International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 10, Issue 4, April-2025, http://ijmec.com/, ISSN: 2456-4265

Agrisphere : Smart Farming Solutions For Better Yields

Ms. Hafsa Tasneem [*1],
Mr. MD Khalid [*2], Mr. Ryan ur Rahman [*3], MOHD Madani Ibrahim Ali Khan [*4]
[*1] Assistant Professor, Dept. of AIML-Lords Institute of Engineering and Technology
[*2, 3, 4] B.E Student Dept. of AIML, Lords Institute of Engineering and Technology
hafsatasneem@lords.ac.in[*1], mdkhalid2702@gmail.com[*2], ryan9652q@gmail.com*[3], qadriasma3@gmail.com[*4]

Abstract: This project is a Crop Prediction System developed using React (frontend) and Node.js with Express (backend). It predicts the most suitable crop for cultivation based on user-inputted environmental parameters such as nitrogen, phosphorus, potassium, temperature, humidity, pH, and rainfall. The frontend collects user data and sends it to the backend via a POST request, where the server processes it and returns a crop prediction along with crop details like description and image. Additional features include a to-do list for task management and a Google search bar for easy web browsing. A visualization component graphically displays the user's input data for better analysis. The project structure separates frontend and backend concerns clearly, enhancing maintainability. Users interact through a simple web interface hosted locally. Future improvements could involve integrating an actual machine learning model, expanding the crop database with more details and images, enhancing visualizations with advanced charts, and adding user authentication for a personalized experience.

INTRODUCTION

GENERAL

Agriculture is a key sector that directly affects food security and economic stability. Farmers often face challenges when deciding which crops to cultivate, as crop selection depends on multiple environmental factors such as soil nutrients, weather conditions, and rainfall patterns. Traditional farming methods rely heavily on experience and manual observations, which may not always be accurate or efficient. With the advancement of technology, predictive systems have been developed to assist farmers in making more informed decisions. By analyzing key parameters, modern systems can suggest the most suitable crops for a given environment. This project aims to leverage technology by building a Crop Prediction System that simplifies decision-making for farmers and users, promoting better yield, resource management, and sustainable agricultural practices

PROJECT OVERVIEW

The Crop Prediction System is a web-based application designed to assist users in selecting the most suitable crop based on specific environmental parameters. Developed using React for the frontend and Node.js with Express for the backend, the system collects user inputs like nitrogen, phosphorus, potassium levels, temperature, humidity, pH, and rainfall. It processes this data and predicts an appropriate crop, displaying relevant details and visuals. The application also includes additional features such as a task management to-do list and an integrated Google search bar. This project aims to combine ease of use, data visualization, and practical functionality for a better user experience.

OBJECTIVE

- To develop a user-friendly web application that predicts the most suitable crop based on environmental parameters like soil nutrients, temperature, humidity, pH, and rainfall.
- To enable quick and accurate crop suggestions using backend data processing and deliver informative results through an interactive frontend interface.
- To integrate additional features like a to-do list and Google search bar for enhancing user productivity and overall application usability.
- To provide graphical visualization of input data, helping users better understand environmental conditions influencing crop selection.

LITERATURE SURVEY:

- Crop Prediction Using Machine Learning

o Summary: Machine learning models like Decision Trees, Random Forests, and Neural Networks have been widely used for crop prediction. These models analyze environmental parameters such as soil nutrients, temperature, and rainfall to predict the most suitable crop.

