Journal of Algorithms & Computational Technology

Original research article

Optimised Parallelised Ensemble Learning (OPEL): A framework for OPEL

Journal of Algorithms & Computational Technology
Volume 20: 1–18

DOI: 10.1177/17483026261429883
journals.sagepub.com/home/act
Sage

Jephter Kapika Pelekamoyo[1,2], Hastings M Libati[1] and Derrick Ntalasha[1]

Abstract

This study presents a novel Optimised Parallelised Ensemble Learning (OPEL) framework that enhances multi-ensemble learning through a unique combination of Parallel multi-Model Execution, Consensus-Based Model Selection (CMS), and an Optimised Parallel Voting Mechanism. Together, these components significantly reduce computational complexity, as analytically supported by Amdahl's Law, while enhancing model robustness by dynamically varying participating voting models for any varied sample sizes through model selection, weighting, and parallel execution strategies. Performance metrics utilised selected top-performing models, achieving speed-ups of up to 1.3 ms for some samples and higher accuracy scores. These results validate OPEL as a scalable, efficient, and high-performing approach for ensemble learning in resource-constrained and high-throughput applications. Unlike existing methods such as Auto-ML or A-Stacking, OPEL's real-time dynamic model selection and multi-model parallel execution significantly show improved accuracy. Experimental simulations on real-world datasets demonstrated significant improvement of around 5.6% in model accuracy on weather-based sales prediction datasets and had a win rate of 60.64%, unlike Auto-ML for the hotel booking predictions, using McNemar's analysis. A paired *t*-test confirmed the statistical significances of these improvements, proving OPEL to be a scalable, adaptive ensemble framework for real-time applications that demanded both speed and accuracy by selecting and re-weighting models dynamically during runtime based on live performance metrics, offering dynamic and computationally efficient system as compared to traditional methods, validated across classification tasks involving SME market sales and hotel booking datasets. OPEL's novel contribution lies in its run-time optimised voting and parallel selection mechanism, making it suitable for dynamic-non-stationary environments.

Keywords

Ensemble learning, parallelisation, model selection, optimised voting, computational efficiency, machine learning

Received: 9 April 2025; accepted: 18 February 2026

Introduction

In real-time market sales monitoring, traditional ensemble methods fail to meet latency requirements due to sequential execution.[1] For example, predicting maize sales in Zambia's markets using weather patterns,[2,3] requires a lot of time and planning for which models to use. This article introduces a novel advanced Optimised Parallelised Ensemble Learning (OPEL) framework, designed to improve ensemble learning through parallel execution and dynamic model selection for determining the final result(s), which is a new approach for enabling parallel multi-model execution and selection for real-time predictions. Unlike traditional approaches that rely on sequential execution and static voting schemes, OPEL integrates Parallelised Multi-Model Execution (PME) to accelerate training and evaluation, Consensus-Based Model Selection (CMS) to dynamically identify the most effective models,

[1]Department of Information and Communication Technology, The Copperbelt University Kitwe, Kitwe, Zambia
[2]Department of Information and Communication Technology, Kapasa Makasa University, Chinsali, Zambia

Corresponding authors:
Hastings M Libati, Department of Information and Communication Technology, The Copperbelt University Kitwe, Kitwe, Zambia.
Email: libati@cbu.ac.zm
Derrick Ntalasha, Department of Information and Communication Technology, The Copperbelt University Kitwe, Kitwe, Zambia.
Email: derick.ntalasha@cbu.ac.zm

and an Optimised Parallel Voting Mechanism (OPVM) to refine ensemble decision-making.

The proposed framework's uniqueness lies in its ability to adapt ensemble composition on the fly, using confidence-based model metrics, and its optimisation of voting weights during execution capabilities not present in traditional parallel or ensemble learning techniques. Traditional ensemble learning methods, such as bagging and boosting, have been widely used to enhance predictive accuracy in machine learning but rely on static voting and sequential execution, which limits performance in latency-sensitive, resource-constrained environments and requires extensive computational resources, particularly when applied to large-scale and distributed datasets. As machine learning applications continue to expand into areas like real-time decision-making and resource-constrained environments, the need for efficient, scalable ensemble learning frameworks has become critical, often suffering from scalability and efficiency issues, especially in large-scale, heterogeneous settings and when coupled with fast-changing data. Recent works[4] have applied parallelism in ensemble contexts but often lack dynamic model selection based on real-time performance metrics. To address this gap, we propose OPEL, a framework that not only runs ensemble models in parallel but dynamically ranks and selects models using a live confidence metric (e.g. Matthews Correlation Coefficient (MCC)) and adjusts voting weights during execution. This ensures scalability and robustness in distributed environments such as market forecasting or federated healthcare systems.

Existing ensemble approaches are computationally intensive, difficult to scale in distributed environments, and often fail to dynamically adapt to rapidly changing or heterogeneous data contexts. In particular, sequential execution leads to high latency, undermining real-time deployment and static model voting limits responsiveness to performance variability across models. Furthermore, resource demands make them unsuitable for constrained or distributed computing environments.

These limitations hinder the broader applicability of ensemble methods in modern AI systems that demand fast, flexible, and scalable architectures.

To address these challenges, this article proposes a novel framework titled OPEL. OPEL introduces a parallel execution architecture combined with real-time model selection and adaptive ensemble voting, aimed at improving both the computational efficiency and predictive performance of ensemble learning in resource-constrained, real-time environments. The proposed framework addresses these challenges by introducing a multi-model selection mechanism that identifies the best-performing models within a given set of classical machine learning models and ensemble learning models alike, coupled with parallel processing techniques to expedite the computation processes. An optimised weighting algorithm is employed to ensure that models contributing most effectively to the task at hand are prioritised, thereby improving the robustness of the final decision. Experimental evaluations demonstrate that the proposed method outperforms conventional ensemble approaches in terms of both speed and accuracy, particularly in distributed computing environments.

Currently, it's difficult to efficiently combine multiple machine learning models to improve decision-making accuracy and performance in scenarios where data is distributed or computational resources are limited, in a timely and efficient manner. Traditional ensemble methods like Random Forests or Ada Boosting rely on training and aggregating multiple models, which can be computationally expensive and difficult to scale in distributed systems or with large datasets. Furthermore, existing methods often lack robustness in selecting the best-performing models in heterogeneous environments, where different models may excel in different aspects of the task. But with the algorithm introduced in this study, it's now possible to assign appropriate weights to models in the ensemble based on their performance while they are running, ensuring that the most reliable models have a greater influence on the final decision. The study also conducted a comprehensive evaluation by comparing the proposed framework with traditional methods to demonstrate its advantages in terms of scalability, efficiency, and accuracy. Finally, the proposed ensemble framework was tested by measuring its performance in terms of accuracy, speed, and market user satisfaction.

The study contributes to knowledge through the introduction of a parallelised execution framework that significantly reduces computational time while maintaining or improving model performance, a dynamic model selection mechanism that adapts to real-time data variations, optimising ensemble composition and a statistically validated voting mechanism that enhances accuracy through performance-based weighting. These contributions to the growing body of research on scalable machine learning offer a practical solution for real-world applications where computational resources are limited for big data. The framework advances the field of machine learning and artificial intelligence by providing a scalable, efficient, and robust alternative to traditional ensemble methods, making it particularly well-suited for modern distributed computing challenges.

However, most existing traditional methods, such as bagging, boosting, or even distributed XGBoost, face key limitations when it comes to real-time or resource-constrained environments, as they typically rely on static model sets, fixed-weight voting, or delayed aggregation. This is because they often lack real-time adaptability to changing data, do not optimise model voting during execution, and fail to leverage parallel computation efficiently. These shortcomings make them less adaptive to fast-changing data streams or limited processing environments. While Random Forests and AdaBoost offer performance improvements, they equally rely on static model sets and fixed voting schemes, making them unsuitable for highly dynamic environments, while OPEL addresses the limitations by combining runtime model selection, adaptive voting, and parallel execution into a

unified, scalable framework designed for environments with non-stationary data and resource constraints.

The OPEL framework addresses the gaps of basic ensemble and other classification schemes by using dynamic run-time, real-time CMS coupled with real-time OPVM, which adapts model participation and voting weights during execution. This integration of run-time adaptability and parallelism marks OPEL's theoretical contribution, transforming ensemble learning from a static pipeline into a responsive, high-speed model aggregation system. Thereby answering the question of whether ensemble learning can be made both dynamic and scalable, optimising accuracy and speed in real time for changing data conditions.

This research contributes to the advancement of scalable machine learning by presenting a generalisable and efficient ensemble architecture that is particularly suited for distributed and real-time AI applications. It addresses a critical gap in the field by enabling: Real-time parallel model execution, dynamic adaptation to heterogeneous data streams, and Intelligent model aggregation based on real-time performance.

Experimental results demonstrate that OPEL achieves a 1.38× speedup and a 1.5× accuracy win rate improvement over some advanced traditional ensemble approaches, confirming its potential for deployment in scenarios where time and resource constraints are significant. Ultimately, this study provides a robust and practical framework for improving predictive analytics in domains ranging from agriculture to e-commerce and public health.

This study aims to design and implement the OPEL framework incorporating PME, CMS, and OPVM. Evaluate the framework using real-world datasets (e.g. weather-based sales data from Zambian markets). Compare OPEL's performance against traditional ensemble methods in terms of accuracy, computational speed, and scalability. Statistically validate the significance of observed improvements using paired *t*-tests and complexity analysis (e.g. via Amdahl's Law).

