ISSN 2723-6471

Enhanced Fall Detection using Optimized Random Forest Classifier on Wearable Sensor Data

Lasmedi Afuan[1,*], R. Rizal Isnanto[2]

[1]*Department of Informatics, Engineering Faculty, Universitas Jenderal Soedirman, Indonesia*

[2]*Department of Computer Engineering, Diponegoro University, Indonesia*

(Received: September 22, 2024; Revised: October 31, 2024; Accepted: November 7, 2024; Available online: December 28, 2024)

Abstract

This study aims to enhance the performance of fall detection systems for elderly care using wearable sensors by optimizing the Random Forest (RF) algorithm. Falls among the elderly are a major health risk, and timely detection can mitigate serious injuries or fatalities. The primary contributions of this research include developing an optimized RF model specifically tailored for real-time fall detection on resource-constrained devices such as smartwatches. Our approach involves feature engineering, hyperparameter tuning using Grid Search and Randomized Search, and model evaluation to achieve optimal performance. Key findings indicate that the optimized RF model achieved an accuracy of 92%, precision of 91%, recall of 89%, and an F1-score of 90%, with an average processing time of 0.045 seconds per prediction. These metrics underscore the model's capability for real-time deployment, demonstrating improved computational efficiency and predictive accuracy compared to traditional machine learning algorithms and deep learning models. The novelty of this study lies in its targeted optimization of the RF model to balance accuracy with low computational demand, addressing the limitations of existing methods that are either computationally intensive or prone to misclassification. This research provides a scalable solution for continuous fall monitoring, with significant implications for wearable healthcare technology, improving both accessibility and response times in elderly care.

Keywords: Random Forest Optimization, Wearable Sensors, Feature Engineering, Hyperparameter Tuning, Elderly Care Technology, Real-Time Monitoring, Computational Efficiency, Healthcare Applications

1. Introduction

As the global population ages, the occurrence of falls has become one of the leading causes of injury, particularly among elderly individuals. According to the World Health Organization (WHO), falls are the foremost cause of injury-related deaths in adults aged 65 years and older [1], [2]. In response, automatic fall detection systems have emerged as a crucial area of research in healthcare, aiming to reduce the consequences of these incidents through timely intervention. Falls often result in severe physical injuries, psychological effects, and extended hospital stays, further burdening healthcare systems and impacting the quality of life of individuals.

In recent years, wearable devices such as smartwatches and fitness trackers, equipped with sensors like accelerometers and gyroscopes, have become a practical and efficient solution for detecting falls in real-time. These devices can continuously monitor physical activity and provide data to machine learning models that aim to identify falls accurately. However, the challenge lies in distinguishing between fall events and other daily activities that may exhibit similar sensor readings, such as sitting down abruptly or jumping. Thus, developing models that can differentiate falls from normal activities with high accuracy and low computational demand is essential.

Fall detection among elderly individuals has become increasingly important due to the high risk of injury or fatality from falls. Automatic fall detection systems have emerged as a vital area of research, particularly for healthcare applications that rely on timely intervention to mitigate injury severity and improve patient outcomes. Wearable devices equipped with motion sensors, such as accelerometers and gyroscopes, offer a practical solution for continuous, real-time monitoring of falls. However, achieving high accuracy in distinguishing fall events from daily activities remains

***Corresponding author: Lasmedi Afuan (lasmedi.afuan@unsoed.ac.id)**
DOI: https://doi.org/10.47738/jads.v6i1.498

challenging due to similarities in sensor readings for various movements. Additionally, computational efficiency is essential for battery-operated wearable devices, which require quick, low-power processing to support prolonged usage. This study addresses these challenges through a structured methodology comprising three main stages: (1) Feature Engineering, where significant features are extracted and selected from raw sensor data to enhance model accuracy; (2) Hyperparameter Tuning, where critical parameters of the Random Forest model are optimized to improve predictive performance while maintaining efficiency; and (3) Performance Evaluation, where the model's effectiveness is assessed using key metrics such as accuracy, precision, recall, and processing time. This structured approach aims to achieve a balance between accuracy and computational demands, making the model suitable for real-time fall detection applications on wearable devices. The following sections will discuss each of these methodological stages in detail, providing insights into the techniques used and their contributions to the study's objectives.

The motivation for this research stems from the need to address the limitations of traditional fall detection algorithms—primarily their lack of accuracy and computational efficiency. While previous studies have explored machine learning approaches, including SVM [3], [4], K-Nearest Neighbors (KNN) [5], [6], [7], and Decision Trees, these methods are often sensitive to noise and irrelevant features, leading to reduced performance. On the other hand, deep learning techniques like CNNs and LSTM networks provide better accuracy but are unsuitable for real-time deployment on wearable devices due to their high computational demands [8], [9].

