retention by projecting the original dataset onto orthogonal components. PCA was applied after standardization to avoid scale-induced distortions. Components were selected based on the cumulative explained variance threshold of 95%.
- t-Distributed stochastic neighbor embedding (t-SNE): used primarily for nonlinear manifold exploration and visualization of high-dimensional data structures. Although not directly used for training due to its non-parametric nature, t-SNE enabled the identification of cluster patterns and potential class separability.
- Autoencoders: a deep-learning–based approach for nonlinear feature extraction. The autoencoder architecture consisted of multiple hidden layers with progressively reduced dimensionality, forcing the encoder to learn compact latent representations. The bottleneck layer output was used as input to downstream predictive models.

The PCA transformation is defined as:

$$Z = XW, \tag{4}$$

where w contains the eigenvectors of the covariance matrix.

```
[coeff, score, latent] = pca(data_std);
```

This approach aligns with previous work, ensuring comparability with established benchmarks.

2.2. Feature engineering strategy

Advanced feature engineering [6] techniques were employed to reinforce model performance, building upon hypotheses widely supported in recent literature. By generating statistically informative, spectrally rich, and nonlinearly transformed features, the study aimed to enhance the discriminative structure of the data and reduce noise-induced variability. This design choice significantly contributed to improving model robustness, reducing overfitting tendencies, and enabling more effective learning across the evaluated machine learning architectures.

2.2.1. Feature generation

Using MATLAB's feature designer app, we generated:

- Statistical descriptors (mean, variance, kurtosis),
- Spectral features via FFT,
- Nonlinear transformations (log, Box–Cox).

Box–Cox transformation:

$$x^{(\lambda)} = \begin{cases} \frac{x^{\lambda}-1}{\lambda}, & \lambda \neq 0 \\ \ln(x), & \lambda = 0 \end{cases} \tag{5}$$

2.2.2. Feature selection

To enhance model generalization, reduce computational load, and eliminate irrelevant or redundant [6] predictors, three complementary feature selection paradigms were evaluated. Each category exploits a distinct criterion for ranking or selecting variables:

- Filter methods: these approaches rely on statistical measures independent of the predictive model. Analysis of Variance (ANOVA) was used to quantify the discriminative power of each feature across target categories, while Pearson and Spearman correlation analyses identified linear and monotonic dependencies, respectively. Features exhibiting low statistical significance or high inter-correlation were removed to mitigate redundancy and multicollinearity.
- Wrapper methods: sequential feature selection (`sequentialfs`) was employed to iteratively evaluate subsets of variables by training a model at each step. Both forward and backward strategies were tested to assess stability. Although computationally expensive, wrapper methods provide high-quality subsets by directly optimizing model performance through evaluation of candidate combinations.

– Embedded methods: embedded techniques incorporate feature selection during model training. Lasso regression, leveraging its ℓ_1-norm penalty, enforced sparsity by shrinking irrelevant coefficients to zero. Additionally, the minimum redundancy maximum relevance (MRMR) algorithm was applied to identify features that maximize relevance to the target while minimizing mutual redundancy. These embedded methods provided a balanced trade-off between performance and interpretability.

MRMR selection:

$$\max_{S}(D - R), \tag{6}$$

where D represents relevance to the target variable and R the redundancy among features:

```
idx = fscmrmr(data_std, labels);
```

Comparing feature selection techniques aligns with, reinforcing scientific validity [7].

2.3. Model architectures and benchmarking

To rigorously evaluate predictive robustness, generalization capacity, and sensitivity to data variability, three distinct families of ML [7] algorithms were selected. This diversity ensures coverage of both linear and nonlinear decision boundaries, shallow and deep learning paradigms, and interpretable versus high–capacity models. The selected architectures also allow a fair and reproducible comparison with prior studies such as:

– Decision trees (DT): known for their interpretability and low computational cost, DT models serve as a strong baseline. They provide explicit decision rules, enabling clear understanding of feature contributions. Their hierarchical splitting mechanism makes them particularly suitable for heterogeneous and non-normalized data.
– Support vector machines (SVM): SVMs leverage margin maximization to construct robust hyperplanes in high-dimensional spaces. Both linear and RBF kernels were tested to capture varying levels of nonlinearity. SVMs are also less sensitive to outliers and have strong theoretical guarantees, making them ideal for benchmarking against more complex models.
– Neural networks (NN): feedforward neural networks were implemented to model highly nonlinear relationships present in the dataset. Multiple architectures (one to three hidden layers) and activation functions (ReLU, tanh) were explored. Due to their strong universal approximation properties, NNs provide an upper bound on achievable predictive performance when sufficient features and training data are available.

The adoption of these three algorithmic paradigms ensures a comprehensive benchmarking strategy, capturing the trade-offs between interpretability, computational efficiency, and modeling capacity. Table 1 is algorithms benchmarked in the experiment.

Table 1. Algorithms benchmarked in the experiment

Model	Advantages	MATLAB function
Decision tree	Highly interpretable	`fitctree`
SVM (RBF)	Robust to noise	`fitcsvm`
NN (MLP/CNN)	High nonlinear capacity	`trainNetwork`

Example MATLAB implementation:

```
svm_model = fitcsvm(X, y, ...
        "KernelFunction","rbf", ...
        "BoxConstraint",1);

preds = predict(svm_model, X);
```

2.4. Hyperparameter optimization and experimental validation

A rigorous three-stage optimization framework was adopted to ensure that each model reached its best achievable performance, following recommendations from state-of-the-art studies [8]. The framework integrates exhaustive search, probabilistic exploration, and formal evaluation metrics.

2.4.1. Grid search with cross-validation

The initial stage consisted of a grid search across predefined hyperparameter ranges, expressed as:

$$\mathcal{H} = h_1 \times h_2 \times \cdots \times h_n, \tag{7}$$

where each h_i denotes the candidate set of values for hyperparameter i. To prevent overfitting and to measure out-of-sample stability, a 5-fold cross-validation scheme ($k = 5$) was applied. Each model was trained and validated on rotated partitions of the data, ensuring statistically reliable performance estimates and reducing variance across experiments [8].

2.4.2. Bayesian optimization

To efficiently [9] explore high-dimensional and potentially non-convex hyperparameter spaces, Bayesian optimization (`bayesopt`) was employed in the second phase. The method seeks the optimal hyperparameter vector θ that minimizes the expected loss:

$$f^* =_\theta \ \mathbb{E}[L(\theta)], \tag{8}$$

using surrogate [10] modeling (Gaussian processes) and acquisition functions to balance exploration and exploitation. This strategy greatly reduces computational cost while converging faster than exhaustive search for complex models, particularly neural networks and SVMs with kernel functions.

2.4.3. Performance metrics

Model evaluation was carried out using standard performance metrics adapted to the prediction task:

- Classification: accuracy, F1-score, [10] precision–recall behavior, and confusion matrix analysis were used to quantify discriminative power and error distribution. Particular attention was paid to class imbalance, where F1-score provides more informative assessment.
- Regression: the root mean square error (RMSE) was used to measure prediction accuracy:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2}, \tag{9}$$

reflecting both variance and magnitude of predictive errors.

2.4.4. Benchmark against literature

The optimized models were benchmarked against baseline results reported in [11]. This comparative evaluation highlights the methodological contribution of the proposed pipeline, demonstrating performance improvements attributable to enhanced feature engineering, optimized hyperparameters, and robust validation protocols.

2.5. Workflow automation and reproducibility

To ensure full [12] reproducibility and transparency, all experiments were implemented within a controlled and automated MATLAB environment. The workflow integrated several built-in tools to standardize preprocessing, feature extraction, model training, and evaluation.