- Relevance: Your project can integrate a machine learning model in the backend to replace the dummy logic for crop prediction.
- Reference: Patel, H., & Patel, D. (2016). "Crop Prediction Using Machine Learning Techniques." International Journal of Computer Applications.
- 2. Google Custom Search API
- Summary: The Google Custom Search API allows developers to integrate Google search functionality into their applications. It provides a way to fetch search results programmatically.
- Relevance: Your project uses the Google Custom Search API to display search results dynamically on the same page.
- Reference: Google Developers Documentation. "Custom Search JSON API." Link.
- 3. Visualization of Agricultural Data
- Summary: Visualization tools like Chart.js and D3.js are commonly used to represent agricultural data graphically. These tools help users understand trends and patterns in data.
- Relevance: Your project uses a Visualization component to display input data graphically, enhancing user experience.
- Reference: Heer, J., & Shneiderman, B. (2012). "Interactive Dynamics for Visual Analysis." Communications of the ACM.
4. To-Do List Applications
- Summary: To-do list applications are simple productivity tools that allow users to manage tasks. They are often integrated into larger systems to enhance usability.
- Relevance: Your project includes a to-do list feature, allowing users to manage tasks alongside crop prediction.
- Reference: Bellotti, V., & Bly, S. (2004). "Walking Away from the Desktop Computer:
- Distributed Collaboration and Mobility in a Product Design Team." Proceedings of CSCW.
- 5. Soil Nutrient Analysis
- Summary: Soil nutrient analysis is critical for determining the suitability of crops. Studies have shown that nitrogen, phosphorus, and potassium levels significantly impact crop yield.
- Relevance: Your project collects soil nutrient data (nitrogen, phosphorus, potassium) as input
- for crop prediction.
- Reference: Brady, N. C., & Weil, R. R. (2008). "The Nature and Properties of Soils." Pearson Education.
- 6. Climate Impact on Agriculture
- Summary: Climate factors like temperature, humidity, and rainfall play a crucial role in crop selection. Predictive models often incorporate these parameters to improve accuracy.
- Relevance: Your project uses climate data (temperature, humidity, rainfall) as input for crop prediction.
- Reference: Lobell, D. B., & Burke, M. B. (2010). "On the Use of Statistical Models to Predict Crop Yield Responses to Climate Change." Agricultural and Forest Meteorology.
- 7. User-Centered Design in Agricultural Tools
- Summary: User-centered design principles ensure that agricultural tools are intuitive and easy to use. Features like clear layouts and responsive designs improve user experience.
- Relevance: Your project incorporates a professional layout with a clear separation of features (crop prediction, to-do list, search bar).
- Reference: Norman, D. A. (2013). "The Design of Everyday Things." Basic Books.
- 8. Open-Source Tools for Web Development
- Summary: Frameworks like React and libraries like Tailwind CSS are widely used for building modern web applications. They enable developers to create responsive and scalable interfaces. Relevance: Your

project uses React for the frontend and Tailwind CSS for styling.

- Reference: React Documentation. "React – A JavaScript Library for Building User Interfaces." Link.
- 9. RESTful API Design
- Summary: RESTful APIs are a standard for building web services. They allow seamless communication between the frontend and backend of an application.
- Relevance: Your project uses a RESTful API (/predict) to send data from the frontend to the backend and receive predictions.
- Reference: Fielding, R. T. (2000). "Architectural Styles and the Design of Network-based Software Architectures." Doctoral Dissertation, University of California, Irvine.
- 10. Integration of Productivity Tools in Agriculture
- Summary: Integrating productivity tools like task managers and search engines into agricultural systems can improve efficiency and decision-making.
- Relevance: Your project combines crop prediction with a to-do list and a search bar, making it a multi-functional tool.
- Reference: Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M. J. (2017). "Big Data in Smart Farming – A Review." Agricultural Systems.

SYSTEM ANALYSIS

EXISTING SYSTEM

The existing systems for crop prediction and agricultural management are often fragmented and lack integration. Key limitations include:

Manual Decision-Making: Farmers rely on traditional knowledge or manual methods to decide which crop to grow, which may not consider all environmental factors.
Lack of Integration: Existing tools often focus on a single feature, such as crop prediction or task management, but do not combine multiple functionalities like prediction, task management, and search.
Limited Accessibility: Many systems require advanced technical knowledge or expensive hardware, making them inaccessible to small-scale farmers.
No Real-Time Insights: Most systems do not provide real-time insights or allow users to visualize their input data effectively.

