Enhancing Driver Identification with a Crow Search-Optimized Stacking Ensemble

Anwar Mehmood Sohail[1], Khurram Shehzad Khattak[1], Zawar Hussain Khan[2], Ahmad Mustafa[1]

[1]Department of Computer System Engineering, University of Engineering and Technology, Peshawar, Pakistan

[2]College of Computer Science and Engineering, University of Hail Hail, Saudi Arabia

*Correspondence: anwarmehmood.sohail@gmail.com, khurram.s.khattak@gmail.com, z.hussain@uoh.edu.sa, cadet.mustafa2011@gamil.com

Citation | Sohail. A. M, Khattak. K. S, Khan. Z. H, Mustafa. A, "Enhancing Driver Identification with a Crow Search-Optimized Stacking Ensemble", IJIST, Vol. 07 Special Issue pp 244-256, May 2024

DOI: https://doi.org/10.33411/IJIST/1361

Received | April 09, 2025 **Revised |** May 11, 2025 **Accepted |** May 13, 2025 **Published |** May 15, 2025.

Driver identification systems play a crucial role in enhancing vehicle security and delivering personalized experiences for drivers. Traditional identification methods typically use individual machine learning models, which often struggle to perform well due to their limited ability to adapt to diverse driving behaviors. In this study, we present a novel stacking ensemble framework optimized using the Crow Search Algorithm (CSA) to overcome these challenges. The CSA-optimized ensemble combines the strengths of several base models—Logistic Regression (LR), Naïve Bayes (NB), Random Forest (RF), and K-Nearest Neighbour (KNN)—with a meta-learner designed to boost both accuracy and robustness. CSA is used to fine-tune the ensemble's hyperparameters, ensuring optimal performance. Experimental results on a driving dataset demonstrated that the proposed method significantly outperforms existing approaches in terms of identification accuracy, precision, and recall. This framework holds promise for a wide range of applications, including intelligent transportation systems and automotive cybersecurity.

Keywords: Crow Search Algorithm, Driver Identification, Stacking Optimization, Stacking Ensemble, Intelligent Transportation System

Introduction:

Driver identification, also known as driver fingerprinting, refers to the process of recognizing or verifying the identity of a vehicle's operator based on their unique driving behavior [1][2]. These behavioral patterns—shaped by personal characteristics such as age, gender, and driving experience—manifest as distinctive driving styles that act as digital signatures. Advances in in-vehicle sensor technology and the increasing availability of driving data have significantly propelled this research area, enabling the development of data-driven methods to distinguish drivers with high accuracy. Driver identification plays a pivotal role in enhancing safety, enabling personalization, and improving security. It helps prevent unauthorized access, reduces accident risk, and supports personalized in-vehicle experiences by adapting settings to individual preferences [3][4]. Additionally, these systems are foundational to intelligent transportation systems (ITS), where secure and adaptive vehicle interactions are essential [2].

Despite advances, accurately identifying drivers remains a complex challenge. Traditional methods, such as license checks, are slow and error-prone. Modern approaches now leverage machine learning to process sensor-derived behavioral data—such as acceleration, braking, and steering patterns—to achieve real-time and automated driver recognition [5][6]. However, single-model classifiers like Support Vector Machines (SVM) and Decision Trees (DT) often underperform due to their limited adaptability to diverse and noisy data. To improve performance and generalization, ensemble learning—particularly stacking—has emerged as a promising solution. Stacking combines the strengths of multiple base classifiers using a meta-learner, resulting in more robust predictions. However, its success depends on optimal model selection and hyperparameter tuning, which are often resource-intensive when done manually.

This study introduces a Crow Search Algorithm (CSA)-optimized stacking ensemble to address these challenges. CSA is a nature-inspired metaheuristic that effectively balances exploration and exploitation, enabling efficient hyperparameter optimization. Our framework leverages CSA to fine-tune model configurations for improved driver identification accuracy. The primary objective of this study is to develop a high-performance driver identification system that addresses the limitations of traditional models by integrating ensemble learning with metaheuristic optimization. The novelty of the proposed framework lies in:

- The hybrid use of Recursive Feature Elimination (RFE) for dimensionality reduction and CSA for hyperparameter optimization.
- A novel ensemble structure that automatically selects and optimizes base learners to maximize predictive performance.
- Comprehensive empirical evaluation demonstrating superior performance over conventional methods.

The paper begins with a review of existing driver identification techniques in Section II. The proposed methodology is detailed in Section III, followed by a comprehensive explanation of the experimental setup in Section IV. Finally, Section V presents the key findings and outlines future research directions.

Related Work:

The challenge of driver identification has attracted significant attention from researchers, leading to a wide range of proposed solutions—from traditional approaches to modern machine learning techniques. This section explores key contributions across three major areas: single-model approaches, ensemble learning techniques, and optimization algorithms.

Early studies focused on individual classifiers such as SVM, DT, and K-Nearest Neighbors (KNN) for behavior-based driver identification [7]. These models utilized sensor

data capturing acceleration, braking, and steering patterns. While effective to some extent, their limited adaptability to diverse and noisy driving behaviors constrained their performance. Subsequent work explored enhancements in feature engineering and preprocessing. For instance, studies applied techniques like Fast Fourier Transform and sensor calibration to improve recognition accuracy [8]. Some researchers also investigated driver state detection (e.g., intoxication) using simulator data and RF-based feature selection [9], highlighting the potential of behavioral indicators like accelerator depth and lane deviation. However, these methods still struggled with generalization in real-world scenarios.