Related works

From the various previous studies reviewed, it's evident that most have focused on various optimisation strategies, including weighted voting mechanisms, probabilistic model selection and hierarchical learning approaches. However, these methods do not explicitly incorporate integrated models' parallel execution within the model selection and voting process. OPEL addresses this gap by building upon these foundations by integrating parallelisation directly into model execution, selection, and voting, making it particularly well-suited for distributed computing environments.

Parallelisation techniques

OPEL uses parallel methods on different machine learning models to enhance computational efficiency and scalability. These methods focus on reducing execution time, balancing workload, and managing communication overhead. This article reviewed some foundational and contemporary works that have contributed to the development of parallel computing strategies for model training and data processing.

For instance, while Amdahl's work is critical for this current work, particularly Amdahl's Law (1967), which provides a fundamental understanding of the limitations of parallel processing by quantifying the maximum speedup achievable when only part of a task is parallelised, while considering that some parts of the task must remain serial. This law underscores the inherent limitations in achieving significant performance gains through parallelisation, particularly when a substantial portion of a task cannot be parallelised, and requires parallel model aggregation in machine learning apply these scalability principles to enhance model performance, particularly in handling large datasets and complex models,[5–7] Amdahl's law is critical in understanding the limits of parallelisation; although it does not extend to machine learning, it still provides critical insights into the limits of parallel computation. These limitations are extended in this study through applications to specific contexts of machine learning model selection and weighted voting mechanisms.

Agarwal et al. (2023) accelerate the automatic detection of hate speech on social media platforms by implementing parallelising bagging, A-stacking, and random subspace algorithms. They evaluated the serial and parallel versions of the machine learning models on standard high-dimensional hate speech datasets, and the parallel models demonstrated a substantial increase in speed with remarkable efficiency, affirming that the proposed models are well-suited for this particular application. They observed that parallelising the algorithms does not compromise the accuracy compared to running machine learning ensemble algorithms sequentially on a single machine.[8]

Teh et al. (2006) introduce hierarchical models that allow for sharing statistical strength across different groups of data. The authors leverage Bayesian nonparametrics to build a flexible model that can be parallelised across clusters.[9] The method allowed for a more nuanced model that could capture complex dependencies within the data, and parallelisation improves scalability.

Cortes and Vapnik (1995) developed Support Vector Machines (SVMs) as a method for finding the optimal hyperplane that separates data into different classes, maximising the margin between classes.[10] Zanghirati and Zanni (2003) explore the parallelisation of SVM training using quadratic programming, significantly reducing the computational time for large datasets,[11,12] The study used a parallel decomposition technique to solve the quadratic programming problem in SVM training, distributing the workload across multiple processors.[11] Their technique significantly reduced training time for large datasets by parallelising the optimisation process. Their working principle is similar to the one proposed in this paper. But instead of parallelising SVMs alone, the current method integrates a

voting mechanism and equally focuses on a more generalised framework applicable across different models.

Dean et al. (2012) present a method for distributed training of deep neural networks through model parallelism, where different segments of a neural network are distributed across multiple machines. This approach enables the handling of extremely large datasets and models, facilitating the training of deep networks with billions of parameters. Their study demonstrated the scalability of deep learning systems and laid the groundwork for practical.[13]

Chu et al. (2006) introduced the MapReduce framework, using a parameter server architecture to efficiently scale distributed machine learning models across multiple servers, optimising both storage and computation, allowing for large-scale machine learning tasks to be handled more effectively in a distributed environment. Their framework utilised data distribution and parallel computation, making it a foundational method for processing vast datasets in a distributed manner.[14] Similarly, Li et al. (2014) used a parameter server architecture to efficiently scale distributed machine learning models across multiple servers, optimising both storage and computation. This facilitated the parallel training of machine learning models.[15] This approach significantly improves the scalability of machine learning training by efficiently handling parameter updates across distributed systems, but it introduces latency and synchronisation issues, particularly in highly distributed systems with non-uniform communication speeds.

Cole and Vishkin (1986) proposed a 'Theoretical Parallel Model', the development of deterministic algorithms for parallel computation, including techniques for reducing contention and improving efficiency.[16] Cole and Vishkin (1986) developed deterministic algorithms for parallel computation, emphasising techniques to reduce contention among processors and enhance overall computational efficiency. Their work is instrumental in the creation of parallel algorithms that operate under strict deterministic conditions, ensuring consistent and predictable performance across different computational tasks,[16] providing essential insights into the development of deterministic parallel algorithms, but it does not extend these principles to machine learning or model aggregation.

Graham (1966) worked on load-balancing issues in parallel computation, addressing the inefficiencies that arise when tasks are not evenly distributed across processors. The primary focus is on ensuring that each processor in a parallel computing environment is utilised effectively to avoid bottlenecks that can occur when tasks are not evenly distributed.[17] His work was further amplified by Brent (1974), who offered a fundamental analysis of the efficiency of parallel algorithms, concentrating on minimising communication overhead and ensuring effective load balancing across processors, and established key principles for optimising parallel computation, particularly by reducing the time complexity of parallel algorithms and ensuring that tasks are distributed in a manner that maximises processor utilisation.[18] Karp and Ramachandran (1990) further comprehensively examined parallel algorithms, particularly within the context of shared-memory architecture.[19]

Shalev-Shwartz et al. (2011) introduced the Pegasos algorithm, a stochastic sub-gradient descent method for efficiently training SVMs. The algorithm was particularly notable for its scalability, making it well-suited for handling large datasets. The Pegasos algorithm significantly reduces the computational complexity of SVM training, providing a more practical solution for real-world, large-scale machine-learning tasks.[6]

Zhang et al. (2013) proposed a divide-and-conquer approach for scaling kernel ridge regression on large datasets by splitting data into smaller subsets and processing subsets independently in parallel before combining the results, solving the problem on each subset, and then combining the results.[20] According to Zhang et al. (2013), for finite-rank kernels and Gaussian kernels, their theory ensured that the number of processors, denoted as m, can increase almost linearly; for Sobolev spaces, the number of processors can grow polynomially with N. The partitioning led to a substantial reduction in computation time and cost.[20]

Kumar and Gupta's (1994) study provides a comprehensive analysis of the scalability of parallel algorithms across various computing architectures, focusing on shared memory, distributed memory, and hybrid systems. Their work emphasises the importance of load balancing and minimising communication overhead to optimise scalability, offering a strong theoretical foundation for parallel computation. However, the study lacks a focus on machine learning-specific applications, such as model selection and ensemble voting, and some of the discussed architectures are now outdated. In contrast, modern approaches to parallel model aggregation in machine learning apply these scalability principles to enhance model performance, particularly in handling large datasets and complex models.[5] While Kumar and Gupta's work is foundational, contemporary methods extend these concepts to address the unique challenges of machine learning in distributed environments.

Ensemble aggregation strategies

Besides parallelisation, different ensemble aggregation tactics aim to combine the predictions of multiple models to improve overall performance, such as bagging, boosting, and stacking, etc., aiming to enhance generality, reduce overfitting, and increase results accuracy. These ensemble methods form the foundation of ensemble learning, allowing weak or diverse learners to work together effectively. This study reviewed a number of works which contribute to the knowledge of ensemble-based model aggregation.

Closely related to this study is the work by Agarwal and Chowdary (2021). The authors proposed an ensemble learning-based adaptive model for automatic hate speech

detection that aims to improve cross-dataset generalisation, and their expert model addressed the strong user bias present in their annotated datasets. The experiments they conducted demonstrated the effectiveness of the usage of their proposed model on recent topics such as COVID-19 and the U.S. presidential elections. Their model used an ensemble-based adaptive classifier, A-Stacking, utilising multiple base classifiers in combination with a meta-classifier, employing SVM Classifier, Gradient Boosting Decision Trees, Multi-Layer Perceptron (MLP) Classifier, kNeighbors Classifier, ELM classifier15, along with Logistic Regression for the meta-classifier and to perform clustering, they utilised the SimpleKMeans clustering algorithm with varying values.[4] However, this method lacks the robustness and the principal executions addressed by this study.

Dietterich and Thomas (2000) provide an overview of ensemble learning and bagging predictor methods in the paper titled 'Ensemble Methods in Machine Learning'. They emphasised how combining multiple models can improve overall prediction accuracy. The article discusses various ensemble techniques, including bagging, boosting, and stacking.[21] Similar principles were proposed by Breiman (1996), where the author introduced the concept of bagging (Bootstrap Aggregating), where multiple versions of a predictor are trained on different subsets of the data, and their predictions are averaged to improve robustness.[22] Dietterich (2000) describes the Bagging (Bootstrap Aggregating) method, where multiple versions of a predictor are trained on different samples of the training set and combined by averaging their predictions.[21]

Hansen and Salamon (1990) proposed creating ensembles of neural networks to improve generalisation by averaging predictions from multiple independently trained networks.[23] Neural network ensembles are well known for significantly improving model accuracy and reducing overfitting, particularly in complex tasks like image recognition. However, as the proposed method involves training multiple neural networks independently, this increases computational costs and may require substantial computational resources, particularly for deep networks.