PROPOSED SYSTEM

The proposed system is a Crop Prediction and Management Tool that combines multiple functionalities into a single, user-friendly platform. Built with modern web technologies, it assists farmers and agricultural enthusiasts by predicting the most suitable crop based on environmental parameters such as nitrogen, phosphorus, potassium, temperature, humidity, pH, and rainfall. It provides detailed crop information, including descriptions and images, alongside a graphical visualization of input data for better understanding. Additional features include a to-do list for task management and Google search integration, allowing users to fetch search results directly within the app. The platform boasts a clean, responsive UI/UX, with crop prediction and task management placed side-byside for improved usability. A Node.js backend handles data processing through RESTful APIs, ensuring smooth communication between frontend and backend. Advantages of the system include real-time insights, easy accessibility through any modern web browser, integrated features enhancing productivity, cost-effectiveness through open-source technologies, scalability for future updates, and data visualization to support informed agricultural decisions.

REQUIREMENT SPECIFICATIONS

SOFTWARE REQUIREMENTS

- The software requirements for the project include the tools, libraries, and frameworks necessary to develop and run the application.
- Frontend:
- React: A JavaScript library for building user interfaces.
- Vite: A fast build tool for modern web development.
- Tailwind CSS: A utility-first CSS framework for styling.

- Chart.js: A JavaScript library for creating charts and visualizations.
- React-Chartjs-2: A React wrapper for Chart.js.
- Google Custom Search API: For integrating search functionality.
- Backend:
- Node.js: A JavaScript runtime for building the backend server.
- Express.js: A web framework for Node.js to handle API requests.
- CORS: Middleware to enable cross-origin requests.
- Development Tools: Visual Studio Code: A code editor for writing and managing the project.
- Postman (optional): For testing backend APIs.
- Git: For version control and collaboration.
- Dependencies (from package.json): react, react-dom, vite, chart.js, react-chartjs-2, express, cors.
- Browser: A modern web browser like Google Chrome, Mozilla Firefox, or Microsoft Edge.
- Operating System:
- Windows 10/11, macOS, or any Linux distribution.

HARDWARE REQUIREMENTS

-
- The hardware requirements specify the minimum and recommended configurations for running the
- application.
- Minimum Requirements:
- Processor: Dual-core processor (e.g., Intel Core i3 or AMD equivalent).
- RAM: 4 GB.
- Storage: 500 MB of free disk space for project files and dependencies.
- Display: 1280x720 resolution.
- Network: Internet connection for API calls (e.g., Google Custom Search API).
- Recommended Requirements:
- Processor: Quad-core processor (e.g., Intel Core i5 or AMD Ryzen 5).
- RAM: 8 GB or higher.
- Storage: 1 GB of free disk space for project files, dependencies, and logs.
- Display: 1920x1080 resolution or higher.
- Network: Stable internet connection for API calls and development dependencies.
- Additional Hardware:
- Development Machine: A laptop or desktop capable of running Node.js and React development environments.
- Optional: A secondary monitor for multitasking during development.
- These specifications ensure that the project can be developed, tested, and deployed efficiently.

SYSTEM DESIGN

SYSTEM ARCHITECTURE

The system architecture for the Crop Prediction System is based on a client-server model. It consists of the following components:

Frontend (Client):

Built using React and styled with Tailwind CSS.

Provides a user-friendly interface for interacting with the system.

Handles user input for crop prediction, task management, and search queries.

Communicates with the backend server via RESTful APIs.

Backend (Server):

Built using Node.js and Express.js.

Processes crop prediction requests and returns results.

Handles API requests for crop information and search queries.

Implements business logic for crop prediction.

