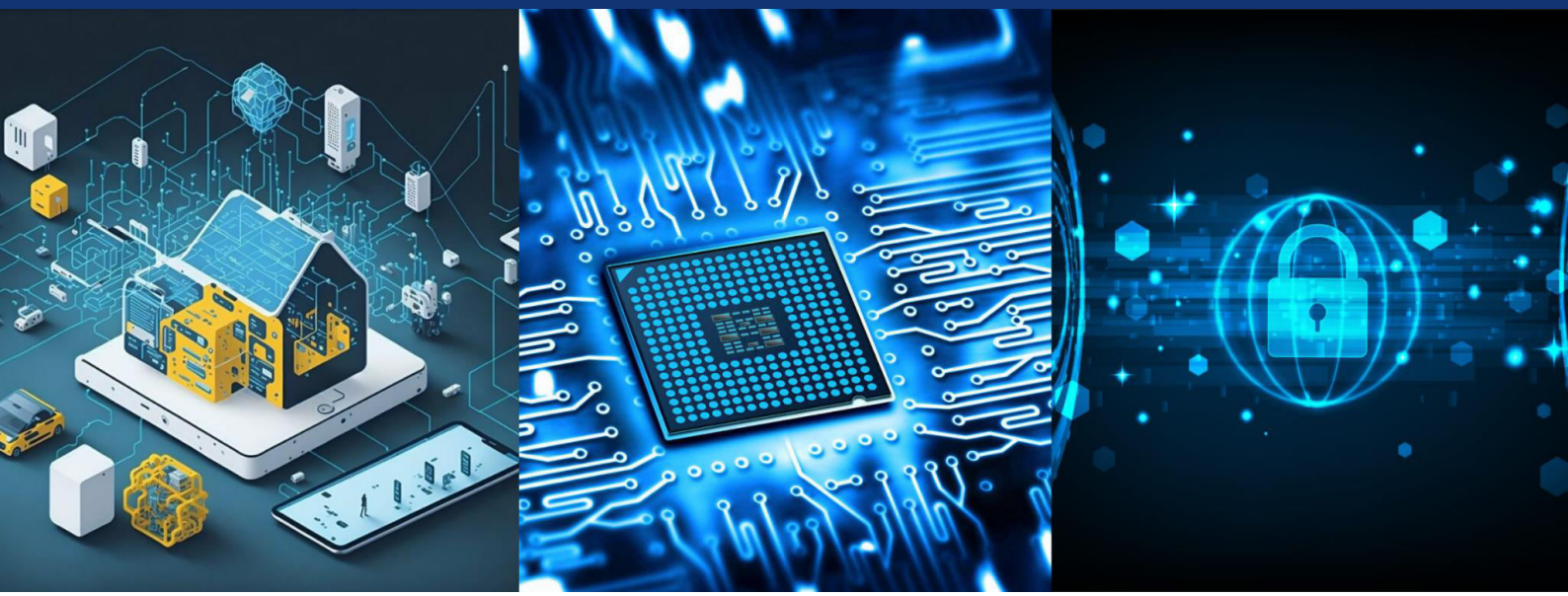
IJIRCCE

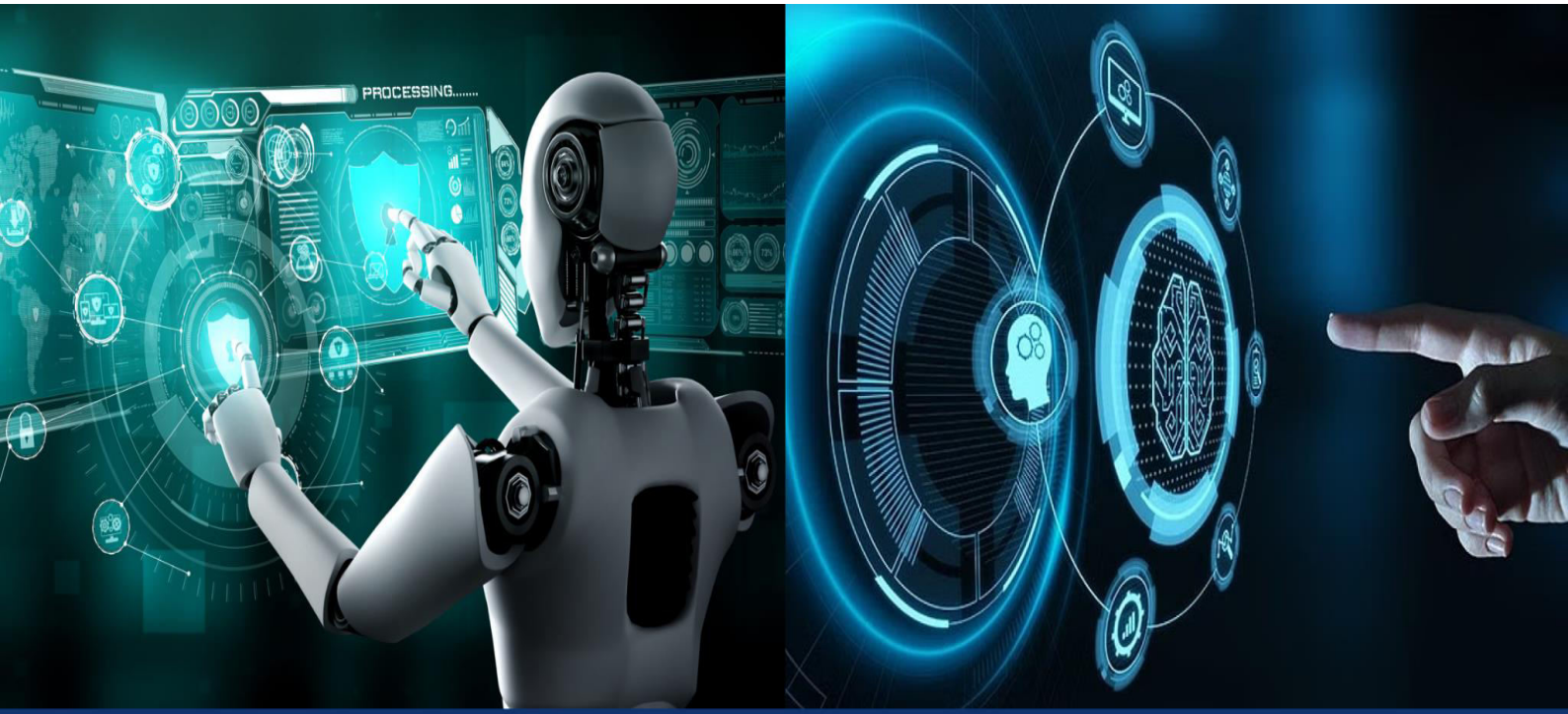
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.927

Volume 14, Issue 6, June 2026

© 2026 IJIRCCE | Volume 14, Issue 6, June 2026 DOI: 10.15680/IJIRCCE.2026.1406114

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Invisible Configuration Conflicts Detector Using Machine Learning a Random Forest–Based Intelligent Framework for Software Configuration Conflict Detection

Dr. T.V.S Sriram[1], Mrs. l.t. Priyanka[2], CH. Anusha[3]

Sr. Asst. Professor, Dept. of MCA, Nadimpalli Satyanarayana Raju Institute of Technology, Visakhapatnam, AP, India[1]

Asst. Professor, Dept. of MCA, Nadimpalli Satyanarayana Raju Institute of Technology, Visakhapatnam, AP, India[2]

Dept. of MCA, Nadimpalli Satyanarayana Raju Institute of Technology, Visakhapatnam, AP, India[3]

ABSTRACT: Managing software configurations has become increasingly challenging as modern systems contain numerous interconnected settings. Incompatible configuration values can create hidden conflicts that negatively affect system performance, reliability, and stability. Identifying these conflicts manually is often difficult because they may not produce immediate errors. This project introduces an Invisible Configuration Conflicts Detector Using Machine Learning, which predicts potential configuration conflicts before they impact the system. The proposed solution employs the Random Forest algorithm to analyze configuration data and classify configurations as either safe or conflicting. A web-based application is developed using Flask, HTML, CSS, and SQLite to provide secure user access and an easy-to-use prediction interface. By automating the conflict detection process, the system minimizes manual effort, improves prediction efficiency, and supports better configuration management. The proposed approach demonstrates how machine learning can assist in maintaining stable and reliable software systems through intelligent conflict prediction.