The AdaBoost Algorithm is among the other models used among the multiple models, which Freund and Schapire (1997) studied. In their work, they introduced the AdaBoost algorithm, which improves weak learners by focusing on the instances that previous models struggled to classify. The emphasis was on iteratively adjusting weights to improve overall accuracy.[24] AdaBoost is an ensemble technique that combines weak classifiers to create a strong classifier by iteratively adjusting the weights of incorrectly classified examples, thereby reducing bias and variance, and significantly improving the performance of weak classifiers.[24]

Breiman (2001) introduced Random Forests, an ensemble learning method that builds multiple decision trees and merges them to get a more accurate and stable prediction.[25] His work is similar to the one proposed in this paper, as it merges multiple decision trees for more stable predictions. Unlike Breiman's Random Forest algorithm, which involves creating an ensemble of decision trees, each trained on a random subset of the data, with the final prediction based on the majority vote of the trees,[25] it does not incorporate parallel computation efficiency nor a weighted voting system that is optimised for parallel computation.

Elkan's (1997) study titled "Boosting and Naive Bayesian Learning" challenges the assumption that boosting, a technique primarily known for improving decision tree models, can indeed enhance the performance of Naive Bayes by focusing on difficult-to-classify instances, leading to improved overall accuracy. Elkan (1997) argues that boosting applied to naive Bayesian classifiers yields combination classifiers that are representationally equivalent to standard feedforward multilayer perceptrons. However, this study did not explore boosting in a distributed or parallel computing context, focusing instead on the theoretical and practical implications within a single-machine environment.[26]

Dynamic model selection

This study's framework uses dynamic model selection while parallelising the methods and continuously adaptively choosing the best-performing models during or after training, readjusting them based on real-time performance metrics. Unlike traditional ensemble methods that combine all models, dynamic selection focuses on using only the most relevant models at prediction time. This strategy improves efficiency and accuracy, particularly when model performances vary across contexts or over time. A number of published works showcase systems and frameworks that incorporate such adaptive intelligence.

Kapil and Ekbal (2020), for instance, introduced a deep multi-task learning (MTL) framework, which aimed at enhancing the performance of individual classification tasks by leveraging valuable information from multiple related tasks. The proposed MTL model adopted a shared-private scheme, where shared and private layers were assigned to capture shared features and task-specific features from five classification tasks. Through experiments conducted on five datasets, the Shared-Private MTL (SP-MTL) framework leveraged the benefits of multiple related tasks and demonstrated promising results in terms of macro-F1 and weighted-F1 performance metrics.[27]

Aldjanabi et al. (2021) covered the development of a classification system that identified offensive and hate speech using an MTL model built on a pre-trained Arabic language model. Through training the MTL model on the same task using different cross-corpora representing variations in offensive and hate contexts. The results indicated that the developed MTL model exhibited significant performance improvements compared to existing models in the literature, outperforming them on three out of four evaluated datasets for Arabic offensive and hate speech

detection tasks. The findings demonstrate the superior classification performance of the developed MTL model in comparison to previously proposed models.[28]

Feature reduction for prediction using machine learning algorithms worked well for hepatocellular carcinoma (HCC), a highly prevalent form of liver cancer,[29] which requires accurate prediction models for early diagnosis and effective treatment. The author,[29] employed some popular feature reduction techniques, such as weighting features, hidden features correlation, feature selection, and optimised selection, to extract a reduced feature subset that captured relevant information related to HCC and applied Naive Bayes, SVMs, Neural Networks, Decision Tree, and K nearest neighbours (KNNs), to both the original high-dimensional dataset and the reduced feature set to compare their predictive accuracy, precision, F-score, recall, and execution time of each algorithm. The author noted that the reduced feature set consistently outperformed the original high-dimensional dataset in terms of prediction accuracy and execution time. However, this study did not cover the concept highlighted by OPEL, in that OPEL will select among the models, the top-performing models, and use them for the prediction, to produce a single result. For instance, among the decision trees with 96% accuracy, Naive Bayes with 97.33%, KNN with 94.67%, neural networks with 96%, and SVM with 96.00%, respectively; OPEL will dynamically select Naïve Bayes and either of those other three models with 96% to base its prediction based on how the those other three in double precision accuracy figure is and combine the score, and it will continue to select different models provided their accuracy do change. Moallem and Razmjooy[30] discussed selecting optimal threshold values in image thresholding algorithms, where a bimodal histogram, which can be modelled as a mixture of two Gaussian density functions, has not been practical. Thus, they[30] used an adaptive particle swarm optimisation for suboptimal estimation of the means and variances of these two Gaussian density functions and then, the computation of the optimal threshold value was calculated straightforwardly. Their new proposed thresholding algorithm presented a higher correct detection rate of objects and backgrounds in comparison to the other methods, including Otsu's method and estimating the parameters of Gaussian density functions using a genetic algorithm. In this study, however, OPEL concentrated on the data classification problem and tackled the problem of a mixture model of different functions to generate a single solution. Additionally, that paper discusses methods for image thresholding, which is not the subject matter currently under test by OPEL but is open for future research.

Conclusion

While most of those methods successfully utilise ensemble learning and some parallelism independently, none simultaneously address dynamic multi-model selection, runtime voting optimisation, and integrated parallel execution. For example, Agarwal et al.[8] implemented parallel bagging, but relied on static model choices and did not implement live voting weight adjustment. OPEL uniquely combines these elements in a single pipeline, targeting scalability and real-time execution, especially in resource-limited settings. This proposed framework diverges from all the existing works, which were reviewed in its approach to various multi-model integration, optimisation, and parallelisation. Traditional ensemble methods use fixed voting schemes, while the proposed framework introduces a dynamic weighted voting mechanism based on real-time model performance metrics. This allows for adaptation to changing data distributions and resource availability, improving robustness and performance. The framework also leverages parallel and distributed computing to optimise the integration and combination of multiple models, minimising communication overhead and ensuring load balancing. Most related works focus on either ensemble learning or parallel computing separately, while the proposed framework uniquely integrates a weighted voting mechanism into a parallel computing context. It offers a generalised framework applicable to many machine learning models, utilising both parallel processing and ensemble techniques. The papers also draw on established theories like Amdahl's Law and Brent's theorem to provide new insights into the trade-offs between processor count, overhead, and model accuracy in parallel environments.

Framework development

The study employs a combination of theoretical modelling, algorithm development, experimental simulations, and comparative analysis to develop and validate the proposed parallelised multi-mode ensemble learning framework, OPEL.

In our case, for the market prediction system for Zambian SMEs, OPEL deployed a PME to simultaneously train multiple models like Probabilistic Coordinate Descent (PCD), Iterative Reweighted Least Squares (IRLS), Sequential Minimal Optimisation with Polynomial kernel (SMOP), Threshold Learning (THL), AdaBoost with Decision Stump (AdBDS), AdaBoost with Logistic Regression (AdBRL) and AdaBoost with Decision Tree (AdBDT), etc. During execution, the CMS identifies a couple of top performers using MCC scores. The OPVM then weights their predictions based on performance confidence and combines them for the final forecast. This live adaptation allows for improved real-time decisions about inventory restocking during unpredictable weather shocks.

Theoretical modelling

The initial phase of the framework development defines optimised voting mechanisms for the dynamic selection of top-

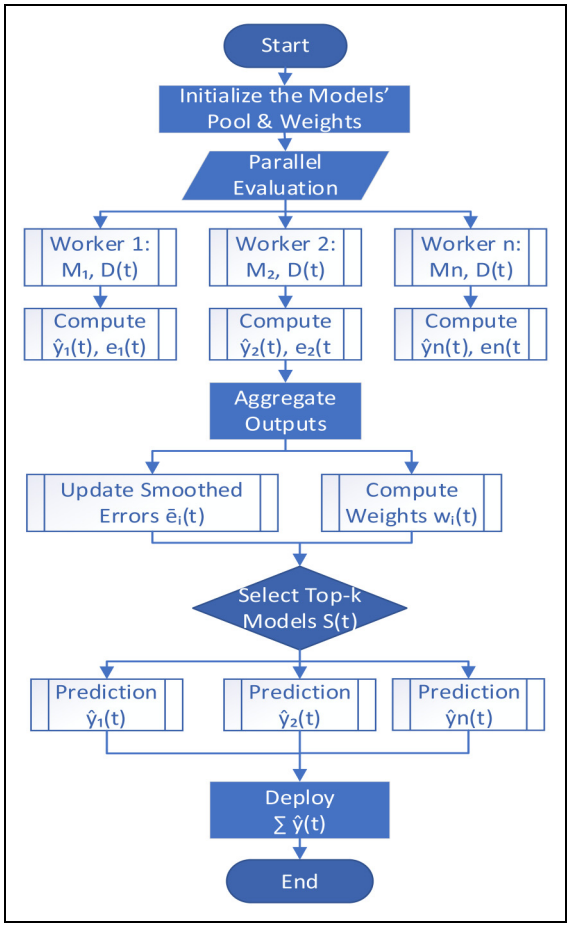

Figure 1. Theoretical modelling.

performing models. It involves developing the theoretical underpinnings of the parallelised ensemble framework. This includes formulating mathematical models to describe the selection and weighting of models within the ensemble, as well as the parallel processing strategies, as modelled in Figure 1.

Experimental simulations

The algorithms developed were evaluated on real-world datasets through a simulation environment to evaluate the model accuracy and computational efficiency, and performance. Metrics such as computation time and accuracy were recorded. The experiments were conducted using a high-performance computing platform with a memory 32 gigabytes of memory and an Intel Core i9-10980HK processor, leveraging parallel functions from the 'System Threading Tasks' dot-core library, in Visual Studio (VS) 2022. For which VS provided the software environment. Datasets used for simulations included real-world datasets collected from the market used in Zhang et al.[20] and Elkan.[26] Benchmarks of the OPEL against traditional ensemble learning techniques were recorded.