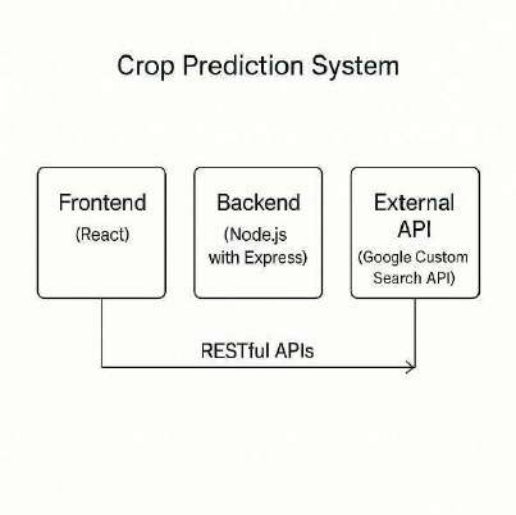
External API:

Integrates with the Google Custom Search API to fetch search results dynamically.

Database (Optional):

Currently, the system does not use a database. However, a database like MongoDB or MySQL can be integrated in the future to store user data, crop details, or task lists.

.



UML DIAGRAMS

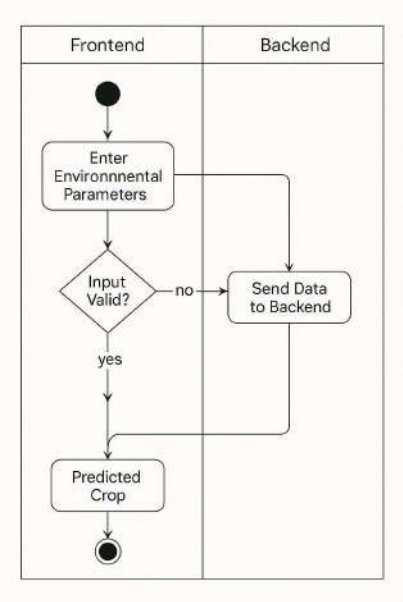
1. Use Case Diagram:

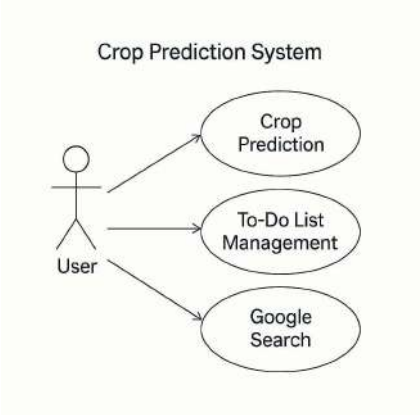
Actors: User (Farmer or Agricultural Enthusiast).
Use Cases:
Enter environmental parameters for crop prediction.
View crop prediction results.
Add or remove tasks in the to-do list.
Perform a Google search and view results.
Diagram:

Activity Diagram:

Steps:
User opens the application.
User enters data for crop prediction and submits the form.
Backend processes the data and returns the prediction.
User views the prediction and crop details.
User adds or removes tasks in the to-do list.
User performs a Google search and views results dynamically.
Sequence Diagram:

Interactions:
User submits crop prediction form.
Frontend sends a POST request to the backend.
Backend processes the request and returns the prediction.
Frontend displays the prediction and visualization.





MODULES

- Packet The system is divided into the following modules:
- Crop Prediction Module:
- Functionality:
- Accepts environmental parameters (e.g., nitrogen, phosphorus, etc.).