To overcome the limitations of single models, ensemble methods such as Bagging, Boosting, and Random Forest (RF) have been widely adopted. These techniques leverage multiple learners to improve accuracy and robustness. Among them, stacking ensembles have shown particular promise by combining predictions from several base models through a meta-learner. However, their performance often hinges on appropriate model selection and parameter tuning. Several recent studies have applied stacking in driving-related applications. For example, lane-change intention recognition using stacked models that combine RF, SVM, LSTM, and Bi-LSTM has achieved over 98% accuracy [10][11]. These results underscore the effectiveness of ensemble architectures but also highlight their reliance on complex deep-learning models and finely tuned-parameters.

To automate and enhance the configuration of ensemble models, various nature-inspired optimization algorithms have been explored. Techniques such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) have been successfully used to select optimal base classifiers and fine-tune model parameters [12][13][14]. These metaheuristic approaches have demonstrated superior performance over traditional ensemble setups, particularly when applied to stacking architectures. For instance, ABC-optimized stacking models have shown notable improvements in prediction accuracy and adaptability across datasets [13][15].

Despite these advances, the CSA—a relatively recent addition to nature-inspired optimizers—remains underutilized in ensemble learning contexts. Known for its effective balance between exploration and exploitation, CSA has shown promise in solving complex problems like feature selection and hyperparameter tuning. This study addresses this gap by applying CSA to optimize stacking ensemble configurations for driver identification.

Overall, while prior research has made significant progress using machine learning and optimization for driver identification, challenges remain in achieving high generalizability and automation. By integrating a CSA-based optimization mechanism with a stacking ensemble framework, this study introduces a novel and efficient solution aimed at enhancing identification performance in diverse and dynamic driving environments.

Methodology:

This section presents the methodology used for driver identification through driving behavior analysis, employing an optimized stacking ensemble model in which base model selection is guided by the CSA, as illustrated in Figure 1. The ensemble model is built using a combination of six base learners: RF, KNN, SVM, Logistic Regression (LR), XGBoost, and Naïve Bayes (NB). These models were chosen due to their complementary strengths in handling structured data. Each base model is individually trained on the dataset to capture distinct patterns in driver behavior.

LR is selected as the meta-learner to combine the outputs of the base models and produce the final prediction. This meta-learner acts as a decision-making layer, mitigating the individual weaknesses of the base models by leveraging their combined strengths. The hyperparameters of all base learners are fine-tuned using the CSA. This optimization process enables the ensemble to achieve high performance with minimal computational overhead. An overview of the proposed system is shown in Figure. 1.

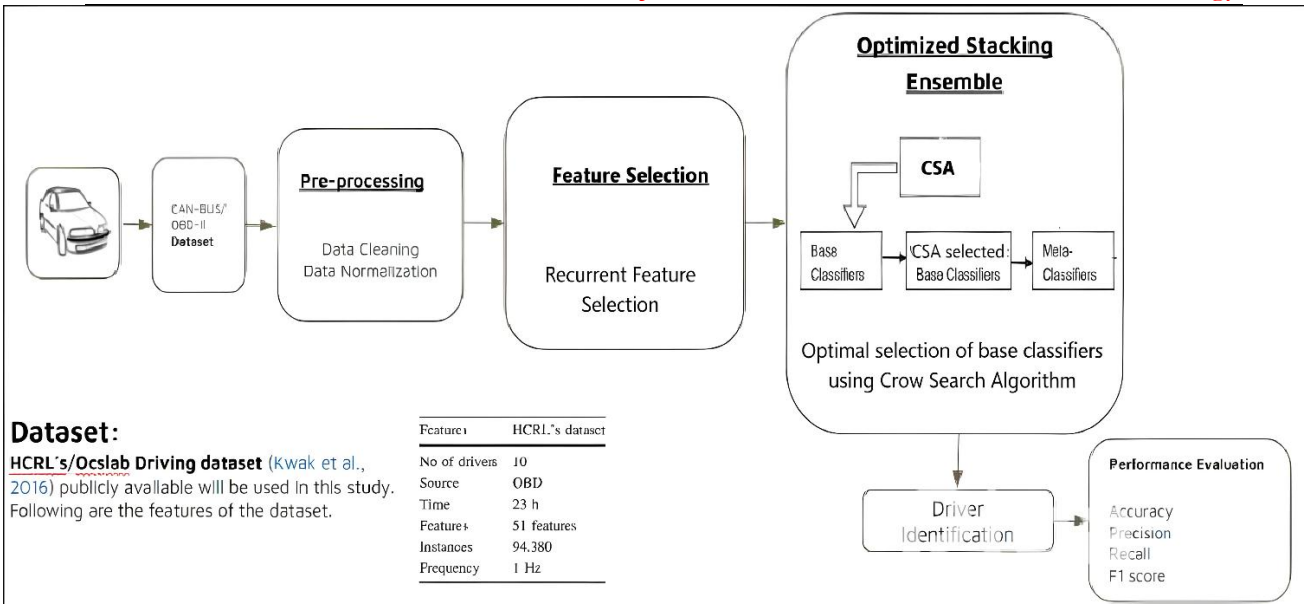


Figure 1. Overview of the Proposed Methodology for Driver Identification

Data Description:

Driver identification heavily relies on behavioral data collected through instrumented vehicles equipped with various sensors, including accelerometers, gyroscopes, GPS modules, and OBD. These sensors record crucial driving patterns such as acceleration, braking, steering angles, and speed. To ensure the system's robustness, data was collected from multiple drivers representing diverse driving styles.

In this study, we utilize the driving behavior dataset provided by the Hacking and Countermeasures Research Lab (HCRL) [16]. This dataset includes 51 features extracted from in-vehicle CAN-BUS data and OBD-II. It encompasses driving data from ten individuals on three different road types in Seoul—highways, city streets, and parking lots—covering a total distance of 23 kilometers. Once the raw driving data is collected, it must be preprocessed to ensure quality and consistency for model training.