KEYWORDS: Machine Learning, Configuration Management, Configuration Conflict Detection, Random Forest, Flask, SQLite, Predictive Analysis, Software Systems.

I. INTRODUCTION

Modern software systems depend on numerous configuration settings to function correctly. Incorrect or incompatible configurations can lead to hidden conflicts that affect system performance, reliability, and security. Since these conflicts are not always visible, identifying them using manual methods becomes difficult and time-consuming.

Traditional configuration management techniques mainly rely on predefined rules and human monitoring, which may fail to detect complex or invisible configuration issues. Machine learning provides a smarter approach by learning patterns from configuration data and predicting potential conflicts automatically.

This project, **"Invisible Configuration Conflicts Detector Using Machine Learning,"** uses the **Random Forest** algorithm to classify configurations as safe or conflicting. The system is developed using **Flask**, **HTML**, **CSS**, and **SQLite**, providing a simple web-based platform for user registration, configuration input, and prediction. The proposed solution helps reduce manual effort, improve prediction accuracy, and support effective configuration management.

II. LITERATURE REVIEW

Configuration management has become an important area of research due to the increasing complexity of modern software systems. Several studies have focused on detecting configuration errors using rule-based methods, system monitoring, and log analysis. Although these approaches help identify common issues, they often fail to detect hidden configuration conflicts.

© 2026 IJIRCCE | Volume 14, Issue 6, June 2026 DOI: 10.15680/IJIRCCE.2026.1406114

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Recent research has shown that **machine learning** techniques can improve conflict detection by analyzing configuration data and identifying patterns that are difficult to recognize manually. Algorithms such as **Decision Tree**, **Random Forest**, and **Support Vector Machine (SVM)** have been successfully applied to classification and prediction problems.

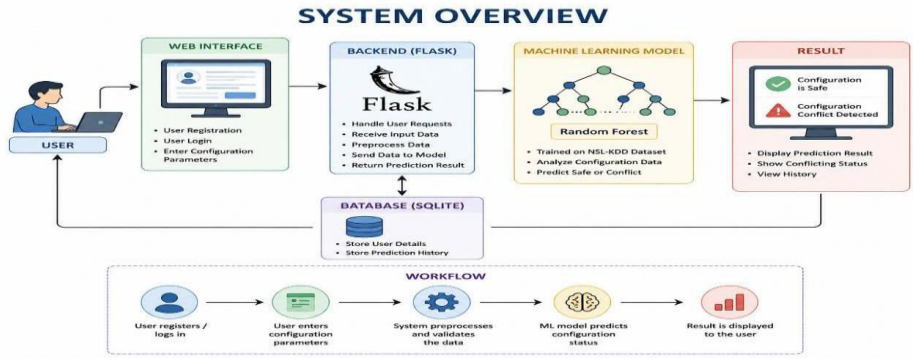
Many existing systems still require significant manual intervention and are limited in handling dynamic software environments. To overcome these limitations, the proposed project uses the **Random Forest** algorithm to automatically predict invisible configuration conflicts. This approach improves prediction accuracy, reduces manual effort, and provides a more efficient solution for configuration management.

III. SYSTEM OVERVIEW

The **Invisible Configuration Conflicts Detector Using Machine Learning** is a web-based application developed to identify hidden configuration conflicts automatically. The system collects configuration-related inputs from users, processes the data, and predicts whether the configuration is safe or contains potential conflicts using the **Random Forest** algorithm.

The application includes modules for user registration, login, configuration input, prediction, and result display. It is developed using **Flask** for backend processing, **HTML** and **CSS** for the user interface, and **SQLite** for database management. The trained machine learning model analyzes the input data and generates accurate prediction results through a simple and user-friendly interface.

The proposed system reduces manual effort, improves prediction accuracy, and helps users identify configuration issues before they affect system performance.



IV. METHODOLOGY

The proposed system follows a systematic approach to detect invisible configuration conflicts using machine learning. Initially, configuration data is collected and preprocessed by removing unnecessary information and converting the data into a suitable format. Important features are then selected to improve the prediction process.