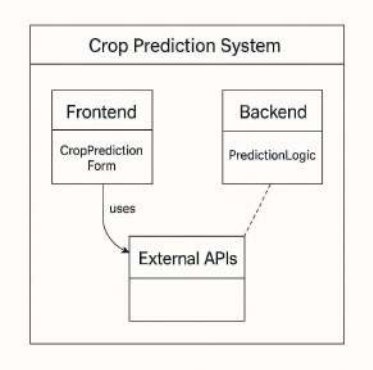
- Sends data to the backend for processing.
- Displays the predicted crop along with its details.
- Components:
- Form for user input.
- Visualization of input data.
- Crop information display.
- To-Do List Module:

- Functionality:
- Allows users to add, view, and remove tasks.
- Tasks are managed locally in the frontend state.
- Components:
- Input field for adding tasks.
- List of tasks with remove buttons.
- Search Module:

- Functionality:
- Allows users to perform Google searches.
- Displays search results dynamically below the search bar.
- Components:
- Search bar for entering queries.
- Results box for displaying search results.
- Backend Module:

- Functionality:
- Processes crop prediction requests.
- Handles API requests for crop information.
- Components:
- /predict endpoint for crop prediction.
- Integration with Google Custom Search API.
- Visualization Module:

- Functionality:
- Displays input data in a graphical format.
- Components:
- Graphs and charts using react-chartjs-2.
- This system design ensures modularity, scalability, and ease of maintenance.



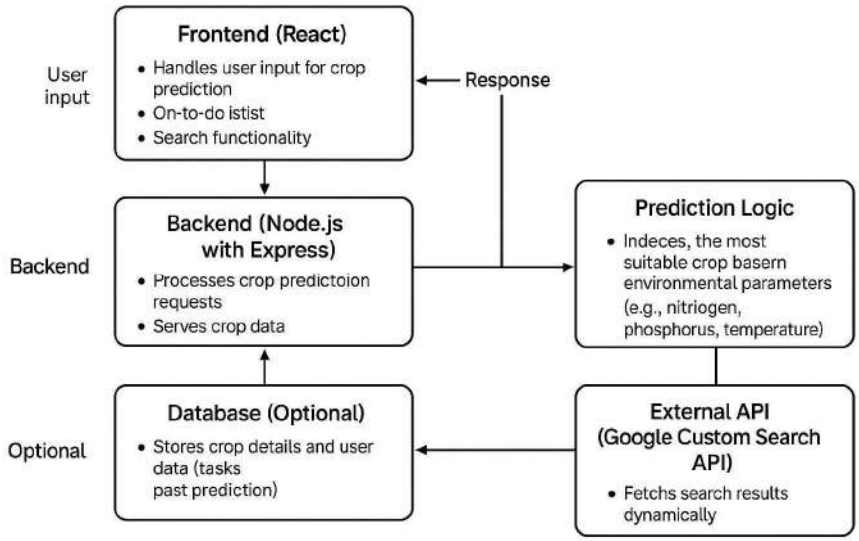
IMPLEMENTATION

INPUT DESIGN

Castpone's input begins with live or stored network traffic in PCAP format. This raw data is converted into structured flow records using CICFlowMeter, extracting features like duration, byte count, protocol, and IPs. The design ensures accurate and relevant data for effective machine learning classification.

OUTPUT DESIGN

The system outputs classification results (benign or malicious) with details like timestamps, IPs, and confidence scores. In real-time, it can trigger alerts and log suspicious flows, offering clear and actionable insights for administrators.



SAMPLE CODE

The Castpone system uses modular Python code built on pandas, numpy, scikit-learn, and joblib for model training and prediction. The process includes:

Data Preprocessing:

Cleaning the dataset by removing duplicates, nulls, zero-only columns, and categorical fields (e.g., "Timestamp"). This ensures compatibility with ML models and improves data quality.

Feature Scaling:

Applying normalization to input features and label encoding to output classes for consistent and unbiased model training.

Feature Selection:

Using Random Forest to rank and retain the most important features, reducing complexity and enhancing performance.

Model Training:

Training various classifiers—Decision Tree, Random Forest, MLP, Naive Bayes, and KNN—to detect bruteforce attacks. The best-performing model is serialized using joblib.

Prediction:

A separate script loads the trained model and classifies new flows in real-time or batch settings.