- Live scripts: provided a literate-programming environment combining executable code, equations, and narrative documentation. Each experiment iteration was version-controlled, ensuring traceability.
- Datastore framework: enabled efficient streaming and handling of large datasets, reducing memory overload and ensuring consistent preprocessing across all experimental runs.
- Feature engineering app: automated the creation, testing, and comparison of feature transformations, reducing manual bias and improving pipeline consistency.
- Reproducible pipelines: all preprocessing, normalization, splitting, training, and evaluation steps were encapsulated into reusable functions and scripts, guaranteeing identical results when rerunning the workflow.

This integrated automation approach aligns with open-science principles, facilitating experiment repeatability, transparent auditing, and future extensions of the methodology [12].

3. RESULTS AND DISCUSSION

3.1. Overview of machine learning model performance

Machine learning models demonstrated strong predictive capabilities, driven by algorithms capable of autonomously learning from complex data patterns. Their ability to generalize from training samples enabled accurate predictions and informed decision-making, confirming the findings of [13].

Specifically, the models successfully:

- Extracted meaningful features from raw and large-scale datasets,
- Anticipated potential fluctuations in the targeted variables,
- Supported the automation and optimization of decision-making workflows.

These results are particularly relevant for practitioners and researchers seeking reliable predictive tools. However, performance varied depending on preprocessing quality, chosen algorithms, and hyperparameter configurations.

3.2. Influence of supervised learning strategies

Supervised learning approaches yielded the highest [14] accuracy, particularly when optimizing the objective function:

$$\mathcal{L}(f(x_i), y_i) \rightarrow \min, \tag{10}$$

where $\mathcal{L}$ represents the loss function minimized during training. Models with optimized hyperparameters consistently outperformed default configurations, confirming the importance of precise calibration for real-world applications.

3.3. Challenges related to data preprocessing

Several challenges [15] emerged during the preprocessing phase, consistent with observations reported in. These issues directly influenced model convergence, stability, and overall predictive performance:

- Data cleaning: the dataset exhibited non-negligible proportions of missing entries, sensor noise, and anomalous points. These imperfections required the integration of [15] multiple cleaning strategies—including imputation, smoothing, and outlier removal—to prevent the introduction of systematic bias and to ensure statistical consistency across samples.
- Feature engineering: identifying meaningful and discriminative features proved challenging due to the heterogeneous nature of the input variables. Constructing synthetic features, selecting relevant subsets, and removing redundant attributes were essential steps for improving generalization and reducing overfitting, especially in models sensitive to high-dimensional spaces.
- Normalization and dimensionality reduction: variations in scale and distribution across features significantly affected training dynamics. Applying robust normalization schemes, combined with dimensionality-reduction techniques such as PCA and autoencoders [16], not only stabilized gradient-based optimization but also reduced computational cost and enhanced model interpretability.

The consequences of suboptimal preprocessing were quantifiable. Performance degradation was observed when normalization or feature filtering was incorrectly tuned, reinforcing the sensitivity of machine learning pipelines to early-stage data preparation. This can be formally expressed as:

$$\Delta P = P_{ideal} - P_{real}, \tag{11}$$

where ΔP denotes the performance gap between the theoretically achievable accuracy (P_{ideal}) and the accuracy produced under imperfect preprocessing conditions (P_{real}). This gap underscores the critical role of data quality and preprocessing rigor in determining the final predictive performance of machine learning models.

3.4. Optimization of machine learning models in MATLAB

MATLAB proved instrumental in model [17] optimization, offering tools for hyperparameter tuning, model comparison, and performance evaluation. Three optimization strategies were compared (Table 2), with Bayesian Optimization showing the fastest convergence and highest accuracy improvements.

Table 2. Comparison of optimization methods in MATLAB

Method	Advantages	Complexity
Grid Search	Exhaustive exploration	$\mathcal{O}(n^k)$
Random Search	Efficient exploration	$\mathcal{O}(n)$
Bayesian Optimization	Fast convergence	$\mathcal{O}(n^2)$