Therefore, this research aims to enhance the Random Forest algorithm for fall detection, an ensemble method known for its robustness and resistance to overfitting, by optimizing its parameters and refining the feature selection process. The goal is to increase the algorithm's accuracy while maintaining computational efficiency, thus making it viable for deployment on wearable devices.

Despite the effectiveness of Random Forest models, existing implementations often rely on default parameter settings and broad feature sets, which can reduce their overall accuracy and lead to longer processing times. Few studies have explored the systematic optimization of these models for fall detection. This research seeks to address this gap by implementing feature engineering and hyperparameter tuning to improve model performance, with a particular focus on wearable sensor data. The study addresses the following research questions: (1) How can the performance of Random Forest in fall detection be improved through feature engineering and hyperparameter tuning? (2) What are the trade-offs between accuracy and processing time in an optimized Random Forest model for fall detection? (3) How does the optimized Random Forest model compare to traditional machine learning algorithms and deep learning models in terms of performance metrics?

The significance of this research lies in its potential to improve fall detection systems that rely on wearable devices, offering both higher accuracy and reduced computational requirements. Accurate fall detection can significantly enhance the quality of life for elderly individuals by allowing caregivers or medical professionals to respond swiftly to potential emergencies. Furthermore, optimizing the Random Forest algorithm makes the system more suitable for real-time application on devices with limited computational resources, thereby ensuring broader accessibility and usability. To address these challenges, this paper proposes an optimized Random Forest model with several key innovations. First, feature engineering is employed to extract and refine key features from the raw sensor data, thereby enhancing the model's ability to distinguish between fall and non-fall events. Second, hyperparameter optimization is performed by fine-tuning Random Forest parameters, such as the number of trees and the depth of each tree, to improve predictive accuracy while maintaining processing efficiency. Finally, the model undergoes a comprehensive evaluation using a set of performance metrics, including accuracy, precision, recall, F1-score, and processing time, ensuring its suitability for real-time fall detection in wearable devices.

This study employs a research methodology that emphasizes the optimization of the Random Forest algorithm through feature selection and hyperparameter tuning. The dataset, collected from wearable devices, includes acceleration and gyroscope sensor data from various physical activities. The research process involves multiple stages. First, data preprocessing is performed, which includes cleaning and normalizing the sensor data to prepare it for analysis. Next, feature engineering is conducted by extracting and selecting the most relevant features to enhance model performance. Following this, the optimized Random Forest model is applied during the model training and validation phase, with cross-validation used to ensure that the model generalizes well to unseen data. Finally, performance evaluation is

conducted by comparing the results of the optimized model against traditional machine learning methods and deep learning models using multiple metrics.

The expected outcome of this research is an optimized Random Forest model that demonstrates improved performance in detecting falls, with a balance between accuracy and computational efficiency. We anticipate that the proposed model will significantly reduce false positives and false negatives while maintaining a low processing time, making it suitable for real-time applications in wearable devices. Furthermore, the study aims to contribute to the literature by providing a clear comparison of Random Forest with other fall detection techniques, thus informing future research and development in this field.

2. Related Work

Several approaches have been explored in the domain of fall detection, each with its own strengths and weaknesses. Threshold-based methods have been among the earliest and simplest approaches, relying on fixed sensor thresholds (such as acceleration or angular velocity) to detect falls. While these methods are computationally efficient and easy to implement, they often suffer from false positives, as they cannot effectively differentiate between falls and vigorous activities like running or jumping [10] noted that while threshold-based systems are popular for their simplicity, their lack of adaptability to varying environments and activities limits their practical use [11] further demonstrated that while tweaking thresholds can improve performance, these systems are inherently rigid, unable to accommodate the nuances of different physical actions or soft falls.

Traditional machine learning classifiers, such as Support Vector Machines (SVMs) and Decision Trees, have demonstrated improved performance over threshold-based methods [3], [4], [12], [13], [14]. SVMs can achieve reliable accuracy, especially in controlled environments [15], [16]. However, their performance often degrades in real-world settings where the variability of sensor data is high. Additionally, SVMs are sensitive to high-dimensional data, which often results in overfitting. Similarly, [17] utilized Decision Trees for fall detection, but found that they tend to create overly complex trees that capture noise, reducing their generalizability across different datasets.