IMPLEMENTATION

The implementation phase executes the full intrusion detection pipeline—from preprocessing to prediction. The dataset, containing flow-based network features, is loaded and cleaned by removing irrelevant fields and handling missing values. Categorical labels are encoded for compatibility.

The data is split into training (70%) and testing (30%) sets. A Random Forest Classifier is trained to detect brute force patterns and evaluated using accuracy, precision, recall, and F1-score.

The trained model and scaler are saved using joblib for future predictions. An inference function processes new flows and predicts intrusions, enabling real-time or batch detection. This implementation confirms the system's effectiveness in identifying brute force attacks.

SOFTWARE TESTING

Software testing ensures the reliability and correctness of the brute force detection system. It verifies the end-toend functionality—from data preprocessing to prediction.

Unit Testing was conducted on individual modules like data loaders and preprocessing functions to ensure proper handling of nulls, label encoding, and scaling.

Integration Testing validated the seamless data flow between components, ensuring compatibility in formats and feature consistency.

Model Evaluation involved testing the Random Forest classifier using metrics like accuracy, precision, recall,

F1-score, and a confusion matrix to assess classification performance.

Performance Testing measured the system's ability to process large datasets efficiently, confirming suitability for real-time detection scenarios.

The system consistently produced accurate and stable results. Minor issues were optimized, confirming the model's robustness and deployment readiness.

Key Metrics Explained:

Confusion Matrix: Shows TP, TN, FP, FN for evaluating model predictions.

Accuracy: Ratio of correct predictions to total predictions.

Precision: Correct positive predictions over total predicted positives.

Recall: Correct positives over total actual positives.

F1-Score: Harmonic mean of precision and recall for balanced performance.

RESULT ANALYSIS

It demonstrated and evaluated multiple machine learning algorithms based on accuracy, speed, and suitability for real-time intrusion detection. Metrics such as Precision, Recall, F1-Score, Accuracy, training time, and prediction latency were used.

Training Time:

- MLP (Neural Network):
  Took the longest training time (~10 seconds).
  Suitable for static systems, not frequent retraining.
- Decision Tree, Random Forest, Naive Bayes, KNN:
  Required minimal training time (close to 0 seconds).
  Highly efficient for rapid model development and frequent updates. Prediction Time:
- K-Nearest Neighbors (KNN):
  Slowest in prediction (~0.14 seconds).
  Not ideal for real-time IDS.
- Decision Tree & Random Forest:
  Fastest predictions (<0.02 seconds).
  Well-suited for real-time detection.
- MLP & Naive Bayes:
  Moderate prediction speed, better than KNN. Performance Metrics:
- Decision Tree & Random Forest: Perfect scores (1.00) across all metrics.
- MLP: 0.87 for Precision, Recall, Accuracy; F1-score was 0.85.
- Naive Bayes: 0.88 for key metrics; Accuracy at 0.86.
- KNN: Precision 0.97, Recall 0.98, F1-score 0.98, Accuracy 0.97.
- Online Decision Tree: Consistent 0.89 across metrics.

Overall comparison:

Decision Tree and Random Forest models offer the best trade-off between accuracy and efficiency. KNN, though accurate, is slower in prediction. The results support using Decision Tree or Random Forest for scalable, real-time intrusion detection systems.

FUTURE SCOPE & CONCLUSION

FUTURE SCOPE

The current system excels in detecting brute-force attacks but has ample potential for expansion. Future improvements include real-time deployment for continuous traffic monitoring, multi-attack classification to detect diverse network attacks like DDoS and phishing, and integrating advanced deep learning models (CNNs and RNNs) for better pattern recognition and analysis. A cloud-based IDS would enable scalable, distributed monitoring across larger networks, while automated threat responses could alert administrators or take preventive actions in real time. Additionally, research can further explore advanced machine learning techniques, ensemble modeling, and real-time threat intelligence to enhance the adaptability and strength of IDS systems in response to evolving cyber threats.