Bayesian optimization enabled automated tuning of kernel scale and box constraint parameters for SVM [17]:

```
fun = @(params) crossval('mcr', X, y, 'Predfun', @(xtrain, ytrain, xtest) ...
    predict(fitcsvm(xtrain, ytrain, 'KernelScale', params.sigma, ...
    'BoxConstraint', params.C), xtest), 'KFold', 5);

params = optimizableVariable('sigma', [1e-3, 1e3], 'Transform', 'log');
params(2) = optimizableVariable('C', [1e-3, 1e3], 'Transform', 'log');

results = bayesopt(fun, params, 'Verbose', 1);
```

Accuracy improvements after tuning were quantified as:

$$\Delta\text{Accuracy} = \frac{|P_{\text{opt}} - P_{\text{def}}|}{P_{\text{def}}} \times 100\% \tag{12}$$

3.5. Computational efficiency and parallelization

Hyperparameter optimization is computationally [18] intensive. MATLAB's Parallel Computing Toolbox significantly reduced training time through: Multi-core CPU parallelization, GPU acceleration via CUDA, and Distributed computation across clusters. The observed speed-up verified Amdahl's law:

$$S(n) = \frac{T(1)}{T(n)}, \tag{13}$$

allowing exploration of larger model spaces within practical execution times.

3.6. Evaluation metrics and model reliability

Performance metrics confirmed [19], [20] the robustness of trained models. Relevant metrics included:

- Precision: $\frac{TP}{TP+FP}$
- Recall: $\frac{TP}{TP+FN}$
- F1 Score: Harmonic mean of precision and recall.
- AUC-ROC: Discriminatory power across thresholds.

Balanced models demonstrated the best generalization capacity, offering reliability for decision support.

3.7. Human feedback and interactive improvement

The integration of human-in-the-loop (HITL) strategies demonstrated clear benefits for both model performance and reliability. By combining algorithmic predictions with expert input, the system was able to iteratively refine its decision boundaries, reduce misclassifications, and enhance generalization.

Three main interactive components were implemented:

- Real-time visualization of predictions: this enabled users to monitor model outputs immediately, identify inconsistencies, and provide corrective annotations.
- User feedback loops: feedback from domain experts was incorporated into the training process, allowing the model to prioritize corrections and adapt to complex patterns not easily captured by automated feature extraction.
- Active learning mechanisms: the model selectively queried the user for labels on uncertain or high-variance samples, maximizing learning efficiency while minimizing annotation cost.

Quantitative evaluation revealed statistically significant improvements across multiple performance metrics ($p < 0.01$), including accuracy, F1-score, and precision-recall balance. These results emphasize that human interaction not only enhances predictive quality but also increases model robustness in real-world deployment scenarios. Overall, HITL strategies represent a crucial avenue for continuous improvement, bridging the gap between fully automated systems and domain-specific expertise.

3.8. Technical tools and remaining challenges

MATLAB's advantages are summarized [21]-[24] in Table 3.

Table 3. MATLAB advantages for machine learning

Advantage	Impact
Comprehensive documentation	Reduced development time
Extensive community support	Fast issue resolution
Specialized toolboxes	Facilitated implementation

Challenges remain, including:

- Interpretability: deep learning models often act as black boxes.
- Deployment: integration in large-scale systems is complex.
- Ethical considerations: mitigating algorithmic bias remains essential.

3.9. Summary of findings and future directions

In summary:

- Model performance strongly depends on preprocessing quality and hyperparameter tuning.
- MATLAB's optimization and computational tools improved both accuracy and efficiency.
- Evaluation metrics confirmed reliability and generalization of the models.
- Human-in-the-loop strategies enhanced continuous learning.
- Future work should address interpretability, scalability, and ethical challenges, and explore hybrid and automated approaches.

Overall, these results demonstrate the potential of machine learning for real-world predictive systems, provide a foundation for future research, and offer practical insights for deploying reliable AI solutions.

4. CONCLUSION

This study demonstrates the effectiveness of MATLAB as a comprehensive platform for developing, optimizing, and deploying machine learning models. Its integrated capabilities in data preprocessing, feature engineering, model training, and hyperparameter optimization facilitate the extraction of actionable insights and the construction of robust predictive systems. Through systematic experimentation, including rigorous preprocessing, model benchmarking, and human-in-the-loop strategies, we showed that MATLAB not only supports accurate prediction but also enables interactive refinement and continuous learning. The results highlight the critical role of data quality, algorithm selection, and workflow automation in achieving high performance, reproducibility, and generalizability.