Recent advancements in deep learning have led to the use of models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for fall detection. These models excel at capturing intricate patterns in time-series sensor data, which is crucial for distinguishing falls from normal activities. For instance, [18] demonstrated that CNNs could achieve high accuracy in fall detection by learning spatial relationships in sensor data. However, these models come with a significant computational cost, limiting their practicality for real-time applications on wearable devices with limited processing power [19] also employed LSTM networks, which proved highly effective in identifying falls by leveraging temporal patterns. Still, like CNNs, LSTMs are resource-intensive, making them less suitable for deployment in low-power environments.

Among the ensemble learning methods, Random Forest has shown promise for its ability to handle high-dimensional data and provide robust predictions [20] demonstrated that Random Forest is effective at mitigating the noise and variability in sensor data, outperforming many traditional classifiers. However, a common challenge with Random Forest is the lack of optimization in terms of feature selection and hyperparameter tuning, which can limit its overall performance [20] explored Random Forest for fall detection and found that while the algorithm could deliver competitive accuracy, it required further optimization to reduce false negatives.

Despite the relative success of Random Forest in fall detection, most studies have relied on default settings or limited optimization efforts, leaving significant room for improvement. This research seeks to address this gap by employing advanced feature engineering and comprehensive hyperparameter tuning to enhance the performance of Random Forest. By focusing on these optimizations, this study aims to improve not only the accuracy of fall detection but also its computational efficiency, making it more practical for real-time applications in wearable technology.

3. Methodology

The methodology of this study follows a structured process designed to evaluate and optimize the performance of the Random Forest algorithm for fall detection using sensor data obtained from wearable devices. The main steps

include data collection, data preprocessing, feature extraction, model selection, hyperparameter optimization, and evaluation metrics.

3.1. Dataset

The dataset used in this study was sourced from the Smartphone Human Fall Dataset available on Kaggle. It contains sensor data from wearable devices that record the acceleration and angular velocity of participants during various activities. These activities include both falls and non-fall events (such as walking or sitting). The dataset comprises 1,428 training samples and 573 testing samples, each labeled as either a fall ("1") or non-fall ("0"). Participant demographics include a mix of ages, genders, and activity levels, providing a varied dataset for realistic testing. Data was collected in controlled environments where participants performed a range of activities under supervised conditions, ensuring consistency and reliability. The key features used in the model include various metrics derived from sensor data, each contributing to the accurate detection of fall events. The feature acc_max represents the maximum acceleration measured by the sensor, while gyro_max captures the maximum angular velocity. Both of these features are crucial in detecting sudden changes associated with falls. In addition, acc_kurtosis and gyro_kurtosis provide insights into the distribution characteristics of acceleration and angular velocity, respectively, by measuring the kurtosis of these values. Similarly, acc_skewness and gyro_skewness measure the asymmetry of the acceleration and angular velocity distributions, adding important information that helps in distinguishing normal activities from fall events. The features post_gyro_max and post_lin_max capture the angular velocity and linear acceleration recorded immediately after the fall event, further enhancing the model's ability to classify fall incidents. Finally, the label feature indicates whether the recorded event is a fall (denoted by 1) or not (denoted by 0). These features collectively provide a robust set of data inputs for optimizing the accuracy of the fall detection model.

3.2. Data Preprocessing

The raw data from the dataset required several preprocessing steps before being used for model training. The first step was data cleaning, during which noise, duplicate entries, and missing values were removed. This step was crucial to ensure the quality of the input data and minimize the risk of biased outcomes. Next, min-max normalization was applied to all sensor readings to ensure they were on a comparable scale. This normalization was particularly important because algorithms like Random Forest can be sensitive to variations in feature scales, potentially impacting model performance. Finally, the dataset was split into training and testing subsets, with 1,428 samples designated for training and 573 samples reserved for testing. The training data was used to fit the model, while the testing data was employed to evaluate its performance and generalizability.

3.3.Feature Engineering

Feature engineering plays a crucial role in improving model performance by selecting the most relevant features that contribute to fall detection. Key features include acc_max (maximum acceleration) and gyro_max (maximum angular velocity), which are essential for identifying sudden movements characteristic of falls. The feature post_gyro_max, representing post-fall angular velocity, helps differentiate fall incidents from similar non-fall movements. This careful selection of features reduces noise and enhances the model’s ability to detect falls accurately.

3.4. Algorithm Selection

The study focused on the Random Forest classifier, an ensemble method that builds multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression). Random Forest was chosen due to its ability to handle high-dimensional data and its robustness against overfitting.