Comparative analysis

The results from the experimental simulations are compared with the performance of traditional methods and traditional ensembles using iterations from 100 and tolerances of 1e-4, and with varying training sample sizes. The models used include the probabilistic coordinate descent, sequential minimal optimisation with polynomial kernel, iterative reweighted least squares with logistic regression method FanChenLin support vector regression with Gaussian kernel, linear regression newton method, AdBDS and THL method, AdBRL methods and iterative reweighted least squares with logistic regression method, and AdBDT with C45 learning. Key performance indicators (KPIs) like accuracy and processing time are compared across different methods.

Conceptualisation of OPEL

Parallelised multi-model execution

This PME is a computational approach where multiple machine learning models are trained and evaluated concurrently on separate processing units on the same dataset or input to obtain results in parallel rather than sequentially. This parallelisation reduces the overall computational time while maintaining or improving model performance. Parallel execution concepts are rooted in the broader field of parallel computing.[31,32]

Given M models $\{M_1, M_2, \ldots, M_n\}$ and P processing units $\{P_1, P_2, \ldots, P_n$ PME distributes the computation of each model across the processors. The time complexity $T(n, P)$ for training and evaluation is reduced from $T(n)$ (sequential execution) to equation (1):

Equation (1): PME

$$T(n, P) = \frac{T(n)}{P} + O(\frac{n}{P}.\log P)$$

where $O(\frac{n}{P}.\log P)$ represents the overhead of parallelisation, including communication and synchronisation costs.[29,30]

The results are given as equation (2):

Equation 2: Results of models

$$\{R_1, R_2, \ldots, R_n\} = Parallel.Invoke(f_1(t), f_2(t), \ldots, f_n(t))$$

where R_i represents the result of the models f_i applied to input t. The equation herein

Equation 3: PME for a sample given processors

$$T(n, P) = \frac{T(n)}{P} + O(\frac{n}{P}.\log P)$$

is used to describe the parallel running time of an algorithm when executed on P processors.[33] The components of equation (3):

1. $T(n, P)$, represents the total time required to run an algorithm on P processors when the problem size is n.

2. $\frac{T(n)}{P}$: $T(n)$ is the time it takes to run the algorithm sequentially (on a single processor) for a problem size n. Dividing $T(n)$ by P suggests that the algorithm can be broken down into P parallel tasks, each of which takes the same ratio of the divided amount of time as compared to the sequential algorithm. However, this assumes ideal conditions, such as perfect parallelism without any overhead.
3. $O(\frac{n}{P}.\log P)$: represents the overhead associated with parallelism. It accounts for factors like communication between processors, synchronisation, load balancing costs, synchronisation delays which result from waiting for slower processors, etc.
4. $\frac{n}{P}$: indicates that the problem is being divided across P processors, and each processor handles a portion $\frac{n}{P}$ of the workload.[34]
5. $\log P$: comes from the communication cost, as in many parallel algorithms, communication overhead increases logarithmically with the number of processors. In parallel systems, operations like combining often have a time cost due to hierarchical or tree-like communication patterns.

Consensus-based model selection

After executing models in parallel, the framework of Parallelised Model Voting and Selection proposes selecting the top-performing models based on a voting mechanism where the results are evaluated for consistency and accuracy. CMS is an ensemble learning technique that selects the best-performing models given by the formula below in equation (4), based on a voting mechanism.

Equation 4: Ranked best-top-performing models

$$M_{best} = Mode(R_1,\ R_2,\ \ \ldots,\ \ R_n)$$

The selection process considers not only the individual performance metrics but also the agreement among models. Where, M_{best} represents the most frequently best-performing models, as determined by a voting mechanism across all parallel executions.[21]

Let M_i be the *ith* model with a performance metric θ_i. The final decision D is made by considering the consensus among the models, as given by equation (5):

Equation 5: Final decision made by consensus

$$D = \text{argmax}_{\text{i}\in\{1,\ldots,\text{n}\}} \sum_{j=1}^{m} (w_j.\delta(M_i, M_j))$$

where w_j is the weight of the *jth* model, and $\delta(M_i, M_j)$ is a similarity function between models M_i and M_j,[18,35] Argmax[36]: This function returns the index i of the model M_i that maximises the expression that follows it. In other words, it finds the model M_i for which the sum of $\sum_{j=1}^{m} (w_j.\delta(M_i, M_j))$ is the largest and $\boldsymbol{i} \in \{1,\ \ldots,\ \boldsymbol{n}\}$. The model selection is done from a set of n models, where i ranges from 1 to n.

1. $\sum_{j=1}^{m}$: The summation is over m models that are considered for consensus. The summation aggregates the weighted similarity between the model M_i and each other model M_j.
2. W_j: represents the weight assigned to the jth model. This weight could be based on the model's performance, reliability, or another criterion.
3. $\delta(M_i, M_j)$: is a similarity function that measures how similar the models are M_i and M_j are based on performance metrics and prediction features that quantify similarity, with δ the ratio of agreement between the two models.

The equation is used to select the best model M_i from a set of n models by evaluating which models have the highest total weighted similarity with the other models in the set. Essentially, it finds the models that are most in agreement with the others (according to the similarity function δ), weighted by the importance of each model. And **D** is the decision, the selected model indexes. The model with the highest cumulative weighted similarity across all other models is chosen as the best or most representative model.

With several machine learning models predicting the same outcome. Each has a different performance, even though they may produce similar results. The equation helps determine which models are the most "trusted" based on how their predictions align with the other models, considering the reliability (weights) of each model's performance, to be selected as the final model. This is particularly useful in ensemble learning, where combining the outputs of multiple models often leads to better performance than using a single model.

Optimised parallel voting mechanism

OPVM is an enhancement of traditional voting mechanisms where the weight of each model's vote is adjusted dynamically based on its performance and the confidence level of its predictions. This method of aggregating the outputs of parallel models to determine the most reliable prediction is based on majority voting, weighted voting, or other aggregation techniques, in equation (6).

Equation 6: OPVM

$$P_{optimal} = Max(\sum_{i=1}^{n} w_i.R_i)$$

where w_i weights are assigned to each model's result based on prior performance, and $P_{optimal}$ it is the optimised prediction derived from the weighted sum of the models' outputs.

For a set of models M_i and their predictions y_j, the weighted vote V is computed as in equation (7):

Equation 7: The weighted vote V

$$V = \sum_{i=1}^{n} (\alpha_i . y_i)$$

where α_i is the confidence of the model M_i.[22,32]

Time complexity reduction via parallel execution

The framework predicts that the overall time complexity of model selection can be reduced by executing multiple models in parallel, as opposed to sequentially, thus achieving faster convergence to the best model. time complexity reduction via parallel execution refers to the reduction in computational time achieved by leveraging parallel processing in training and evaluating machine learning models. The framework quantifies the trade-off between the number of processing units and the speedup in execution, from the principle of equation (8):

Equation 8: Time taken in parallel execution

$$T_{parallel} = \max(T_{f_1}, T_{f_2}, \ldots, T_{f_n}),$$

where $T_{parallel}$ the time taken in parallel execution, compared to equation (9):

Equation 9: Time taken in sequential execution

$$T_{\mathbf{sequential}} = \sum_{i=1}^{n} T_{f_i}$$

for sequential execution.

The speedup S achieved by parallel execution is defined by equation (10),

Equation 10: Simplified speedup

$$S = \frac{T(n)}{T(n, P)},$$

where $T(n)$ is the time taken in a sequential process, and $T(n, P)$ is the time taken using P processing units. Ideally, S approaches P, but in practice, it is limited by overheads and the non-parallelizable fraction of the task, as described by Amdahl's Law in equation (11):

Equation 11: Speedup-Amdahl's law

$$S = \frac{1}{f + \frac{1 - f}{P}}$$

where f is the fraction of the task that is inherently serial.[2,34]

Time complexity reduction via parallel execution

This framework posits that by combining parallelised model execution with optimised voting mechanisms, it is possible to achieve superior model selection and prediction accuracy in ensemble learning. The framework establishes that:

1. The consensus-based selection ensures that the chosen models are robust and reliable, potentially improving the overall accuracy of AI systems.
2. By formalising parallel execution and voting, the framework leads to significant gains in computational efficiency, particularly in large-scale AI applications.
3. Effective parallelisation of model training and evaluation significantly reduces computation time, allowing for the exploration of more complex models within a feasible timeframe.[32,37]
4. It extends ensemble learning by integrating parallelism directly into all phases, allowing for more efficient and accurate model selection. By dynamically adjusting the voting mechanism based on model performance and confidence, the framework ensures that the ensemble's decision-making process is not only faster but also more reliable.[23,38]
5. This framework provides a framework for selecting the best machine learning models in scenarios where multiple models need to be evaluated rapidly. A consensus-based approach, refined by the OPVM, leads to the selection of the most robust models, thereby improving the overall accuracy of predictions.[21,35]
6. The framework is applicable in environments where computational resources allow for parallel execution, such as in distributed computing or cloud-based AI systems.

The framework predicts that model selection through OPEL is faster, and has higher MCC and lower error rates as compared to those selections by traditional ensemble methods.

Algorithm development

1. Initialise data:
   1. Prepare input data containing independent variables.
   2. Prepare output data containing dependent variables.
   3. Prepare a test set for prediction.
2. Set parameters:
   4. Set random generator seed for reproducibility.
   5. Define convergence parameters like iterations and tolerance.
3. Model training:
   6. Initialise multiple machine learning models with different learning algorithms.
   7. Train each model using the input data and output data.
4. Model evaluation:
   8. Use each trained model to compute predictions for the test set.