Data Preprocessing:

High-quality data is critical for building an effective driver identification system. The preprocessing phase includes the following tasks:

1. **Handling Missing Values:** Missing values, often caused by sensor errors or data transmission issues, are handled using techniques such as interpolation or statistical imputation to ensure data completeness.
2. **Normalization:** Normalization is applied to scale all feature values to a common range. This standardization ensures that features with larger numerical ranges do not disproportionately influence the model's learning process.

Overall, preprocessing involves cleaning the data, ensuring consistency, and preparing relevant features for subsequent modeling stages. After data preprocessing, we focus on selecting the most relevant features to improve model performance.

Feature Selection:

To improve classification accuracy and computational efficiency, feature selection is applied. The objective is to reduce the feature space by retaining only the most relevant and non-redundant attributes.

A correlation heatmap is used as a visual tool to examine the relationships among features. This helps identify highly correlated features that may introduce multicollinearity, which can reduce model interpretability and stability. As illustrated in Figure. 2, the heatmap

supports informed decision-making during feature selection, ensuring that the chosen features enhance the model's performance while minimizing redundancy.

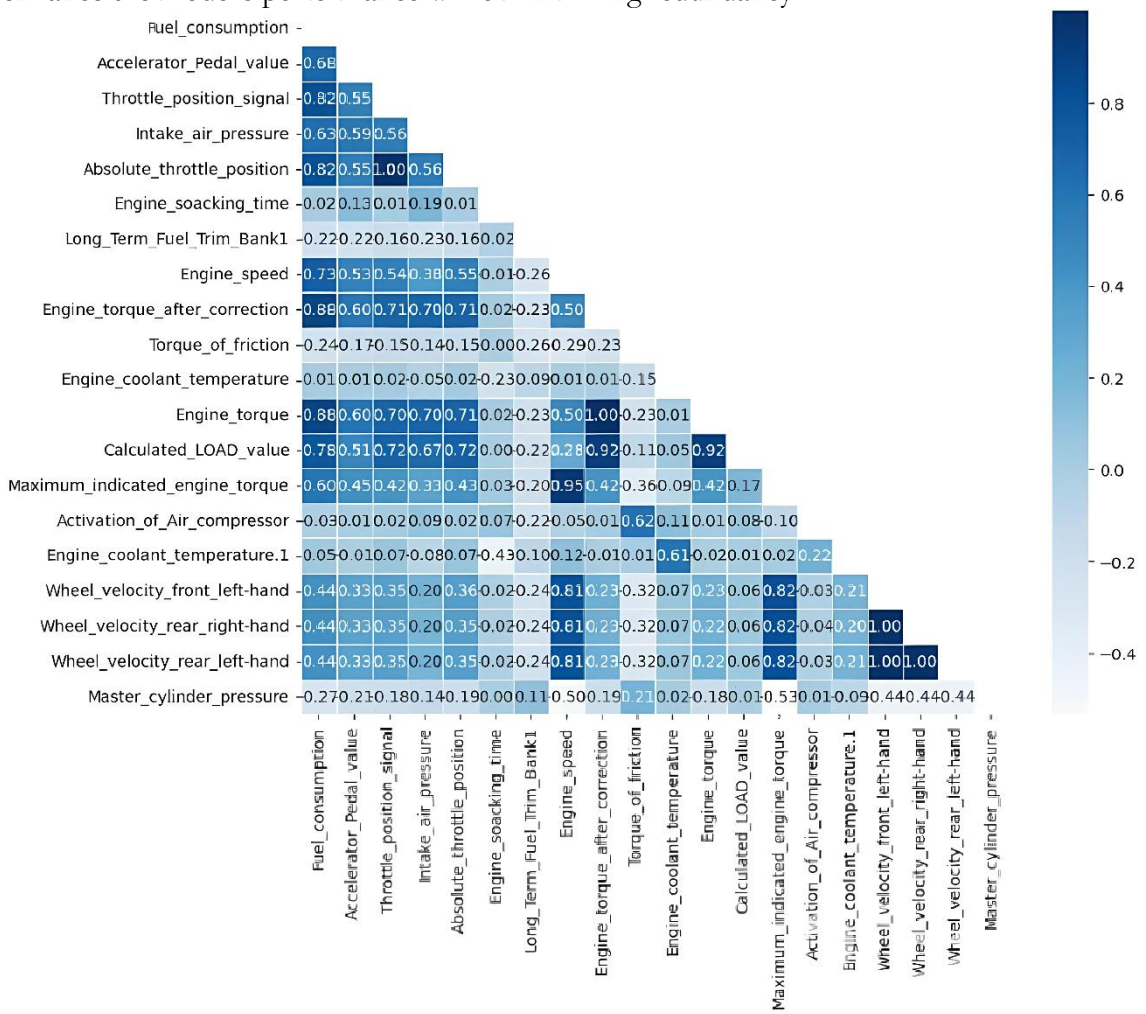


Figure 2. Correlation Heatmap of Selected Features

Recursive Feature Elimination (RFE):

RFE is employed as the primary technique for feature selection in this study. RFE works by recursively removing the least significant features and building a model using the remaining subset. The importance of each feature is evaluated based on its contribution to the predictive performance of a machine learning model, such as SVM or RF.

The RFE process involves the following steps:

1. **Initial Model Training:** A machine learning model is trained using the complete set of features.
2. **Feature Ranking:** Features are ranked according to importance scores, which are derived from model coefficients (for linear models) or feature importance metrics (for tree-based models).
3. **Feature Elimination:** The least important features are removed.
4. **Iteration:** Steps 2 and 3 are repeated until an optimal subset of features is selected, based on evaluation metrics such as classification accuracy.