3.5. Hyperparameter Optimization

Random Forest’s performance was enhanced through hyperparameter tuning. Hyperparameter tuning was conducted to improve the performance of the Random Forest model by optimizing parameters such as the number of trees, maximum depth, and minimum samples for node splitting. Increasing the number of trees (n_estimators) enhances the model’s stability by reducing variance, while controlling the maximum depth prevents overfitting by limiting the model’s complexity. Tuning these parameters using Grid Search and Randomized Search allowed the model to capture relevant data patterns more effectively, balancing high accuracy with computational efficiency essential for real-time applications on wearable devices. The following parameters were optimized using Grid Search and Randomized Search

with cross-validation to enhance model performance. The number of trees (n_estimators) in the forest was tuned to ensure that the model could capture sufficient variance without becoming overly complex. The maximum depth (max_depth) of each tree was optimized to control the growth of the trees and prevent overfitting. Additionally, the minimum number of samples required to split a node (min_samples_split) was adjusted to ensure that splits were made only when statistically significant, thus maintaining model stability. The maximum number of features considered when determining the best split (max_features) was also optimized to balance model accuracy and computation time. To ensure robustness and prevent overfitting, each model was trained using a 10-fold cross-validation approach, which provided a reliable estimate of model performance.

3.6. Model Training and Validation

Once the hyperparameters were tuned, the Random Forest model was trained using the preprocessed training data. The training process involved fitting the model to the sensor data while utilizing cross-validation to assess its performance on unseen subsets of the data. This ensured that the model did not overfit the training data.

3.7. The Performance of the Random Forest Classifier was Evaluated Using the Following Metrics:

The evaluation of the model's performance relied on several key metrics. Accuracy was used to determine the proportion of correctly predicted fall events out of all predictions, providing an overall measure of how well the model performed. Precision was calculated as the ratio of true positive fall detections to the total positive predictions, thereby indicating how many of the detected falls were actual falls. Recall, on the other hand, represented the ratio of true positives to the actual number of falls, reflecting the model's ability to detect fall events accurately. The F1-score, which is the harmonic mean of precision and recall, provided a balanced assessment of the model's accuracy, especially in the presence of class imbalances. Lastly, processing time was measured to determine how quickly the model could be trained and make predictions. This metric is particularly important for real-time fall detection systems implemented in wearable devices, where rapid response is essential.

4. Results and Discussion

4.1. Data Preprocessing Results

Before model training, the data preprocessing stage involved cleaning, normalizing, and splitting the data. The dataset was split into 1,428 samples for training and 573 samples for testing. All feature values, such as acc_max and gyro_max, were normalized between 0 and 1 to ensure uniformity across the dataset, which is crucial for consistent model performance. A sample of the normalized data after preprocessing is presented in table 1. Additionally, no missing values were detected in the dataset, which eliminated the need for imputation.

Table 1. Sample of Normalized Data After Preprocessing

Sample	acc_max	gyro_max	acc_kurtosis	gyro_skewness	fall_label
1	0.745	0.512	0.278	0.610	1
2	0.415	0.830	0.102	0.390	0
3	0.556	0.672	0.340	0.785	0
4	0.851	0.490	0.295	0.560	1

In table 1, we can observe the distribution of normalized features for fall and non-fall events, demonstrating the need for accurate separation using classification algorithms.

4.2. Feature Engineering and Dimensionality Reduction

The correlation analysis conducted in this study identified specific features that played a crucial role in enhancing model accuracy for fall detection. Key features such as acc_max (maximum acceleration) and gyro_max (maximum angular velocity) were found to be highly correlated with fall events and provided valuable insights into the distinction between falls and non-fall activities. The feature acc_max captures sudden changes in acceleration, which are often indicative of a fall. High values of acc_max are typically associated with abrupt impacts or downward movements characteristic of falls. Including this feature improved the model's sensitivity to detecting high-magnitude acceleration

 ISSN 2723-6471

events, thereby enhancing recall for true fall instances. Similarly, gyro_max measures rapid changes in angular velocity, such as twisting or rotational movements, which often occur during a fall. High gyro_max values helped the model differentiate between typical daily movements and falls, as falls generally involve a combination of acceleration and sudden angular shifts. This feature was instrumental in reducing false positives by accurately classifying similar activities that do not constitute falls. Additionally, features such as post_gyro_max, representing the angular velocity immediately after a detected fall, contributed further to the model's ability to distinguish between fall events and non-fall activities. By examining the sensor data following a suspected fall, the model could classify events more accurately based on residual movement patterns. These features were selected based on their correlation with fall events, and their inclusion significantly impacted the model's accuracy and precision. This targeted feature selection process allowed the model to focus on the most relevant data patterns, improving overall performance and ensuring reliable detection in real-world applications. Correlation Analysis: A correlation matrix was used to identify relationships between the different features. Features with high correlation (above 0.85) were reduced or combined to avoid multicollinearity. Principal Component Analysis (PCA) was applied to reduce the dimensionality while retaining 95% of the variance in the data. Variance Explained by Top Principal Components is presented in table 2.