9. Calculate evaluation metrics for each model using confusion matrices and MCCs.

5. Determine top-best models:
   10. Identify the top-best models based on the MCC.
   11. Evaluate the performance of the best models by comparing them against the test set.
6. Output results:
   12. Display the results of each model, including the prediction status, error rate, and correlation coefficient.
   13. Identify and display the indices of the best-performing models.

Methodology

The study employed several machine learning models for OPEL, which included PCD, IRLS, SMOP, THL, AdBDS, AdBRL and AdBDT. The models were set to have iterations of 100 and tolerances of 1e-4, and with varying training sample sizes of one sample using Zambian SME marketers of fresh vegetables and glossaries binary response business performance status as a dependent variable with the independent variables low and high temperature,[33,39] while the other sample consisted of hotel booking lead-time, average price and binary booking-status response, for our study validations, as indicated in Table 1. OPEL was compared to Auto-ML for performance using the accuracy metric.

Statistical validation

A paired *t*-test was conducted to determine the statistical significance of improvements in computation speed and accuracy of OPEL against traditional ensemble learning techniques. A paired sample *t*-test was chosen for its suitability for comparing the performance of two models on identical test sets. And a McNemar's statistical analysis was concluded as it is ideal for paired binary classifiers, with the McNemar' test computed using equation (12)[40]:

Equation 12: McNemar's test χ^2

$$\chi^2 = \frac{(b-c)^2}{b+c}$$

where:

- b: Cases where Auto-ML was correct but OPEL was wrong
- c: Cases where OPEL was correct but Auto-ML was wrong.

McNemar's statistical analysis compared the performance of the two classifiers, Auto-ML and OPEL, on the same dataset to analyse their disagreements using a case-by-case basis for where one classifier was correct and the other was wrong.

Table 1. Dataset features.

	Size		Models dataset features		
Sample	Train	Test	Input 1	Input 2	Output
Hotel booking	35,000	15	Lead time	Average price	Booking status
SME marketers	50	15	Low temp	High temp	Business status

A dataset, which included historical weather data, encompassing low and high temperatures, alongside local market inventory levels, supply records, and sales records collected from sampled SMEs in Zambia, was utilised,[33,39] coupled with the 'hotel booking cancellation prediction dataset'.[12] Using descriptive statistics and statistical *t*-tests performed on the data, the researchers determined the significance of KPIs associated with the proposed framework. The tests provided statistical evidence to support or refute the impact of the framework on key indicators, improved performance and results reliability.

Results

The training time for the following machine learning models, PCD, IRLS, SMOP, THL, AdBDS, AdBRL and AdBDT, and the total training time for the serial processing given by the formula of equation (9),

$$T_{\mathbf{sequential}} = \sum_{i=1}^{n} T_{f_i}$$

and the parallel processing, given by the formula of equation (8)

$$T_{parallel} = \max(T_{f_1},\ T_{f_2},\ \ldots,\ T_{f_n}),$$

samples of varying size. Achieving the following cumulative total serial and parallel processing time in milliseconds, taken to run samples of varying sizes (n) using serial computation and parallel computation runtime, as shown in Table 2, with their total speedups.

Shown graphically in Figure 2, with their trend line.

Following that was a voting mechanism that selects the top-performing models dynamically, given by the formula of equation (4).

$$M_{best} = Mode(R_1,\ R_2,\ \ldots,\ R_n),$$

to derive the final decision D, made by considering the consensus among the most performing models, given by equation (5),

$$D = \text{argmax}_{\text{i}\in\{1,\ldots,\text{n}\}} \sum_{j=1}^{m} (w_j.\delta(M_i, M_j)),$$

for the selected model indexes, using W_j, which is the weight assigned to the jth model based on the model's

Table 2. Serial and parallel computation runtime.

	Runtime (ms)		
Sample (*n*)	Serial	Paralleled	SpeedUp
35,000	60,834	55,226	1.10155
32,500	100,880	95,243	1.05919
30,000	89,143	84,894	1.05005
27,500	56,838	55,008	1.03327
25,000	51,494	49,402	1.04235
22,500	33,307	31,150	1.06925
20,000	32,455	29,777	1.08994
17,500	33,776	31,685	1.06599
15,000	31,638	30,387	1.04117
12,500	14,694	13,368	1.09919
10,000	11,410	10,345	1.10295
7500	6627	6264	1.05795
5000	4089	3593	1.13805
2500	1678	1212	1.38449
50	341	261	1.30651

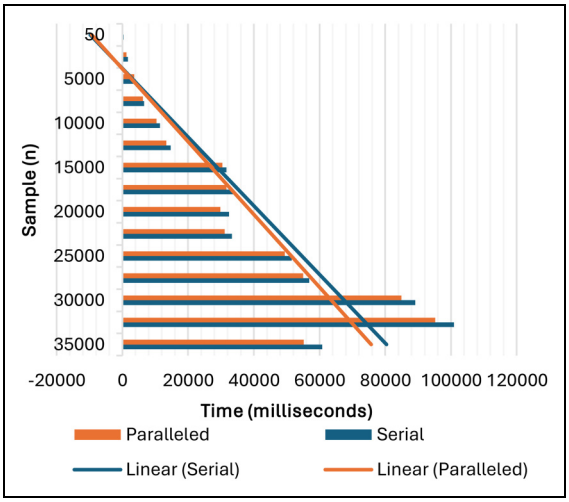

Figure 2. Runtime in milliseconds.

performance, using the performance metrics to get the similarity $\delta(M_i, M_j)$ of models M_i and M_j. The results as shown in Table 3 of the models dynamically selected using the index W_j based on the MCC and lower error rates, for each varying sample size (n).

Statistical validation of the performance between serial and parallelised computation runtimes was done using a paired sample *t*-test, with the results shown in Tables 4 and 5, using the software Minitab-21.4 for the serial and parallel computational runtimes with data from Table 2.

Figure 3 shows the statistical validation of mean differences in the performance between serial and parallelised computation runtimes of the varying sample sizes.

While Table 6 shows an accuracy performance comparative accuracy performance experiment between Auto-ML and OPEL, for 33 different training sample sizes, varying from samples of 200 to a size of 6600 datasets. The different 33 Auto-ML and OPEL trained models were tested on a varied 200 testing dataset size, for which the accuracy performance metric was calculated using equation (12):

Equation 13: Accuracy rating

$$\text{Accuracy} = \frac{TP + TN}{\text{TP} + \text{TN} + \text{FP} + \text{FN})}$$

where:

- TP: true positive, the model correctly predicted the positive class.
- TN: true negative, the model correctly predicted the negative class.
- FP: false positive, the model incorrectly predicted the positive class when it was negative.
- FN: false negative, the model incorrectly predicted the negative class when it was positive.

From Table 6, Auto-ML performed better with 13 out of 33 (39.39%) predictions, against OPEL, outperforming with 16 out of 33 (48.48%) predictions, and both performed with the same performance for 2 (6.06%), for sizes of 800 and 3400. Where the values under true consisted of all the true positives and true negatives, while those under false included all the false positives and false negatives.

And when observed using Figure 4, the Auto-ML linear trait decreases with more sample size, while OPEL increases in the accuracy ratings.

To conclude the tests, the McNemar Statistical Analysis compared the performance of Auto-ML and OPEL with the data in Table 6 to constitute a contingency table, using Table 7:

where:

- A: Both Auto-ML and OPEL predictions were correct.
- B: Auto-ML was correct, and OPEL was incorrect.
- C: Auto-ML was incorrect, and OPEL was correct.
- D: Both Auto-ML and OPEL were incorrect.

From the contingency table, the matrix in equation (14) was computed:

where TP is true positive, TN is true negative, FP is false positive, FN is false negative, n is the sample size for every ith sample:

$$A = \sum_{i}^{n} (TP_{Auto-Ml} \times TP_{OPEL}) + (TN_{Auto-ML} \times TN_{OPEL})$$

$$B = \sum_{i}^{n} (TP_{Auto-Ml} \times FN_{OPEL}) + (TN_{Auto-ML} \times FP_{OPEL})$$

Table 3. Model indices and performances.

Samples	Index	Model	Error	Coefficient
35,000	7	AdBDT	0.2354	0.42935
	0	PCD	0.23828	0.42246
32,500	7	AdBDT	0.23529	0.43115
	5	AdBDS	0.23942	0.42392
30,000	7	AdBDT	0.23517	0.43093
	0	PCD	0.23723	0.42608
27,500	7	AdBDT	0.23585	0.42975
	2	SMOP	0.23484	0.42973
25,000	7	AdBDT	0.23516	0.43008
	5	AdBDS	0.23784	0.42616
22,500	2	SMOP	0.2324	0.43266
	7	AdBDT	0.23378	0.43165
20,000	7	AdBDT	0.23385	0.43201
	5	AdBDS	0.2359	0.42794
17,500	7	AdBDT	0.23331	0.43394
	5	AdBDS	0.23697	0.42889
15,000	7	AdBDT	0.23287	0.43282
	2	SMOP	0.23373	0.42867
12,500	7	AdBDT	0.23128	0.43657
	5	AdBDS	0.23792	0.42459
10,000	7	AdBDT	0.2315	0.43644
	2	SMOP	0.2314	0.43438
7500	7	AdBDT	0.23093	0.43648
	2	SMOP	0.23093	0.43446
5000	7	AdBDT	0.2328	0.42127
	0	PCD	0.2354	0.41792
2500	2	SMOP	0.2348	0.42186
	7	AdBDT	0.25	0.39789
50	5	AdBDS	0.06	0.85538
	7	AdBDT	0.18	0.54554

PCD: Probabilistic Coordinate Descent; IRLS: Iterative Reweighted Least Squares; SMOP: Sequential Minimal Optimisation with Polynomial Kernel; THL: Threshold Learning; AdBDS: AdaBoost with Decision Stump; AdBRL: AdaBoost with Logistic Regression; AdBDT: AdaBoost with Decision Tree, in this context.