RFE assists in identifying the most informative features that significantly contribute to the classification task, thereby improving generalization and reducing model complexity.

The final set of 20 features selected using RFE includes:

- Engine_soaking_time
- Fuel_consumption

- Accelerator_Pedal_value
- Throttle_position_signal
- Intake_air_pressure
- Absolute_throttle_position
- Long_Term_Fuel_Trim_Bank1
- Engine_speed
- Engine_torque_after_correction
- Torque_of_friction
- Engine_coolant_temperature
- Engine_torque
- Calculated_LOAD_value
- Maximum_indicated_engine_torque
- Activation_of_Air_compressor
- Wheel_velocity_front_left-hand
- Wheel_velocity_rear_right-hand
- Wheel_velocity_rear_left-hand
- Master_cylinder_pressure

Stacking Ensemble Classification:

Stacking is a powerful ensemble learning technique that combines multiple models to improve overall classification performance. It operates on a two-level architecture:

- **Level 1 (Base Learners):** A diverse set of classifiers (e.g., RF, KNN, SVM, NB, XGBoost, etc.) are trained independently on the training dataset. Their outputs (predictions or class probabilities) form a new dataset.
- **Level 2 (Meta Learner):** An LR model is employed as the meta-learner, trained on the outputs from the base models to make final predictions.

The base learner selection is a critical aspect of stacking and is performed using the CSA, which optimizes the choice of classifiers and their configurations.

The architecture of the stacking model is depicted in Figure. 3. The original input data is first processed through the base-level models (Level 1), and their predictions are then fed into the meta-level model (Level 2). This layered approach allows the meta-learner to capture patterns missed by individual classifiers, thereby enhancing model accuracy and robustness.

By leveraging the strengths of heterogeneous models and using CSA for optimal base model selection, the proposed stacking ensemble significantly improves the performance of the driver identification system.

Stacking Optimization Using CSA:

The selection of base models plays a crucial role in constructing an effective stacking ensemble classifier. To ensure optimal performance, this study employs the **CSA** to identify the most suitable combination of base classifiers from a pool of candidate models.

Crow Search Algorithm (CSA):

CSA is a population-based metaheuristic optimization algorithm inspired by the intelligent foraging behavior of crows. Introduced by Askarzadeh in 2016 [17], CSA mimics how crows use memory and deception to locate and protect food resources. Its biological inspiration and competitive performance make it suitable for applications in engineering, machine learning, and data science.

The algorithm operates through two main mechanisms:

- **Exploration**: Crows explore the search space by randomly moving to new positions, which promotes diversity and prevents premature convergence.
- **Exploitation**: Crows use memory to move toward the best-known solutions (hiding places), with a probability of deception (controlled by the Awareness Probability, AP) to avoid being followed.

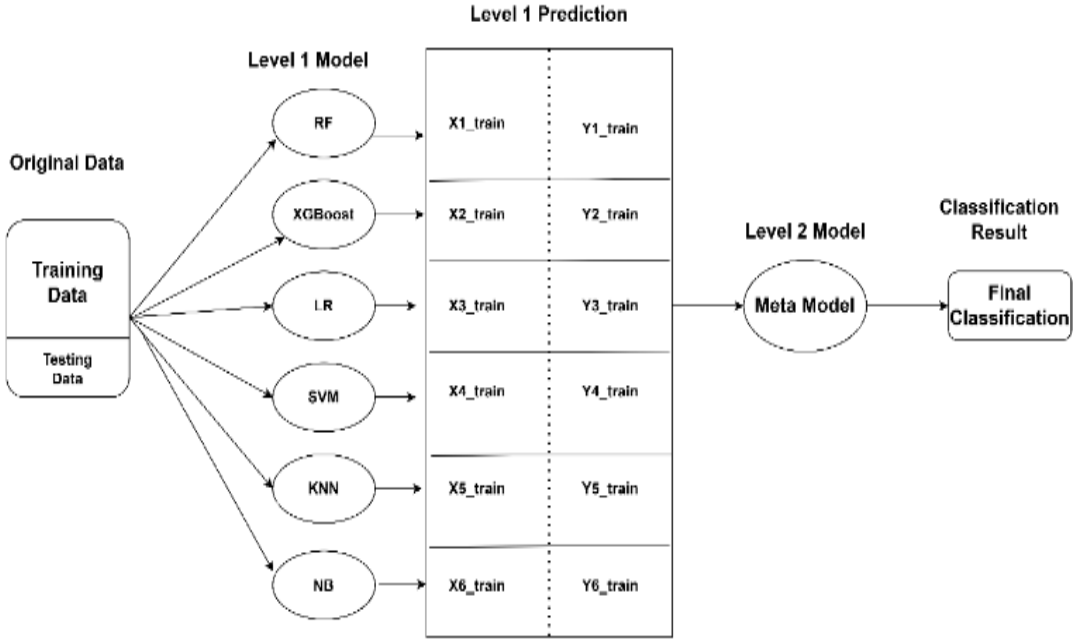


Figure 3. Architecture of the Stacking Ensemble Model

The general behavior of the CSA is illustrated in Figure. 4.

CSA Implementation for Base Model Selection:

In this study, CSA is applied to select the optimal combination of base learners for the stacking ensemble. The implementation involves the following steps:

- **Initialization**: A population (swarm) of n crows is randomly initialized in a d-dimensional space, where each crow represents a unique combination of base classifiers.
- **Fitness Evaluation**: Each crow's position (i.e., selected classifier combination) is evaluated using a fitness function that reflects the classification performance (e.g., accuracy or F1 score) of the resulting stacking ensemble. The fitness value is stored as the crow's memory mi, representing its best-known solution.
- **Position Update**: Each crow selects another crow at random, denoted as xj, and generates a random number r.
  - If r > AP, the crow xi moves toward the memory location mj of crow xj, simulating the act of following another crow's hiding place.
  - If r ≤ AP, the crow moves randomly, representing deceptive behavior to mislead others.
- **Iteration**: The process is repeated iteratively, updating the memory and positions of all crows, until a convergence criterion is met or a predefined number of iterations is reached.