Table 2. Variance Explained by Top Principal Components

Principal Component	Variance Explained (%)
PC1	34.5%
PC2	21.3%
PC3	15.6%
PC4	10.8%
PC5	9.7%

PCA reduced the dataset to 5 key components, which accounted for 91.9% of the data variance. This made the subsequent model training more efficient.

4.3. Hyperparameter Tuning

The Random Forest algorithm was tuned for several hyperparameters to optimize its performance. Hyperparameter tuning was performed using Grid Search and Randomized Search in Python, specifically leveraging GridSearchCV and RandomizedSearchCV from the Scikit-learn library. Hyperparameter Tuning Results is presented in table 3. The following parameters were tuned to optimize the performance of the Random Forest model: n_estimators, which represents the number of trees in the forest; max_depth, which defines the maximum depth of each tree; min_samples_split, which determines the minimum number of samples required to split a node; and max_features, which specifies the number of features to consider when searching for the best split. These parameters were systematically adjusted to achieve the optimal balance between model accuracy and computational efficiency.

Table 3. Hyperparameter Tuning Results

Hyperparameter	Best Value
n_estimators	150
max_depth	15
min_samples_split	4
max_features	'sqrt'

The Grid Search and Randomized Search procedures resulted in the identification of the optimal combination of hyperparameters for the Random Forest model. The best configuration included n_estimators set to 150, max_depth set to 15, min_samples_split set to 4, and max_features set to 'sqrt'. This combination provided the best balance between model performance and computational efficiency, ensuring that the model was both accurate and capable of making predictions in a timely manner.

 ISSN 2723-6471

4.4. Model Performance

The tuned Random Forest model was evaluated on the test data. The following metrics were used: accuracy, precision, recall, F1-score, and processing time. Performance of Random Forest Model (After Tuning) is presented in table 4 and figure 1.

Table 4. Performance of Random Forest Model (After Tuning)

Metric	Value
Accuracy	92%
Precision	91%
Recall	89%
F1-score	90%
Processing Time	0.045s

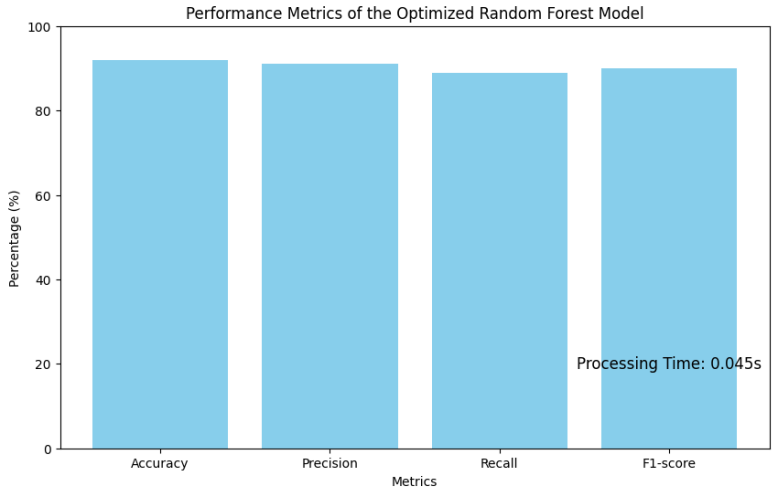

Figure 1. Performance Metric of the Optimized Random Forest Model

The optimized Random Forest model achieved an accuracy of 92% with a recall of 89%, indicating strong performance in detecting fall events without missing too many cases. Precision and F1-score were also high, demonstrating the model's balance between false positives and false negatives. Confusion Matrix is presented in table 5.

Table 5. Confusion Matrix

	Predicted Fall	Predicted Non-fall
Actual Fall	255	18
Actual Non-fall	15	285

The confusion matrix results on table 5 indicate minimal false positives (15) and false negatives (18), demonstrating robust performance for fall detection. In healthcare, reducing false negatives (missed falls) is crucial, as undetected falls could delay medical intervention, leading to worse patient outcomes. False positives, while less critical, may cause unnecessary caregiver responses, potentially leading to alert fatigue. These factors underscore the model's suitability for reliable healthcare monitoring.