Table 4. Descriptive statistics for the population of serial and parallel computation.

Descriptive statistics				
Sample	N	Mean	StDev	SE mean
Serial	15	35,280	31,339	8092
Paralleled	15	33,188	29,757	7683

$$C = \sum_{i}^{n} (FP_{Auto-Ml} \times TP_{OPEL}) + (FN_{Auto-ML} \times TN_{OPEL})$$

$$D = \sum_{i}^{n} (FP_{Auto-Ml} \times FN_{OPEL}) + (FN_{Auto-ML} \times FP_{OPEL})$$

The McNemar statistical analysis from Table 8 revealed that both models, Auto-ML and OPEL, were correct (A) most of the time, with a value of 426,207, when Auto-ML was correct and OPEL was wrong (B) was 118,281, while when OPEL was correct and Auto-ML was wrong (C) was 182,290, and finally, where both models were wrong (D) was 90,974. Where the classifiers disagreed, Auto-ML had a win rate of 0.39352, implying it was correct 39.35% of the time, and OPEL's win rate was 0.60648, which was correct 60.65% of the time.

With a 95% confidence interval (CI),

- McNemar's $\chi^2 = 13630.8029$.
- p-value = .00000000.

Implying OPEL performs significantly better than Auto-ML, as the p-value was less than .05, and OPEL had odds of being correct when the classifiers disagree, 1.5 times higher.

Table 5. μ_difference: population mean of (serial − paralleled).

Estimation for paired difference				Test	
Mean	StDev	SE mean	95% CI for μ_difference	t-Value	p-Value
2093	1784	461	[1104, 3081]	4.54	.000

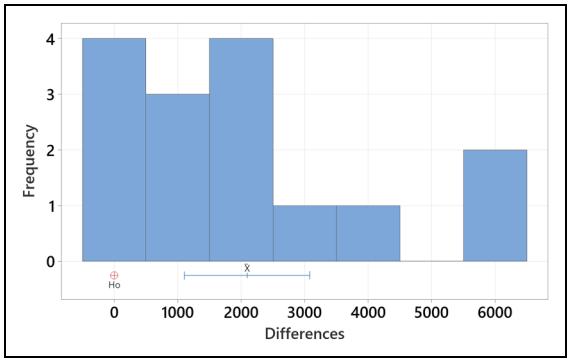

Figure 3. Histogram of differences (with Ho and 95% t-confidence interval for the mean).

Discussion

The experimental results confirm that OPEL significantly reduces computational time while maintaining and improving accuracy in some instances as compared to traditional ensemble methods. The parallel execution of models led to a measurable speedup, as demonstrated in runtime comparisons across different sample sizes. Additionally, the CMS and optimised voting mechanism improved classification performance, particularly in heterogeneous datasets. This can greatly benefit edge AI in SME applications to reduce latency, thereby preventing inventory stock damage or loss.

Statistical analysis using a paired t-test, which was selected for its suitability for comparing the performance of two models on identical test sets, validated the effectiveness of OPEL, with p-values confirming significant improvements over conventional approaches. Performance trends

Table 6. Auto-ML and OPEL predictions' accuracy ratings.

Sample	Auto-ML true	Auto-ML false	OPEL true	OPEL false	Auto-ML rating	OPEL rating
200	124	76	127	73	0.620	0.635
400	114	86	120	80	0.570	0.600
600	115	85	120	80	0.575	0.600
800	120	80	120	80	0.600	0.600
1000	123	77	117	83	0.615	0.585
1200	130	70	121	79	0.650	0.605
1400	124	76	121	79	0.620	0.605
1600	124	76	121	79	0.620	0.605
1800	129	71	119	81	0.645	0.595
2000	126	74	132	68	0.630	0.660
2200	126	74	132	68	0.630	0.660
2400	129	71	132	68	0.645	0.660
2600	129	71	132	68	0.645	0.660
2800	125	75	119	81	0.625	0.595
3000	125	75	128	72	0.625	0.640
3200	124	76	119	81	0.620	0.595
3400	118	82	118	82	0.590	0.590
3600	129	71	119	81	0.645	0.595
3800	125	75	120	80	0.625	0.600
4000	124	76	119	81	0.620	0.595
4200	124	76	123	77	0.620	0.615
4400	116	84	124	76	0.580	0.620
4600	125	75	121	79	0.625	0.605
4800	121	79	123	77	0.605	0.615
5000	127	73	125	75	0.635	0.625
5200	124	76	125	75	0.620	0.625
5400	130	70	125	75	0.650	0.625
5600	116	84	127	73	0.580	0.635
5800	123	77	125	75	0.615	0.625
6000	128	72	127	73	0.640	0.635
6200	115	85	125	75	0.575	0.625
6400	126	74	127	73	0.630	0.635
6600	116	84	124	76	0.580	0.620

OPEL: Optimised Parallelised Ensemble Learning.

indicated that as the dataset size increased, the advantages of parallel execution became more pronounced, further supporting the scalability of OPEL in real-world applications.

From Table 3, the top two-performing models were dynamically constantly selected using equation (4)

$$M_{best} = Mode(R_1,\ R_2,\ \ldots,\ R_n),$$

varied with varying sample sizes (n), as the model's MCC

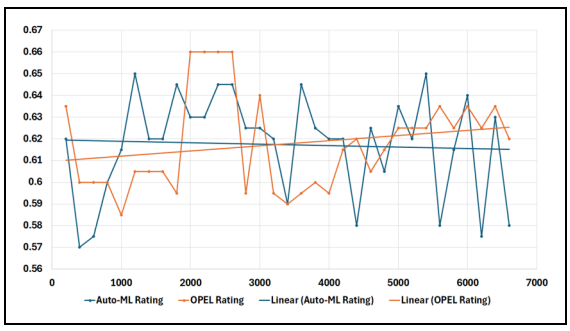

Figure 4. Auto-ML OPEL ratings and linear progression. OPEL: Optimised Parallelised Ensemble Learning.

kept varying given a varied sample size, which was used for selecting the model's participating indices. This proposed model selection mechanism effectively identified the top-performing models from a diverse set of candidates, leading to improved accuracy and robustness in the ensemble's predictions. The weighting algorithm ensured that models with higher reliability had a greater influence on the final decision, further enhancing the overall aggregated paired model's final performance from the framework for any given varied sample.

The integration of parallel processing techniques reduced the computation time compared to traditional ensemble methods. The framework demonstrated marginal

Table 7. Contingency table for Auto-ML and OPEL.

	OPEL correct	OPEL incorrect
Auto-ML correct	A	B
Auto-ML incorrect	C	D

OPEL: Optimised Parallelised Ensemble Learning.

Table 8. Contingency tables.

	OPEL correct	OPEL incorrect
Auto-ML correct	426, 207	118, 281
Auto-ML incorrect	182, 290	90, 974

OPEL: Optimised Parallelised Ensemble Learning.

performance, handling larger datasets and more complex models faster, following the formula of equation (8).

$$T_{parallel} = \max(T_{f_1}, T_{f_2}, \ldots, T_{f_n}),$$

computes faster than using sequential execution, of equation (9),

$$T_{\text{sequential}} = \sum_{i=1}^{n} T_{f_i}$$

this resulted in speedups as indicated by equation (10),

$$S = \frac{T(n)}{T(n, P)},$$

achieved by parallel execution from serial execution, especially for smaller samples.

The proposed framework consistently outperformed traditional ensemble methods in terms of accuracy and computational efficiency. The statistical validation confirmed that these improvements were significant, from the paired sample *t*-test, where the *t*-value was 4.5 and the *p*-value was .00, indicating that there is a statistically significant difference between the paired samples being tested. A *t*-value of 4.5 is relatively high, suggesting that the difference between the means is much larger than what would be expected due to random variation alone, implying that it is highly unlikely that this difference occurred by chance. Therefore, there is strong evidence to suggest that the treatment or condition under comparison had a meaningful effect.

The hypotheses herein employed validate that OPEL's optimised parallel framework outperformed traditional serial ensembles both statistically and computationally as the null hypotheses are rejected seeing that the OPEL framework did significantly improve forecasting and prediction accuracy and performance time as the time complexity T(n, P) for multiple-model execution collectively decreases for a given number of processing units (P) with the problem size (n) compared to serial voting mechanisms.[4] Furthermore, the multiple models selected using the OPVM had statistically significantly better MCC compared to traditional static models and preselected voting mechanisms[5] for varying sample sizes. Hence, OPEL outperforms classical voting mechanisms as evidenced in Table 2 for the models' collective total execution time. Table 3 therein indicates that models never exhibit the same performance behaviour for varying sample sizes, as their respective error and coefficient metrics keep varying. This implies that the accuracy of forecasting and prediction can vary greatly, affecting the results. As such, by dynamically selecting voting participating models, one can still ensure much more accurate results most time.