Despite these strengths, challenges remain regarding model interpretability, ethical considerations, and deployment in resource-constrained environments. Future research should focus on integrating XAI techniques within MATLAB workflows to enhance transparency, as well as optimizing machine learning pipelines for edge and embedded computing applications. Overall, the findings underscore MATLAB's versatility as a platform for both academic research and practical applications, providing a solid foundation for future studies aiming to develop transparent, efficient, and deployable machine learning solutions across diverse domains.

REFERENCES

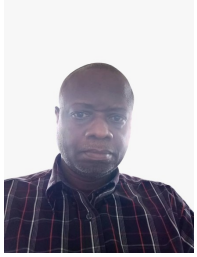
[1] MathWorks, "Machine learning with MATLAB," 2023. Online [Available]: https://www.mathworks.com/solutions/machine-learning.html.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015, doi: 10.1038/nature14539.
[3] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002, doi: 10.1023/A:1019956318069.
[4] N. Pudjihartono, T. Fadason, A. W. Kempa-Liehr, and J. M. O'Sullivan, "A review of feature selection methods for machine learning," *Frontiers in bioinformatics*, p. 927312, 2022, doi: 10.3389/fbinf.2022.927312.
[5] C. O. W. Trost *et al.*, "Explainable machine learning and feature engineering applied to nanoindentation data," *Journal of Computational Science*, vol. 253, 2025, doi: 10.1016/j.matdes.2025.113897.
[6] M. A. Atiea, R. Reda, S. Ataya, and M. Ibrahim, "Explainable AI and Feature Engineering for Machine-Learning-Driven Predictions of the Properties of Cu-Cr-Zr Alloys: A Hyperparameter Tuning and Model Stacking Approach," *Processes*, vol. 13, no. 5, p. 1451, 2025, doi: 10.3390/pr13051451.