The processed data is used to train the **Random Forest** algorithm, which learns the relationship between different configuration parameters and their corresponding outcomes. When a user enters configuration details through the web application, the trained model analyzes the input and predicts whether the configuration is **Safe** or **Conflict Detected**. Finally, the prediction result is displayed to the user through an interactive interface.

Methodology Steps

- Data Collection : Configuration data is collected from the **NSL-KDD dataset** for training and testing the machine learning model.
- Data Preprocessing : Configuration data is collected from the **NSL-KDD dataset** for training and testing the machine learning mode

© 2026 IJIRCCE | Volume 14, Issue 6, June 2026 DOI: 10.15680/IJIRCCE.2026.1406114

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- Feature Selection : *I*mportant configuration features are selected to improve the performance of the prediction model
- Model Training using Random Forest : The Random Forest algorithm is trained using the processed dataset to identify configuration conflicts.
- Configuration Prediction : The trained model analyzes user input and predicts whether the configuration is **Safe** or **Conflict Detected**.
- Result Display : The prediction result is displayed to the user through a simple and interactive web interface.

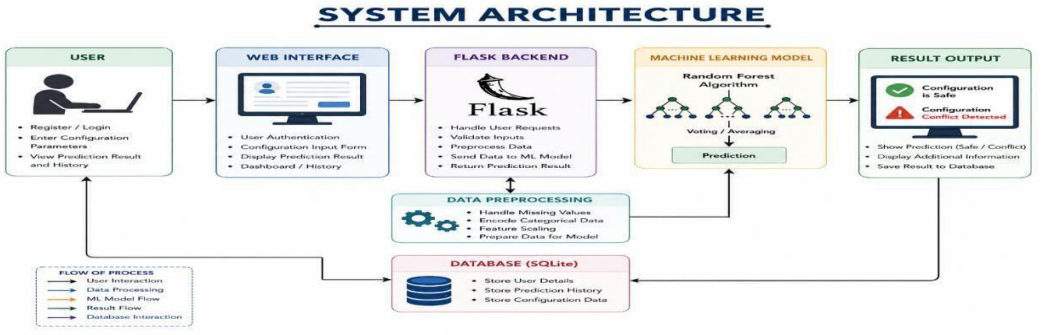
.

V. SYSTEM ARCHITECTURE

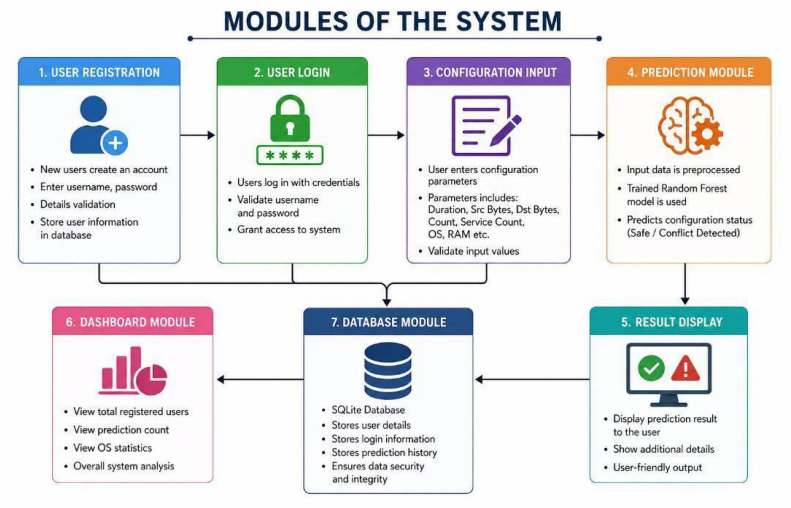
The proposed system follows a simple architecture that integrates user interaction, data processing, machine learning, and result generation. The user first registers and logs into the application before entering configuration details. The backend validates and preprocesses the input data, which is then passed to the trained **Random Forest** model. The model analyzes the configuration and predicts whether it is **Safe** or **Conflict Detected**. Finally, the prediction result is displayed to the user, while user information is securely stored in the SQLite database.