CONCLUSION

This study on Intrusion Detection Systems (IDS) emphasizes the importance of machine learning (ML) in cybersecurity, specifically in detecting brute-force attacks. By utilizing the CSE-CIC-IDS-2018 dataset and a customized data collection environment, the research uncovered valuable insights into the efficacy of ML models for intrusion detection. Key findings include the identification of influential features that improve classification precision and reduce false positives and negatives. Among the evaluated models, Decision Tree and Random Forest exhibited exceptional accuracy and efficiency, confirming their suitability for real-time IDS deployment. Feature selection, especially using Random Forest, improved model performance and execution speed. This work lays the foundation for future research into advanced machine learning strategies and scalable IDS

solutions, which will be crucial for adapting to the ever-changing landscape of cyber threats.

BIBLIOGRAPHY

1. Performance analysis of intrusion detection for deep learning model based on CSE-CIC-IDS2018 dataset. Link
2. SSH-Brute Force Attack Detection Model based on Deep Learning. ResearchGate, 2023. Link
3. Aljanabi, M., Ismail, M.A., Ali, A.H. Intrusion detection systems, issues, challenges, and needs. Int. J. Comput. Intell. Syst., 2021.
4. Alzaqebah, A., Aljarah, I., Al-Kadi, O., Damaševičius, R. Modified Grey Wolf Optimization Algorithm for IDS. Mathematics, 2022.
5. Ambusaidi, M.A., He, X., Nanda, P., Tan, Z. Building an IDS using a filter-based feature selection algorithm. IEEE Trans. Comput., 2016.
6. Canadian Institute For Cybersecurity. CICFlowMeter-V4.0 for anomaly detection. Link
7. Chimphlee, S., Chimphlee, W. Machine learning to improve anomaly-based network intrusion detection. Indones. J. Electr. Eng. Comput. Sci., 2023.
8. Gautam, R.K.S., Doegar, E.A. An ensemble approach for IDS using machine learning algorithms. 2018 8th International Conference on Cloud Computing.
9. IDS 2018 Datasets, Canadian Institute for Cybersecurity. Link
10. Jaradat, A.S., Barhoush, M.M., Easa, R.B. Machine learning approach for network intrusion detection. Indones. J. Electr. Eng. Comput. Sci., 2022.
11. Kaja, N., Shaout, A., Ma, D. Intelligent intrusion detection system. Appl. Intell., 2019.
12. Karatas, G., Demir, O., Sahingoz, O.K. Improving IDS performance on an imbalanced dataset. IEEE Access, 2020.
13. Khan, M.A. HCRNNIDS: Hybrid convolutional recurrent neural network-based IDS. Processes, 2021.
14. Kim, J., Shin, Y., Choi, E. IDS based on a Convolutional Neural Network. J. Multimed. Inf. Syst., 2019.
15. Malliga, S., Nandhini, P.S., Kogilavani, S.V. Review of deep learning techniques for DoS attack detection. Inf. Technol. Control, 2022.
16. Momand, A., Jan, S.U., Ramzan, N. Survey of IDS using machine learning, deep learning, datasets, and attack taxonomy. J. Sens., 2023.
17. Muhsen, A.R., Jumaa, G.G., Bakri, N.F.A., Sadiq, A.T. Feature selection strategy for NIDS using Meerkat Clan Algorithm. Int. J. Interact. Mob. Technol., 2021.
18. Nassif, A.B., Talib, M.A., Nasir, Q., Dakalbab, F.M. Machine Learning for Anomaly Detection: A Systematic Review. IEEE Access, 2021.
19. patator | Kali Linux Tools. Link
20. Qusyairi, R., Saeful, F., Kalamullah, R. Ensemble learning and feature selection for improved IDS performance. IAICT, 2020.
21. Songma, S., Sathuphan, T., & Pamutha, T. Optimizing IDS on the CSE-CIC-IDS-2018 dataset. Computers, 2023. DOI