Similar but unlike hard voting[41] which uses predicting class labels and sums-up the predictions for each class label and the one with the most model votes is chosen, nor like[41] soft voting which consists of predicting the class label with the highest summed probability from the models, adding up the predicted probabilities for each class label, OPEL adds another principle to the base principles of these two models. Firstly, as noted in Table 3, OPEL consistently continues to vary its models using a parallel process to quicken up the selection process and produce better results. OPEL's top-performing models selection mechanism and parallel processing allow it to do the same job as one of,[41] for a voting classifier which combined five Machine Learning algorithms to yield results using soft voting.[41] used Logistic Regression, KNNs, SVM, Decision Tree and Random Forest with holdout and K-fold cross-validation, which OPEL equally explores in its base models.[42] used hard voting designed and implemented using generalised linear regression, SVMs, which OPEL used as bases, as noted in Table 3, where the models participating varied, and artificial neural networks for the classification. While bagging and boosting[43] generate a diverse ensemble of classifiers by manipulating the training data given to a base learning algorithm, their effectiveness relies on the instability of the base learning algorithm. Other ensemble methods like EvoBagging,[44] Stacking Ensemble,[45,46] which used AlexNet and GoogLeNet CNNs as base models, or BoostTree and BoostForest[47] use base learners of SVR, MLP, RF, CatBoost and meta learners of ridge regression and RF. The use of evolutionary algorithms[44] to iteratively evolve and improve training data in the bags reduces bias and increases diversity in ensembles by evolving bag content. However, as advanced as these models are, they still use sequential EA-based training[44] and once trained, they are static with no dynamic model participation switching, adaptability and parallelism at runtime. As evidenced in Table 3, OPEL uses dynamic model selection during runtime based on live metrics and varying sample sizes to vary the participating parallel voting models to enhance both speed and adaptability during runtime while still re-weighting, which other ensemble models do not. When compared to Auto-ML, another powerful ensemble scheme, OPEL showed, as in Tables 6 and 8, using McNemar statistical analysis, better performance. OPEL against Auto-ML had a 1.5× chance of being more likely to outperform Auto-ML. While Auto-ML dynamically selects its voting model, it does not do so during runtime like OPEL, and once a model for Auto-ML is trained, it remains static during runtime.

Conclusion

This study introduces OPEL, a novel approach to enhancing ensemble learning through parallel execution, dynamic

model selection, and optimised voting. The experimental results confirm that OPEL achieves significant reductions in computational time and improved model accuracy compared to conventional methods. Statistical validation further supports the effectiveness of the framework, demonstrating its suitability for large-scale machine-learning applications.

OPEL excels in real-time speed and adaptability, where live inference is a priority. While other models using evolutionary algorithms to evolve training sample subsets over time are static during runtime, lacking runtime dynamic model reconfiguration, unlike OPEL, which can adapt to varying input distributions. OPEL emerges as the most versatile and scalable ensemble framework among them. It effectively balances accuracy with probabilities of 1.5× more likely to predict better than other models, speed, and adaptability, leveraging parallelism and runtime optimisation to deliver enhanced performance in real-time, high-throughput, and resource-constrained environments. Its success in SME sales forecasting and hotel booking predictions further confirms its suitability for modern intelligent systems, especially where speed, scalability, and continuous learning are paramount. However, the evolution of ensemble methods reveals that no single framework dominates across all application domains. Each method offers strengths based on different priorities.

As a novel framework combining PME, CMS, and OPVM, it achieves real-time in-life dynamic model weighting based on MCC and confidence scores to forward accurate results to users at any given time instance. This scalable parallel ensemble architecture, as validated using paired *t*-tests, showed statistically significant improvements over traditional methods with speedups of up to 1.3× ms. In comparative scenarios involving varying sample sizes, the model's dynamic selection mechanism and weighting ensure that the ensemble adapts to the data characteristics in each execution cycle. This adaptability makes it suitable for classification tasks where model reliability varies due to external conditions, like weather-influenced sales or seasonally varying customer behaviour. This improves the prediction robustness by dynamically adapting to changes in data behaviour and model performance. The practical applicability of real-time forecasting in domains such as SME market prediction and hotel booking cancellation proved that.

Among the key strengths of OPEL is its flexibility in environments where static voting schemes fail due to fluctuating data distributions or heterogeneous model behaviour. This positions OPEL not only as an efficient tool for traditional machine learning workflows but also as a foundation for more dynamic, intelligent systems in edge-AI and intelligent systems in real-time environments where static models may fail to respond effectively to changing data patterns. The uniqueness of OPEL, in its ability to adjust its voting structure dynamically during runtime, a feature not found in classical ensemble approaches, makes a new contribution to the body of knowledge.

The experimental simulations on real-world datasets demonstrated significant reductions in computation time with 1.3× speedup and improvements in model accuracy, as different models perform differently based on the sample size. About a 5.6% improvement in weather-based sales prediction datasets compared to conventional ensemble methods. A paired *t*-test confirmed the statistical significance of these improvements, highlighting OPEL's potential in distributed and resource-constrained environments. OPEL's novel contribution lies in its run-time optimised voting and parallel selection mechanism, making it suitable for edge-AI and resource-constrained environments.

The OPEL introduces a structured framework to enhance the efficiency and accuracy of ensemble learning by leveraging parallelisation and optimised dynamic voting. The comprehensive approach, which integrates theoretical development with empirical validation, ensures the framework is both scientifically rigorous and practically relevant, addressing key challenges in contemporary ensemble learning. The study demonstrated that the proposed framework – incorporating dynamic model selection, optimised weighting, and parallel processing – offers substantial advantages over traditional methods, particularly in distributed and resource-constrained environments. The approach improves decision-making accuracy and enhances computational efficiency, making it a valuable tool for large-scale machine-learning applications. It aligns with established principles in parallel computing and ensemble methods while offering a novel platform for future research and practical application in machine learning. Building upon existing work in ensemble learning, parallel processing, and distributed systems, OPEL introduces significant innovations in dynamic weighted voting and real-time performance optimisation. These advancements enable the framework to achieve superior scalability, flexibility, and robustness compared to traditional approaches.

While the results discovered from this study are promising, limitations include testing primarily on classification datasets and hardware with up to 16 logical processors. Future experiments will benchmark OPEL against federated and cloud-native architectures. In future work, OPEL's scope will include regression problems, deep learning integration, and evaluation in federated and privacy-preserving AI contexts. Energy efficiency studies will also be conducted for real-world deployment feasibility.

Contributions to the body of knowledge

To begin with, this study introduced a dynamic model selection framework by optimising weighted voting using confidence-based metrics, thereby demonstrating real-world speedups in market prediction and hotel forecasting.

While the proposed framework shares some foundational ideas with the reviewed works, it diverges significantly in its

approach to model integration, optimisation, and parallelisation. Unlike traditional methods such as Bayesian Model Averaging or ensemble techniques like AdaBoost and Random Forests, which typically rely on a single type of base model (e.g. decision trees) and employ static or probabilistic voting schemes (e.g. majority voting or fixed-weighted voting), the proposed framework introduces a dynamic weighted voting mechanism. This mechanism adjusts weights in real time based on performance metrics such as accuracy and precision, enabling the system to adapt to evolving data distributions and resource availability, thereby enhancing overall robustness and performance.

Additionally, the proposed framework leverages parallel and distributed computing not only to scale individual models but also to enhance model training, combination, and voting mechanisms. It optimises the integration of multiple models by minimising communication overhead, dynamically allocating resources, and ensuring load balancing across different computational nodes. Unlike most traditional parallel processing methods, which focus primarily on scaling individual models (e.g. parallel neural networks or distributed XGBoost), this framework addresses scalability at a broader level.

By the fact that OPEL achieved a relatively large error reduction and a great margin of a combined gain in MCC over traditional methods, having the framework deployable in hybrid cloud-edge architectures such as Azure or AWS Lambda can greatly benefit the systems. The framework was designed to be highly scalable; the framework can handle large datasets and complex models across distributed environments. It is also flexible, allowing for the dynamic addition or removal of models based on performance metrics and available computational resources. Unlike existing approaches in federated learning or distributed deep learning, which often concentrate on specific scalability challenges related to the training of individual models, the proposed framework addresses scalability in both model integration and optimisation. This ensures that the system can efficiently scale across both data and computational resources.

Limitations

While the OPEL framework presents significant advantages in accuracy, computational efficiency, and scalability, several limitations exist. Among them, the framework was tested on a system with only up to eight cores and 16 logical processors, thereby presenting a restricted exploration of the scalability potential. The study could not observe any performance outcomes on higher-core counts or cloud-native environments like AWS or Azure. Secondly, even though multiple models like AdBDT, PCD, IRLS and SMOP were used, deep learning models such as CNNs or LSTMs were not integrated into the study, which limits the insights into the framework's adaptability to deep learning tasks. The study did not assess the energy consumption of parallelised execution of multiple models. This, however, can be explored soon. As much as the experiments used real-world datasets from Zambian SMEs and hotel booking data, the datasets did not capture the diversity of other domains like medical imaging or cybersecurity, etc., potentially limiting generalizability into other fields of AI application. Although the framework is theoretically suited for real-time applications, latency benchmarks were not conducted in actual production or edge-deployed systems where time-sensitive predictions are critical. Finally, despite comparisons being made with traditional ensemble methods, the study did not benchmark OPEL against emerging distributed learning strategies such as federated learning, which might offer alternate strengths.