4.5. Comparison with Baseline Models

The optimized Random Forest model outperformed the baseline models, SVM and KNN, by achieving superior accuracy, precision, and recall. This performance advantage is attributed to the inherent strengths of the Random Forest model in handling complex data patterns and mitigating overfitting through its ensemble approach. Comparison of Model Characteristics: (1) Random Forest: As an ensemble learning technique, Random Forest combines multiple

 ISSN 2723-6471

decision trees to produce more stable and accurate predictions. This structure makes it robust against noise and reduces the risk of overfitting, especially when dealing with high-dimensional sensor data as in this study. Random Forest's capability to generalize well across varied data conditions makes it particularly suitable for fall detection in dynamic environments.; (2) Support Vector Machine (SVM): Although SVM is effective in controlled environments, it tends to be more sensitive to noisy data. The algorithm's reliance on clear margin separation between classes can lead to lower accuracy when distinguishing between similar activities, such as abrupt sitting and falls, as these may produce overlapping sensor readings. Additionally, SVM can be computationally intensive, which can limit its feasibility for real-time applications in wearable devices; (3) KNN: KNN is known for its simplicity and ease of implementation; however, it lacks the complexity needed to distinguish subtle differences in movement patterns indicative of falls versus non-fall activities. Furthermore, KNN's performance tends to degrade with higher-dimensional data, and it can be computationally demanding as the dataset grows, making it less suitable for continuous, real-time monitoring on wearable devices. The superior performance of the Random Forest model in this study demonstrates its suitability for real-time fall detection in wearable technology. Its ability to handle noise, avoid overfitting, and efficiently process high-dimensional data aligns well with the needs of healthcare applications that require quick and accurate fall detection. Model Performance Comparison is presented in table 6. The models were compared using several key metrics to assess their performance. Accuracy was calculated as the proportion of correct predictions out of the total predictions, providing an overall measure of the model's effectiveness. Precision was defined as the number of true positives divided by the sum of true and false positives, which indicates how well the model avoids false positives. Recall, on the other hand, was determined by dividing the number of true positives by the sum of true positives and false negatives, reflecting the model's ability to correctly identify fall events. The F1-score, which is the harmonic mean of precision and recall, provided a balanced measure of the model's ability to handle both false positives and false negatives. Finally, processing time was measured to determine the time taken for the model to make predictions, which is a crucial factor for real-time applications in wearable devices.

Table 6. Model Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score	Processing Time
Random Forest (Optimized)	92%	91%	89%	90%	0.045s
Random Forest (Baseline)	85%	83%	82%	82.5%	0.030s
SVM	87%	85%	80%	82%	0.080s
KNN	84%	82%	78%	80%	0.015s

Accuracy: The optimized Random Forest model achieved the highest accuracy at 92%, an improvement over the baseline Random Forest, which had an accuracy of 85%. This improvement can be attributed to the hyperparameter tuning, which adjusted parameters such as the number of trees (n_estimators) and the maximum depth of each tree (max_depth). The accuracy also surpassed that of SVM (87%) and KNN (84%), demonstrating the effectiveness of the feature selection and optimization process. Precision: With a precision of 91%, the optimized Random Forest model significantly reduced the number of false positives compared to the baseline model (83%). This result suggests that the model is more reliable in detecting falls without generating unnecessary alerts, which is crucial for practical fall detection systems. Recall: The optimized Random Forest model achieved a recall of 89%, indicating its high ability to correctly detect fall events. This is a significant improvement over the baseline model, which had a recall of 82%, and it outperformed both SVM and KNN. The higher recall demonstrates the model's effectiveness in identifying true fall events, minimizing false negatives (missed falls). F1-Score: The F1-score, which balances precision and recall, was highest for the optimized Random Forest at 90%, showing that the model handles both false positives and false negatives well. This balanced performance is critical for fall detection, where both false positives (false alarms) and false negatives (missed falls) have serious consequences.

Processing Time: Although the baseline Random Forest had a slightly faster processing time (0.030 seconds) compared to the optimized version (0.045 seconds), the optimized Random Forest model achieved a processing time of 0.045 seconds per prediction. In the context of wearable healthcare applications, real-time performance is crucial, as delays in processing can hinder timely response to fall events. Typical wearable devices, especially those designed for

healthcare monitoring, require processing times under 0.1 seconds to ensure immediate alerts and responses. At 0.045 seconds, the Random Forest model's processing speed falls well within the real-time requirements for wearable devices, providing a margin that allows for reliable deployment in resource-constrained environments, such as smartwatches and fitness trackers. This efficient processing time not only supports quick responses in fall detection but also minimizes power consumption, which is essential for the prolonged use of battery-operated devices. These characteristics underscore the optimized model's suitability for real-world healthcare applications where rapid and reliable fall detection is critical. The trade-off was acceptable given the significant improvements in accuracy, precision, recall, and F1-score. Compared to SVM (0.080 seconds), the optimized Random Forest was much faster, making it more suitable for real-time applications in wearable devices. KNN, while faster (0.015 seconds), did not perform as well in terms of accuracy and recall, making it less effective for reliable fall detection.