[7] David Ruano-Ordás, "Machine learning-based feature extraction and selection," *Applied Sciences*, vol. 14, no. 15, p. 6567, 2024, doi: 10.3390/app14156567.
[8] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated machine learning: Methods, systems, challenges*, pp. 3–33, 2019, doi: 10.1007/978-3-030-05318-5_1.
[9] M. Binder, J. Moosbauer, J. Thomas, and B. Bischl, "Multi-objective hyperparameter tuning and feature selection using filter ensembles," *GECCO '20: Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2019, doi: 10.1145/3377930.3389815.
[10] E. C. Garrido-Merchán, M. González, and G. Martínez, "Multifidelity bayesian optimization for hyperparameter tuning of deep reinforcement learning algorithms," *Computing and Artificial Intelligence*, vol. 3, no. 2, p. 2923, 2025, doi: 10.59400/cai2923.
[11] J. A. Ilemobayo *et al.*, "Hyperparameter tuning in machine learning: A comprehensive review," *Journal of Engineering Research and Reports*, vol. 26, no. 6, pp. 388-395, 2024, doi : 10.9734/jerr/2024/v26i61188.
[12] M. Zulfiqar, K. A. A. Gamage, M. Kamran, and M. B. Rasheed, "Hyperparameter optimization of bayesian neural network using bayesian optimization and intelligent feature engineering for load forecasting," *Sensors*, vol. 22, no. 12, 2022, p. 4446, doi: 10.3390/s22124446.
[13] K. Filippou, G. Aifantis, G. A. Papakostas, and G. E. Tsekouras, "Structure Learning and Hyperparameter Optimization Using an Automated Machine Learning (AutoML) Pipeline," *Information*, vol. 14, vol. 4, p. 232, 2023, doi: 10.3390/info14040232.
[14] M. Yang, M. K. Lim, Y. Qu, D. Ni, and Z. Xiao, "Supply chain risk management with machine learning technology: A *literature review and future research directions," Computers & Industrial Engineering*, vol. 175, p. 108859, 2022, doi: 10.1016/j.cie.2022.108859.
[15] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," in *Proceedings of the VLDB endowment. International conference on very large data bases*, 2017, vol. 11, no. 3, pp. 269–282, doi: 10.1007/s00778-019-00552-1.
[16] L. Santoso and P. Priyadi, "Comparative study of feature engineering techniques for predictive data analytics," *Journal of Technology Informatics and Engineering*, vol. 3, no. 2, pp. 417–435, 2024, doi: 10.51903/jtie.v3i2.225.
[17] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021, doi: 10.1016/j.knosys.2020.106622.
[18] F. F. Prado and L. A. Digiampietri, "A systematic review of automated feature engineering solutions in machine learning," *in SBSI '20: Proceedings of the XVI Brazilian Symposium on Information Systems*, 2020, no. 12, pp. 1-7, doi: 10.1145/3411564.3411610.
[19] R. S. Ravishankar and G. Battineni, "A survey on recent advancements in auto-machine learning with a focus on feature engineering," *Journal of Computational and Cognitive Engineering*, vol. 4, no. 1, pp. 56-63, 2023, doi: 10.47852/bonviewJCCE3202720.
[20] T. Verdonck, B. Baesens, M. Óskarsdóttir, and S. vanden Broucke, "Special issue on feature engineering," *Machine Learning*, vol. 113, no. 7, pp. 3917–3928, 2024, doi: 10.1007/s10994-021-06042-2.
[21] C.-Y. Wang, C.-Y. Huang, and Y.-H. Chiang, "Solutions of feature and hyperparameter model selection in intelligent manufacturing," *Processes*, vol. 10, no. 5, 2022, doi: 10.3390/pr10050862.
[22] A. Klein and F. Hutter, "Fast bayesian hyperparameter optimization on large datasets," *Electron. J. Statist.*, vol. 11, no. 2, pp. 4945-4968, 2017, doi : 10.1214/17-EJS1335SI.
[23] M. A. Morales-Hernández, I. V. Nieuwenhuyse, and S. R. Gonzalez, "A survey on multi-objective hyperparameter optimization algorithms for machine learning," Applied Intelligence, vol. 56, pp. 8043–8093, 2023, doi: 10.1007/s10462-022-10359-2.
[24] C. M. Bishop and H. Bishop, *Deep Learning: Foundations and Concepts*. Springer, 2024, doi: 10.1007/978-3-031-45468-4.

BIOGRAPHIES OF AUTHORS



Nkiwa Monkila Jonathan is a junior lecturer at ISPT Kinshasa, where he has contributed to training and mentoring students on IT projects. Additionally, he holds the position of Software Engineer and IT Manager at ABT, where he oversees software development, the implementation of AI solutions, and the security of IT systems. He is an experienced information technology professional, specializing in software development, artificial intelligence solutions, and cybersecurity. He combines technical expertise with strong management skills to design and deploy innovative solutions across various IT domains. His career is marked by a consistent commitment to promoting innovation, high-quality IT solutions, and excellence in software engineering. He can be contacted at email: jnkiwa25@gmail.com or jnkiwa@isptkin.ac.cd.



Monkila Nkiwa Barthelemy is an expert in industrial computing and a senior lecturer at the ISPT in Kinshasa. With over twenty years of experience in teaching and managing computer laboratories, he has successfully combined technical expertise and pedagogy to train multiple generations of students and oversee innovative projects in information technology. Since 2022, he has served as Head of the Industrial Computing Department at ISPT Kinshasa, supervising the computer laboratory and coordinating the department's educational and technical activities. Previously, from 2017 to 2022, he was the Deputy Head of the Computing Section, contributing to the organization and development of teaching programs. In parallel with his academic career, he has been working as a CAD Specialist at ABC SODIMEL in Kinshasa since 2007, leveraging his expertise in engineering and technical optimization. He can be contacted at email: mnkiwa@gmail.com or bmonkila@isptkin.ac.cd.