By balancing exploration and exploitation, CSA enables the automatic selection of an optimal base model set that enhances the predictive power of the stacking ensemble without exhaustive manual testing.

- Crow updates its position by selecting a random other crow i.e. x_j and following it to know m_j. Then new x_j is calculated as follows:

$$x_{i,iter} + r_i \times fl_{i,iter} \times \left(m_{j,iter} - x_{i,iter}\right) \quad r_j \geq AP_{j,iter} \quad \text{(1)}$$
$$x_{i,iter+1} = \{ a\ random\ position \quad otherwise$$

Where $AP_{j,iter}$ refers to crow j awareness probability, iter refers to iteration number, r_i, r_j refers to random numbers,

$fl_{i,iter}$ is the crow i flight length to denote crow j memory.

- Updating memory

$$mi,iter+1 = \begin{cases} xi,iter & f(x_{i,iter+1}) \le f(m_{i,iter}) \\ m_{i,ier} & otherwise \end{cases} \quad (2)$$

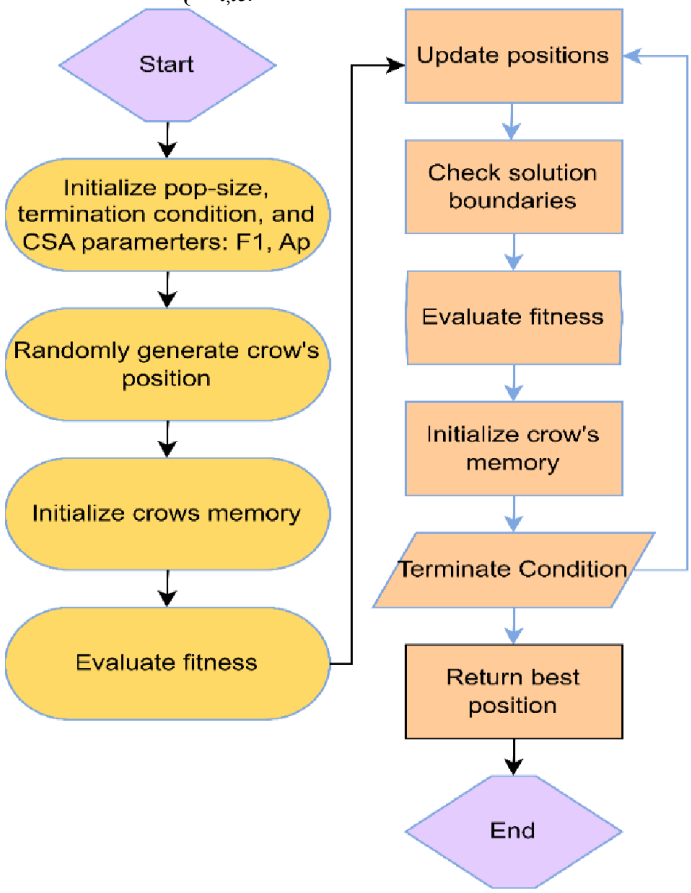


Figure 4. Crow Search Algorithm Architecture

CSA-Selected Base Models for the Stacking Ensemble:

Based on the optimization performed using the CSA, the following machine-learning algorithms were selected as the optimal base learners for the stacking ensemble:

- RF Classifier
- LR
- KNN Classifier
- NB

These models were chosen due to their complementary strengths in capturing various patterns in driver behavior, as identified during the CSA optimization process. Meanwhile, XGBoost and SVM were excluded from the final ensemble configuration, indicating that their inclusion did not enhance the overall performance of the stacked model. This selection outcome is illustrated in Figure. 5, which visualizes the base model selection process and the comparative performance of candidate classifiers.

Performance Evaluation:

By applying CSA, the optimal combination of base models is selected, resulting in a more effective stacking ensemble classifier with enhanced predictive accuracy. The performance of the proposed driver identification system is rigorously assessed using widely recognized evaluation metrics to ensure the robustness and reliability of the classification results:

- **Accuracy**: Measures the proportion of correctly identified drivers out of the total number of test instances.
- **Precision**: Indicates the ratio of true positive predictions to the total predicted positives, reflecting the classifier's ability to avoid false positives.
- **Recall**: This represents the ratio of true positives to the actual positives in the dataset, highlighting the model's ability to capture all relevant instances.
- **F1-Score**: The harmonic mean of precision and recall, offering a balanced metric that accounts for both false positives and false negatives.

These metrics collectively provide a comprehensive evaluation of the classifier's performance in real-world driver identification scenarios.