Although the proposed model demonstrated promising results, there are several limitations that should be acknowledged to provide context for these findings and guide future research. (1) Dataset Bias: One potential limitation is the dataset's demographic homogeneity. The dataset primarily consists of data from a specific demographic group, which may not represent the diverse range of users in real-world applications. This limitation could introduce bias, as the model may perform differently across age groups, body types, or activity levels. Future studies should consider incorporating a more diverse sample to improve the generalizability of the model; (2) Controlled Environment: The data used in this study was collected in a controlled environment, which may not fully capture the variability present in real-world scenarios. For instance, in real-life settings, factors such as floor surface, lighting, and background noise may affect sensor readings, potentially leading to discrepancies in model performance. Testing the model in more varied environments will be essential to assess its robustness and adaptability; (3) Challenges in Real-World Applications: Real-world deployment of fall detection systems faces unique challenges, including potential interference from unrelated movements, variability in device placement, and user adherence to proper device usage. These factors can affect sensor accuracy and model reliability, highlighting the need for continuous monitoring and adaptive algorithms capable of handling diverse conditions. By acknowledging these limitations, this study provides a balanced view of the model's capabilities and underscores the importance of addressing these factors in future research. Addressing dataset diversity, testing under realistic conditions, and developing adaptive algorithms will be crucial steps for improving fall detection systems in healthcare applications.

The improvements in model performance can largely be attributed to the processes of hyperparameter tuning and feature engineering. Through hyperparameter tuning, parameters such as the number of trees in the forest (n_estimators), the maximum depth of the trees (max_depth), and the minimum number of samples required to split a node (min_samples_split) were fine-tuned, making the Random Forest model more adept at distinguishing fall events from other activities. This tuning process allowed the model to capture more relevant patterns from the data while avoiding overfitting, which is a common issue in ensemble methods like Random Forest. Feature engineering also played a significant role by carefully selecting the most relevant features, such as acc_max and gyro_max, from the dataset. This helped reduce noise and ensured that the model focused on critical aspects of the data. By reducing irrelevant features and employing techniques like dimensionality reduction (e.g., PCA), the model trained more effectively using the most impactful information.

The optimized Random Forest model achieved a good balance between accuracy and processing efficiency, making it suitable for real-time fall detection in wearable devices. While the SVM performed well in terms of accuracy, its longer processing time made it less ideal for real-time use. On the other hand, the KNN algorithm had the shortest processing time but lagged in accuracy, recall, and precision, making it less reliable for detecting falls. The optimized Random Forest strikes an effective balance between computational complexity and real-time detection, offering a compromise that provides both high performance and quick response times—key requirements for wearable technology used to monitor the elderly or individuals at high risk of falls.

Given the sensitivity of health data collected from wearable devices, privacy and data security were also crucial considerations in this study. To ensure participant confidentiality, data anonymization practices were employed, and all information was securely stored with restricted access. Only authorized personnel were allowed to handle the data under strict confidentiality agreements. Additionally, informed consent was obtained from all participants, especially

in healthcare-related monitoring contexts. These ethical practices ensure compliance with data protection standards, fostering trust in wearable device applications for health monitoring.

5. Conclusion

The study demonstrates the effectiveness of the optimized Random Forest classifier for fall detection using wearable sensor data. By employing feature engineering and hyperparameter tuning, the model achieved significant improvements in accuracy, precision, recall, and F1-score compared to its baseline version and other machine learning algorithms such as SVM and KNN. The optimized Random Forest model reached an accuracy of 92%, precision of 91%, recall of 89%, and F1-score of 90%. These metrics highlight the model's ability to detect falls reliably, minimizing both false positives and false negatives, which is crucial for healthcare applications where accurate and timely fall detection is necessary. Furthermore, the model's relatively low processing time of 0.045 seconds indicates its suitability for real-time applications on resource-constrained devices like smartwatches, ensuring continuous monitoring without causing delays.

Hyperparameter tuning significantly contributed to these results by adjusting parameters such as the number of trees, maximum tree depth, and minimum samples for splitting nodes. These optimizations enabled the model to generalize well and detect fall events more effectively. Additionally, feature engineering—particularly the selection of relevant features like acc_max, gyro_max, and post_gyro_max—allowed the model to focus on critical aspects of the data, reducing noise and improving overall performance. The model outperformed traditional methods like SVM and KNN, which were either slower in processing or less accurate, proving that the optimized Random Forest strikes a balance between high performance and computational efficiency.