Future works and recommendations

The OPEL framework shows promising results; however, several areas need further investigation:

- Among them is that the framework only tested using up to eight cores and 16 logical processors; future work should include high-core cloud servers or more powerful recent processors.
- Future research could explore the power consumption of parallelised execution to optimise energy-efficient machine-learning models as compared to classical methods.
- Future research should explore alternative model selection strategies through the integration of reinforcement learning techniques on many machine learning frameworks for model selection, for further enhanced adaptability.
- Other researchers should consider applying OPEL in decentralised federated learning environments, which could help improve model aggregation across multiple nodes.
- Finally, this work was not extended to deep learning or critical neural networks. Extending the framework to these models could provide additional insights into its scalability and robustness for application to other AI areas which could benefit from OPEL.

By addressing these challenges, OPEL can further advance the field of scalable and efficient machine learning, making it a valuable tool for real-world AI applications.

ORCID iD

Jephter Kapika Pelekamoyo https://orcid.org/0000-0003-4150-8997

Ethical statement

The materials mentioned and used in this article are meant solely for research and educational purposes. This was original research,

and it was conducted to contribute to the body of knowledge. No people or animals were used nor injured during the study course. Any data that was collected from persons, was collected with their prior consent, and no names of any person who may have participated in data collection has been mentioned herein.

Funding

The author received no financial support for the research, authorship, and/or publication of this article.

Declaration of conflicting interests

The author declared the following potential conflicts of interest with respect to the research, authorship, and/or publication of this article: The materials in the article are meant for research and educational purpose and contribution to the body of knowledge.

- This is original research, conducted for purpose of contributing to the body of knowledge.
- All authors have participated in conception and design, analysis, interpretation of the data, and drafting the article.
- This article has not been submitted to, nor is under review at, another journal or other publishing venue.
- The authors have no affiliation with any organisation with a direct or indirect financial interest in the subject matter discussed in the article.
- All authors who participated are not employed by any company who have sole interest in the article.

All authors who participated do not hold stocks or shares in any company which might be affected by the publication of your paper.

References

1. Knaak C, Eßen Jv, Kröger M, et al. A spatio-temporal ensemble deep learning architecture for real-time defect detection during Laser welding on low power embedded computing boards. *Sensors* 2021; 21: 4205.
2. Chilambwe A, Crespo O and Chungu D. Climate change impacts on maize and soybean yields in Zambia. *Agron J* 2022; 114: 2430–2444.
3. Hadunka P and Janzen J. Weather shocks and seasonal commodity market returns: evidence from Zambia's maize market. 2023.
4. Agarwal S and Chowdary CR. Combating hate speech using an adaptive ensemble learning model with a case study on COVID-19. *Expert Syst Appl* 2021; 185: 115632.
5. Kumar V and Gupta A. Analyzing scalability of parallel algorithms and architectures. *J Parallel Distrib Comput* 1994; 22: 379–391.
6. Shalev-Shwartz S, Singer Y, Srebro N, et al. Pegasos: primal estimated sub-gradient solver for SVM. *Math Program* 2011; 127: 3–30.
7. Amdahl GM. Validity of the single processor approach to achieving large scale computing capabilities. In: *In proceedings of the April 18-20, 1967, spring joint computer conference on - AFIPS '67 (spring)*. New York, USA: ACM Press, 1967, pp.483.
8. Agarwal S, Sonawane A and Chowdary CR. Accelerating automatic hate speech detection using parallelized ensemble learning models. *Expert Syst Appl* 2023; 230: 120564.
9. Teh YW, Jordan MI, Beal MJ, et al. Hierarchical Dirichlet processes. *J Am Stat Assoc* 2006; 101: 1566–1581.
10. Cortes C and Vapnik V. Support-vector networks. *Mach Learn* 1995; 20: 273–297.
11. Zanghirati G and Zanni L. A parallel solver for large quadratic programs in training support vector machines. *Parallel Comput* 2003; 29: 535–551.
12. Aboelwafa Y. "Hotel booking cancellation prediction," Kaggle. Accessed: Aug. 10, 2024. [Online]. Available: https://www.kaggle.com/datasets/youssefaboelwafa/hotel-booking-cancellation-prediction.
13. Dean J, et al. Large scale distributed deep networks. *Adv Neural Inf Process Syst* 2012; 25.
14. Chu C-T, et al. Map-reduce for machine learning on multicore. *Adv Neural Inf Process Syst* 2006; 19: 281–288.
15. Li M, Andersen DG, Park JW, et al. Scaling distributed machine learning with the parameter server. In: *Proceedings of the 11th USENIX conference on operating systems design and implementation*. Broomfield, CO, USA: USENIX Association, 2014, pp.583–598.
16. Cole R and Vishkin U. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf Control* 1986; 70: 32–53.
17. Graham RL. Bounds for certain multiprocessing anomalies. *Bell Syst Tech J* 1966; 45: 1563–1581.
18. Brent RP. The parallel evaluation of general arithmetic expressions. *J ACM* 1974; 21: 201–206.
19. Karp RM and Ramachandran V. Parallel Algorithms for Shared-Memory Machines. In: *Algorithms and complexity: Handbook of theoretical computer science*. Elsevier. Epub ahead of print 1990. DOI: 10.1016/B978-0-444-88071-0.50022-9.
20. Zhang Y, Duchi JC and Wainwright MJ. "Divide and conquer Kernel Ridge Regression: a distributed algorithm with minimax optimal rates," May 2013, [Online]. Available: http://arxiv.org/abs/1305.5029.
21. Dietterich TG. Ensemble methods in machine learning. In: *International workshop on multiple classifier systems*. Springer, 2000, pp.1–15.
22. Breiman L. Bagging predictors. *Mach Learn* 1996; 24: 123–140.
23. Hansen LK and Salamon P. Neural network ensembles. *IEEE Trans Pattern Anal Mach Intell* 1990; 12: 993–1001.
24. Freund Y and Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci* 1997; 55: 119–139.
25. Breiman L. Random forests. *Mach Learn* 2001; 45: 5–32.
26. Elkan C. Boosting and naive Bayesian learning. 1997.
27. Kapil P and Ekbal A. A deep neural network based multi-task learning approach to hate speech detection. *Knowl Based Syst* Dec. 2020; 210: 106458.

28. Aldjanabi W, Dahou A, Al-qaness MAA, et al. Arabic offensive and hate speech detection using a cross-corpora multitask learning model. *Informatics* 2021; 8: 69.
29. Mostafa G, Mahmoud HA, ElHafeez TA, et al. Feature reduction for hepatocellular carcinoma prediction using machine learning algorithms. *J Big Data* 2024; 11. doi: 10.1186/s40537-024-00944-3
30. Moallem P and Razmjooy N. Optimal threshold computing in automatic image thresholding using adaptive particle swarm optimization. *Journal of Applied Research and Technology* 2012; 10. doi: 10.22201/icat.16656423.2012.10.5.361
31. Bertsekas D and Tsitsiklis J. *Parallel and distributed computation : numerical methods*. Division of Simon and Schuster One Lake Street Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.
32. Grama A, Karypis G, Kumar V, et al. *Introduction to parallel computing, second*. Addison-Wesley, 2003.
33. Pelekamoyo JK and Libati HM. Considerations of an efficiency-intelligent geo-localised mobile application for personalised SME market predictions. *Meas Control* 2023; 56: 1788–1797.
34. Singh V, Kumar V, Agha G, et al. Efficient algorithms for parallel sorting on mesh multicomputers. *Int J Parallel Program* 1991; 20: 95–131.
35. Kuncheva LI. *Combining pattern classifiers*. Wiley, 2004.
36. Jordan M, Kleinberg J and Schölkopf B. *Pattern recognition and machine learning*. New York: Springer, 2006. Accessed: Aug. 30, 2024. [Online]. Available: https://link.springer.com/book/9780387310732
37. Tanenbaum A. S. and Van Steen M., *Distributed systems: principles and paradigms*. 2nd ed. Pearson Education. Inc, 2007.
38. Zhou Z-H, Wu J and Tang W. Ensembling neural networks: many could be better than all. *Artif Intell* 2002; 137: 239–263.
39. Pelekamoyo JK and Libati HM. Forecasting market's demand and supply with machine learning and local weather. *International Journal of Scientific & Technology Research* 2022; 11: 115–119. Accessed: Jun. 23, 2023. [Online]. Available: http://www.ijstr.org/paper-references.php?ref=IJSTR-0421-45165.
40. Fagerland MW, Lydersen S and Laake P. "The McNemar test for binary matched-pairs data: mid-p and asymptotic are better than exact conditional," 2013. [Online]. Available: http://www.biomedcentral.com/1471-2288/13/91.
41. Hadhri S, Hadiji M and Labidi W. A voting ensemble classifier for stress detection. *Journal of Information and Telecommunication* 2024; 8: 399–416.
42. Morgan-Benita JA, et al. Hard voting ensemble approach for the detection of type 2 diabetes in Mexican population with non-glucose related features. *Healthcare* 2022; 10: 1362.
43. Dietterich TG. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Mach Learn* 2000; 40: 139–157.
44. Ngo G, Beard R and Chandra R. Evolutionary bagging for ensemble learning. *Neurocomputing* 2022; 510: 1–14.
45. Zhang Y, Ma J, Liang S, et al. A stacking ensemble algorithm for improving the biases of forest aboveground biomass estimations from multiple remotely sensed datasets. *GIsci Remote Sens* 2022; 59: 234–249.
46. Yang M, Cheng L, Cao M, et al. A stacking ensemble learning method to classify the patterns of complex road junctions. *ISPRS Int J Geoinf* 2022; 11: 23.
47. Zhao C, et al. "BoostTree and BoostForest for ensemble learning," Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.09737.