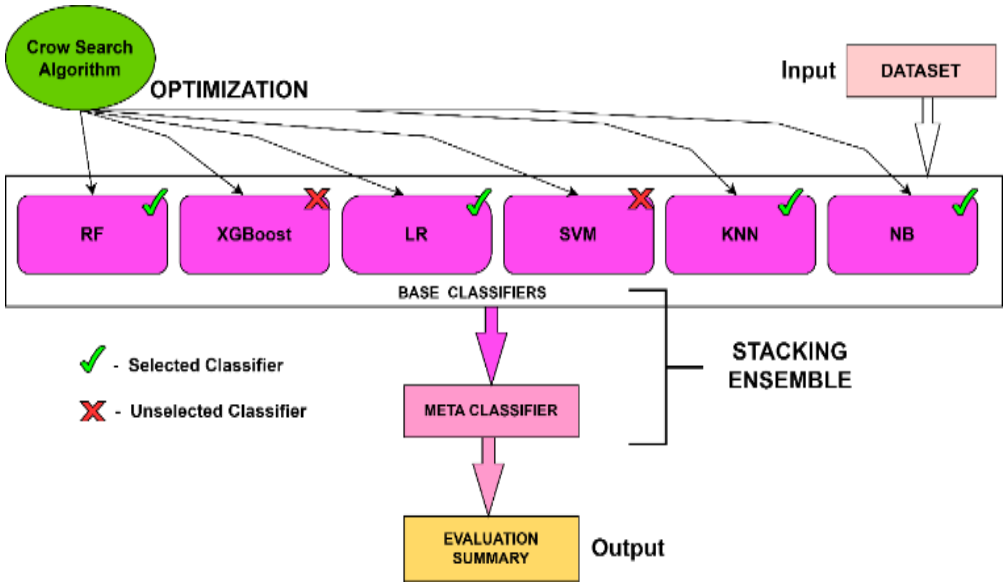


Figure 5. Stacking Ensemble Framework with CSA for Classifier Selection

Results:

All experiments were conducted using Google Colab, leveraging its cloud-based computational resources. The framework was implemented in Python, utilizing popular machine-learning libraries such as Scikit-learn**,** XGBoost, and NumPy. To enhance training and optimization efficiency, parallel processing techniques were employed, ensuring optimal usage of available computing power.

The experimental setup was carefully designed to simulate realistic conditions for driver identification, ensuring a fair and meaningful evaluation of the proposed system. The optimized stacking ensemble model, with base classifiers selected via CSA and hyperparameters tuned accordingly, achieved the following performance metrics:

- **Accuracy**: 0.9843
- **Precision**: 0.9843
- **Recall**: 0.9843
- **F1 Score**: 0.9843

These results demonstrate the superior predictive power and balanced classification capabilities of the proposed approach.

To contextualize these results, six machine learning models were evaluated and compared based on Accuracy**,** Precision**,** Recall, and F1 Score. The Optimized Stacking Model outperformed all other models across every metric, reflecting its near-perfect classification ability and strong generalization to both positive and negative instances. A comparative summary is presented in Table 1**.**

- **Optimized Stacking Model**: Achieved the best performance with 98.43% across all metrics, confirming the effectiveness of CSA-based model selection.
- **Standard Stacking Ensemble**: Showed strong results with 95.83% accuracy, indicating the inherent strength of ensemble learning even without optimization.
- **RF** and **KNN**: Provided moderate to good results, with accuracies of 90.69% and 85.73%, respectively, demonstrating reasonable classification capability.

These findings validate the proposed method's robustness and effectiveness in driver identification tasks. The accuracy of each model is shown in Figure. 6.

Table 1. Comparative Performance of Machine Learning Models Based on Classification Metrics

Metric	Random Forest	Logistic Regression	K-Nearest Neighbor	Naive Bayes	Stacking Ensemble	Optimized Stacking Model
Accuracy (%)	90.69	57.39	85.73	31.99	95.83	98.43
Precision	0.9106	0.5745	0.8570	0.4048	0.9582	0.9843
Recall	0.9069	0.5739	0.8573	0.3199	0.9583	0.9843
F1 Score	0.9062	0.5669	0.8569	0.3015	0.9582	0.9843

Evaluation of Model Performance and Overfitting Mitigation:

In our evaluation, LR exhibited relatively lower performance with an accuracy of 57.39%. This suggests that the model may not be well-suited for the driver identification task, as it struggled to correctly classify instances across the dataset. The low performance of LR indicates that its linear decision boundaries may not capture the complex, non-linear patterns inherent in driving behavior data.

NB, on the other hand, demonstrated the lowest performance, with an accuracy of 31.99%**.** This result underscores the challenges that NB faces when its assumptions, such as feature independence and Gaussian distribution, do not align well with the driving behavior data. Consequently, the Precision, Recall, and F1 Scores were also significantly low, reaffirming the model's limited efficacy for this task.

To mitigate the possibility of overfitting, particularly in light of the near-perfect results from the optimized stacking ensemble, several strategies were implemented during the experimental process:

1. **K-Fold Cross-Validation (k = 5)**: This technique partitions the dataset into five equally sized folds. For each iteration, the model is trained on $k-1$ folds and validated on the remaining fold, ensuring that every data point is used for both training and validation. This process provides a more comprehensive view of the model's performance, helping to prevent the model from fitting too closely to any one subset of the data.
2. **Dataset Balancing**: To avoid class imbalance biases, the dataset was balanced during training. This step ensures that the model generalizes better across all instances, reducing the likelihood of the model being biased towards the majority class and improving its robustness.
3. **Data Leakage Prevention**: Special care was taken to avoid data leakage throughout the experimentation process. All data transformations, including normalization and encoding, were applied strictly within the training set and then independently transferred to the validation and testing sets. This careful separation prevented any unintended influence of the test data on model training, ensuring that the model's evaluation metrics were fair and reliable.

These combined strategies ensured minimal risk of overfitting. The model showed consistent performance across the various cross-validation folds and a separate holdout test set, confirming the robustness of the proposed system.