Architecture Components

- User Interface (Login & Configuration Input)
- Flask Backend
- Data Preprocessing
- Random Forest Model
- SQLite Database
- Prediction Result Display



VI. MODULES



© 2026 IJIRCCE | Volume 14, Issue 6, June 2026 DOI: 10.15680/IJIRCCE.2026.1406114

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

The proposed system consists of the following modules:

6.1 User Registration Module

This module allows new users to create an account by providing a username and password. The entered details are validated and securely stored in the SQLite database.

6.2 User Login Module

Registered users can log in using their credentials. The system authenticates the user before granting access to the prediction module.

6.3 Configuration Input Module

Users enter configuration-related parameters such as duration, source bytes, destination bytes, count, service count, operating system, and RAM. The input data is validated before processing.

6.4 Prediction Module

The validated input is processed by the trained **Random Forest** machine learning model, which predicts whether the configuration is **Safe** or **Conflict Detected**.

6.5 Result Display Module

The prediction result is displayed in a clear and user-friendly format, allowing users to understand the configuration status instantly.

6.6 Dashboard Module

The dashboard provides information such as the total number of users and operating system usage statistics, helping monitor application activity.

6.7 Database Module

The SQLite database stores user credentials, login information, and prediction-related data, ensuring secure and efficient data management.

VII. IMPLEMENTATION DETAILS

7.1 Technology Stack

The proposed system is developed using **Python** and the **Flask** framework for backend development. The user interface is designed using **HTML**, **CSS**, and **JavaScript**, while **SQLite** is used for database management. The **Random Forest** algorithm is implemented using the **Scikit-learn** library for configuration conflict prediction.

7.2 Frontend Implementation

The frontend provides a simple and responsive interface for user registration, login, configuration input, prediction results, and dashboard visualization. It is designed to ensure easy interaction and a better user experience.

7.3 Backend Implementation

The backend is developed using Flask, which manages user authentication, input validation, data preprocessing, communication with the machine learning model, and result generation.

7.4 Database Design

The SQLite database stores user credentials, login information, and prediction history. It enables secure storage and efficient retrieval of application data.

7.5 Machine Learning Model

The **Random Forest** algorithm is trained using the **NSL-KDD dataset** after applying preprocessing and feature selection techniques. The trained model predicts whether a given configuration is **Safe** or **Conflict Detected**.

7.6 Security Measures

The application provides secure login authentication, input validation, session management, and protected database access to ensure user data confidentiality and system reliability..

© 2026 IJIRCCE | Volume 14, Issue 6, June 2026 DOI: 10.15680/IJIRCCE.2026.1406114

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VIII. EXPERIMENTAL DETAILS AND RESULTS

8.1 Experimental Setup

The proposed system was developed using **Python**, **Flask**, **HTML**, **CSS**, and **SQLite**. The **Random Forest** machine learning model was trained using the **NSL-KDD dataset**. The application was tested on a Windows operating system with standard hardware configuration to evaluate its functionality, prediction capability, and overall performance.

8.2 Test Results

Test Case	Description	Result Status
User Registration	Create a new user account	Passed ✓
User Login	Login with valid credentials	Passed ✓
Configuration Input	Enter valid configuration values	Passed ✓
Conflict Prediction	Predict configuration status	Passed ✓
Result Display	Display prediction result	Passed ✓
Dashboard	Display user statistics	Passed ✓

8.3 Performance Evaluation

The Random Forest model successfully classified configuration data with good prediction accuracy. The system generated prediction results within a short response time and handled user requests efficiently. The web application performed smoothly during testing and provided reliable results for different configuration inputs.

8.4 Observations

The developed system successfully detected potential configuration conflicts based on the user-provided input. The prediction results were accurate and generated quickly through the web interface. User registration, login, dashboard, and database operations functioned correctly throughout the testing process. The proposed system reduced manual effort and improved the efficiency of configuration conflict detection.

IX. APPLICATIONS

The proposed **Invisible Configuration Conflicts Detector Using Machine Learning** can be applied in various software and IT environments, including:

- **Software Configuration Management:** Detects hidden configuration conflicts before they affect system performance.
- **Cloud Computing:** Assists in monitoring and validating cloud server configurations.
- **IT Infrastructure Management:** Helps administrators identify configuration issues in enterprise systems.
- **Network Administration:** Supports the analysis of network-related configuration settings to prevent failures.
- **Data Centers:** Improves the reliability and stability of server configurations.
- **System Maintenance:** Reduces manual effort by automating configuration conflict detection.
- **Research and Education:** Serves as a learning tool for machine learning and configuration management concepts.
- **Decision Support:** Provides quick and accurate predictions to assist administrators in making better configuration decisions.

X. LIMITATIONS

The proposed system has a few limitations that can be improved in future versions:

- The prediction accuracy depends on the quality of the training dataset.
- The system supports only predefined configuration parameters.
- It cannot perform real-time monitoring of system configurations.
- The model may require retraining when new configuration patterns are introduced.
- The application is designed for a limited number of users and may require optimization for large-scale environments.
- The current system uses a single machine learning algorithm, which may not handle all complex configuration scenarios

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

XI. FUTURE ENHANCEMENTS

The proposed system can be enhanced further to improve its functionality and performance. Future improvements include:

- Implementing advanced machine learning and deep learning algorithms to improve prediction accuracy.
- Supporting real-time monitoring of system configurations.
- Integrating cloud platforms for large-scale configuration management.
- Adding graphical dashboards and detailed analytics for better visualization.
- Expanding the system to support a wider range of configuration parameters.
- Developing a mobile-friendly version for easy access across different devices.
- Implementing automatic conflict resolution suggestions based on prediction results.
- Enhancing system security with advanced authentication and data protection techniques.

XII. CONCLUSION

The **Invisible Configuration Conflicts Detector Using Machine Learning** provides an intelligent solution for identifying hidden configuration conflicts in software systems. By utilizing the **Random Forest** algorithm, the system accurately predicts whether a given configuration is safe or contains potential conflicts. The web-based application, developed using **Flask**, **HTML**, **CSS**, and **SQLite**, offers a simple and user-friendly interface for configuration analysis.

The proposed system reduces manual effort, improves prediction efficiency, and supports better configuration management. Experimental results demonstrate that the application performs reliably and generates accurate prediction results. Overall, the project highlights the practical application of machine learning in software configuration management and provides a scalable foundation for future enhancements.

REFERENCES

[1] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[4] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Burlington, MA, USA: Morgan Kaufmann, 2017.
[5] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York, NY, USA: Springer, 2009.
[7] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2012.
[8] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.
[9] F. Pedregosa *et al*., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[10] W. McKinney, *Python for Data Analysis*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.
[11] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2002.
[12] R. S. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2019.
[13] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson, 2016.
[14] M. Grinberg, *Flask Web Development*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018.
[15] D. Beazley and B. K. Jones, *Python Cookbook*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2013.
[16] SQLite Consortium, "SQLite Documentation," 2025.
[17] Python Software Foundation, "Python Documentation," 2025.
[18] Pallets Project, "Flask Documentation," 2025.
[19] Scikit-learn Developers, "Scikit-learn User Guide," 2025.
[20] NSL-KDD Dataset, "NSL-KDD: Network Intrusion Detection Dataset," University of New Brunswick.
[21] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

[22] K. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
[23] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, 2nd ed. Springer, 2021.
[24] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2013.
[25] N. V. Chawla, "Data Mining for Intelligent Systems," *IEEE Computer*, vol. 44, no. 3, pp. 88–91, 2011.
[26] IEEE Computer Society, "Software Configuration Management Best Practices," 2023.
[27] National Institute of Standards and Technology (NIST), *Guide to Configuration Management*, 2022.
[28] OWASP Foundation, "OWASP Secure Coding Practices," 2024.
[29] Oracle Corporation, "Java Documentation," 2025.
[30] Mozilla Foundation, "HTML5 Documentation," 2025.
[31] World Wide Web Consortium (W3C), "CSS Specifications," 2025.
[32] ECMA International, "JavaScript Language Specification," 2024.
[33] P. Harrington, *Machine Learning in Action*. Manning Publications, 2012.
[34] A. Burkov, *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.
[35] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., 2023.
[36] Google Developers, "Machine Learning Crash Course," 2025.
[37] Microsoft Learn, "Introduction to Machine Learning," 2025.
[38] IBM, "Machine Learning Documentation," 2025.
[39] Oracle, "Database Concepts," 2025.
[40] IEEE Xplore Digital Library, "Research Articles on Machine Learning and Configuration Management," 2025.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details