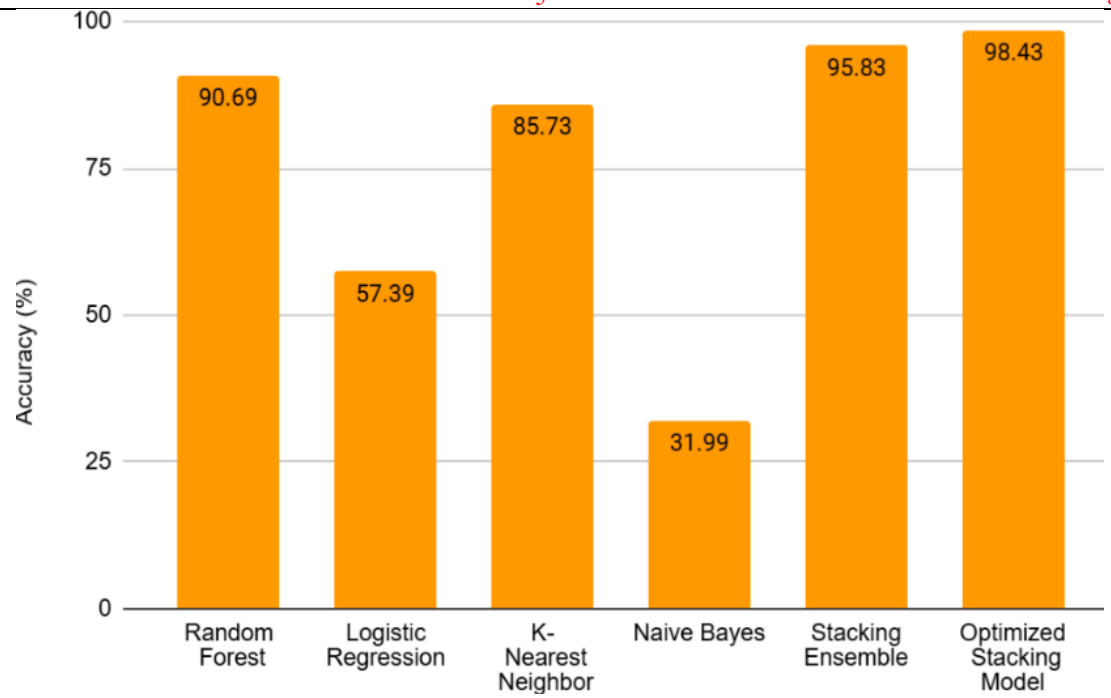


Figure 6. Accuracy of the models.

Discussion:

Results obtained from this study further strengthen the facts of the integration of CSA in stacking ensemble with driver identification. The stacking ensemble optimized by CSA achieved the highest accuracy value of 98.43%, which surpasses the performance of individual classifiers like RF (90.69%), KNN (85.73%), LR (57.39%), and NB (31.99%) and that of the standard, non-optimized stacking ensemble (95.83%). Top-notch improvement comes from two things: (1) smart hyperparameter tuning and base model selection by CSA, which ensures classifier feasibility, and (2) using RFE, thus reducing noise and selecting the most informative subset of the features.

Most importantly, though, its efficiency in computation makes it stand apart from others as compared to conventional state-of-the-art models available in the literature such as Bi-LSTM or attention-based architectures, reporting the accuracies of around 98.24% [10] and 95.54% [11] respectively. Unlike deep learning models, which usually operate with large training datasets, high-performance GPUs, and long training times, the CSA-optimized stacking ensemble is lightweight and well-suited for real-time applications in ITS.

The findings also highlight the shortcomings of classical models such as NB and LR concerning high-dimensional, non-linear behavioral data. The low performance of these models can be attributed to simplistic assumptions, such as feature independence in NB and linearity of decision boundaries in LR.

Additionally, this research showcases the model's generalizability and strength. Through the use of methods such as k-fold cross-validation, data balancing, and a clear division of training and testing datasets (to avoid data leakage), we guaranteed that the performance measured is not a consequence of overfitting. The model showed stable results throughout all validation folds, indicating its viability for real-world use.

Despite these strengths, certain limitations should be acknowledged. First, the dataset comes from a limited set of ten drivers in a controlled setting and we might not see an exhaustive set of variation across populations in real-world driving scenarios. Second, even though CSA was successful, its performance and convergence rate may have to be compared with newer metaheuristic optimizers like Grey Wolf Optimizer (GWO), Firefly Algorithm (FA), or PSO variants in the future to test their results. Furthermore, no real-time inference

latency and performance on embedded systems or edge devices were tested that should be considered in coming future.

Conclusion:

This study presents a CSA-optimized stacking ensemble framework for driver identification, achieving significant improvements in predictive performance compared to traditional methods. By leveraging the complementary strengths of ensemble learning and CSA-based optimization, the framework effectively addresses key challenges in driver identification, such as high variability in driver behavior and the complexity of sensor data.

Previous studies, such as those in [10] and [11], demonstrated the advantages of integrating multiple classifiers, including RF**,** SVM**,** Long Short-Term Memory (LSTM), and Bidirectional LSTM (Bi-LSTM), with reported accuracy levels of 98.24% and 95.54%**,** respectively. In this study, the proposed framework further improves upon these results, achieving an accuracy of 98.43%, thereby surpassing the performance of existing models in driver identification tasks.

While the current study validates the framework using a controlled dataset, its potential for real-world applications in ITS is substantial. Future work will focus on deploying the framework in real-world fleet environments, enabling the evaluation of its adaptability and robustness under diverse conditions. These conditions may include factors such as unseen drivers, varying environmental conditions (e.g., weather, traffic patterns), and different vehicle types**.**

Additionally, the integration of dynamic datasets from connected vehicles and smart infrastructure will enhance the scalability and real-time applicability of the system. Another promising direction is the incorporation of edge computing techniques to enable on-device processing, ensuring low-latency, efficient operation in real-time ITS environments.

These future efforts will contribute to a more comprehensive evaluation of the framework's performance and its practical usability in large-scale ITS deployments, ultimately paving the way for its application in advanced transportation systems that require scalable, adaptive, and real-time driver identification solutions.

References:

[1] A. M. Sohail, T. Y. Wah, N. A. Khan, Z. A. Khan, and A. Koubâa, "A Comprehensive Review: Analysis of Machine Learning, Deep Learning, and Large Language Model Techniques for Revolutionizing Driver Identification," *Authorea Prepr.*, Mar. 2024, doi: 10.36227/TECHRXIV.170974128.87339641/V1.

[2] E. K. Pei-Yu Tseng, Po-Ching Lin, "Vehicle theft detection by generative adversarial networks on driving behavior," *Eng. Appl. Artif. Intell.*, vol. 117, p. 105571, 2023, doi: https://doi.org/10.1016/j.engappai.2022.105571.

[3] S. Hernández Sánchez, R. F. Pozo, and L. A. H. Gómez, "Driver Identification and Verification from Smartphone Accelerometers Using Deep Neural Networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 97–109, Jan. 2022, doi: 10.1109/TITS.2020.3008210.

[4] K. Gunapriya Balan, Singaravelan Arumugam, Suresh Muthusamy, Hitesh Panchal, Hossam Kotb, Mohit Bajaj, Sherif S. M. Ghoneim, "An Improved Deep Learning-Based Technique for Driver Detection and Driver Assistance in Electric Vehicles with Better Performance," *Int. Trans. Electr. Energy Syst.*, 2022, doi: https://doi.org/10.1155/2022/8548172.

[5] K. A. N. Ayaz Shehzad, Khattak Khurram S, Khan Zawar H, Minallah Nasru, Khan Mushtaq A, "Sensing technologies for traffic flow characterization: From heterogeneous traffic perspective," *J. Appl. Eng. Sci.*, vol. 20, no. 1, pp. 29–40, 2022, [Online]. Available: https://scindeks.ceon.rs/Article.aspx?artid=1451-41172201029A

[6] A. N. K. ALI ZEB, KHURRAM S. KHATTAK, AREEB AGHA, ZAWAR H.KHAN, M. ATHAR JAVED SETHI, "On-Board Diagnostic (OBD-II) Based Cyber

Physical System for Road Bottlenecks Detection," *J. Eng. Sci. Technol.*, vol. 17, no. 2, pp. 0906–0922, 2022, [Online]. Available: https://jestec.taylors.edu.my/Vol 17 Issue 2 April 2022/17_2_07.pdf
[7] et al Li, Z., "Driver identification in intelligent vehicle systems using machine learning algorithms," *IET Intell. Transp. Syst.*, vol. 13, no. 1, pp. 40–47, 2019, [Online]. Available: http://crossref.org/metadatamanager/publications/10.33411%2Fijasd/addarticle
[8] Y. X. Huihuan Qian, Yongsheng Ou, Xinyu Wu, Xiaoning Meng, "Support Vector Machine for Behavior-Based Driver Identification System," *J. Robot.*, 2010, doi: https://doi.org/10.1155/2010/397865.
[9] X. Z. ZhenLong Li, HaoXin Wang, "Random forest–based feature selection and detection method for drunk driving recognition," *Sage Journals*, 2020, doi: https://doi.org/10.1177/1550147720905234.
[10] H. Zhang, S. Gao, and Y. Guo, "Driver Lane-Changing Intention Recognition Based on Stacking Ensemble Learning in the Connected Environment: A Driving Simulator Study," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 2, pp. 1503–1518, Feb. 2024, doi: 10.1109/TITS.2023.3314443.
[11] R. F. Hongjia Zhang, Yingshi Guo, Chang Wang, "Stacking-based ensemble learning method for the recognition of the preceding vehicle lane-changing manoeuvre: A naturalistic driving study on the highway," *IET Intell. Transp. Syst.*, 2021, doi: https://doi.org/10.1049/itr2.12154.
[12] Y. Chen and M. L. Wong, "Optimizing stacking ensemble by an ant colony optimization approach," *Genet. Evol. Comput. Conf. GECCO'11 - Companion Publ.*, pp. 7–8, 2011, doi: 10.1145/2001858.2001863;GROUPTOPIC:TOPIC:ACM-PUBTYPE>PROCEEDING;CTYPE:STRING:BOOK.
[13] S. K. P. Shunmugapriya, "Optimization of stacking ensemble configurations through Artificial Bee Colony algorithm," *Swarm Evol. Comput.*, vol. 12, pp. 24–32, 2013, [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S2210650213000199?via%3Dihub
[14] A. Gupta and A. R. Thakkar, "Optimization of stacking ensemble Configuration based on various metahueristic algorithms," *Souvenir 2014 IEEE Int. Adv. Comput. Conf. IACC 2014*, pp. 444–451, 2014, doi: 10.1109/IADCC.2014.6779365.
[15] G. Priyadharshini and M. F. Ukrit, "Stacking optimized with artificial bee colony for driving style classification by feature reconstruction from OBD II data," *Soft Comput.*, vol. 27, no. 1, pp. 591–603, Jan. 2023, doi: 10.1007/S00500-022-07135-3/METRICS.
[16] B. Il Kwak, J. Y. Woo, and H. K. Kim, "Know your master: Driver profiling-based anti-theft method," *2016 14th Annu. Conf. Privacy, Secur. Trust. PST 2016*, pp. 211–218, 2016, doi: 10.1109/PST.2016.7906929.
[17] Alireza Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Comput. Struct.*, vol. 169, pp. 1–12, 2016, doi: https://doi.org/10.1016/j.compstruc.2016.03.001.

Copyright © by authors and 50Sea. This work is licensed under Creative Commons Attribution 4.0 International License.