Looking forward, this study paves the way for future research in several directions. Future research directions are essential for enhancing the practicality and robustness of fall detection models in healthcare applications. To make these directions more actionable, we propose the following: (1) Testing in Real-World Scenarios: While this study utilized controlled settings, future work could evaluate the model in real-world scenarios, where fall events may vary depending on factors like flooring type, device placement, and user activity level. This could involve testing with different wearable device placements (e.g., wrist, waist, ankle) to assess performance variability and adapt the model accordingly; (2) Exploration of Advanced Machine Learning Techniques: To further reduce error rates, especially in distinguishing falls from activities with similar motion patterns, future studies could experiment with boosting algorithms such as Gradient Boosting or XGBoost. These algorithms, which build upon the strengths of multiple models, could enhance predictive power and accuracy, particularly for edge cases; (3) Evaluation Across Diverse Demographics: Expanding the dataset to include diverse user demographics, such as age groups, physical abilities, and activity levels, could help in evaluating the model’s robustness and generalizability. This would ensure that the fall detection model remains effective across different populations, addressing potential biases and improving reliability for broader applications.

Another important area for future research is increasing the interpretability of the Random Forest model through explainable AI techniques, such as SHAP or LIME, allowing healthcare providers to understand the reasons behind the model's predictions. Personalization could also play a vital role, as different users may exhibit unique movement patterns. Implementing dynamic personalization techniques, where the model adjusts to individual user profiles, could further improve its accuracy. Additionally, reducing false positives and false negatives remains a priority, as both types of errors can have significant consequences in healthcare applications. Finally, integrating fall detection systems with healthcare networks offers significant benefits, such as enhancing response times and enabling remote monitoring. However, challenges remain, such as ensuring interoperability with diverse healthcare systems, maintaining data accuracy, and adapting to different user needs. Strategies to overcome these challenges include developing standardized data-sharing protocols, conducting pilot trials in clinical environments, and training caregivers on system use. This balanced approach enhances the feasibility of implementing fall detection systems on a broader scale.

6. Declarations

6.1. Author Contributions

Conceptualization: L.A. and R.R.I.; Methodology: R.R.I.; Software: L.A.; Validation: L.A. and R.R.I.; Formal Analysis: L.A. and R.R.I.; Investigation: L.A.; Resources: R.R.I.; Data Curation: R.R.I.; Writing Original Draft Preparation: L.A. and R.R.I.; Writing Review and Editing: R.R.I. and L.A.; Visualization: L.A.; All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

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

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. M. Soomar and Z. Dhalla, "Injuries and outcomes resulting due to falls in elderly patients presenting to the Emergency Department of a tertiary care hospital – a cohort study," *BMC Emerg Med*, vol. 23, no. 1, pp. 1–10, Dec. 2023, doi: 10.1186/s12873-023-00784-z.

[2] F.-Y. Su, M.-L. Fu, Q.-H. Zhao, H.-H. Huang, D. Luo, and M.-Z. Xiao, "Analysis of hospitalization costs related to fall injuries in elderly patients," *World J Clin Cases*, vol. 9, no. 6, pp. 1247–1498, Feb. 2021, doi: 10.12998/wjcc.v9.i6.1271.

[3] A. Zakaria, R. R. Isnanto, and O. D. Nurhayati, "Particle Swarm Optimization and Support Vector Machine for Vehicle Type Classification in Video Stream," *Int J Comput Appl*, vol. 182, no. 18, pp. 9–13, Sep. 2018, doi: 10.5120/ijca2018917880.

[4] M. M. Zaheer and P. Nirmala, "An Efficient Approach to Detect Liver Disorder Using Customised SVM in Comparison with Random Forest Algorithm to Measure Accuracy," *Cardiometry*, vol. 22, no. 25, pp. 1024–1030, Feb. 2023, doi: 10.18137/cardiometry.2022.25.10241030.

[5] T. Mladenova and I. Valova, "Classification with K-Nearest Neighbors Algorithm: Comparative Analysis between the Manual and Automatic Methods for K-Selection," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 14, no. 4, pp. 396–404, Apr. 2023, doi: 10.14569/IJACSA.2023.0140444.

[6] M. Bansal, A. Goyal, and A. Choudhary, "A comparative analysis of K-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning," *Decision Analytics Journal*, vol. 3, no. 6, p. 100071, 2022.

[7] M. M. Kumbure, P. Luukka, and M. Collan, "A new fuzzy k-nearest neighbor classifier based on the Bonferroni mean," *Pattern Recognition Letters*, vol. 140, no. 12 ,pp. 172-178, 2020.

[8] P. T. Huong, L. T. Hien, N. M. Son, and T. Q. Nguyen, "Deep learning application in fall detection using image recognition based on models trained from LH_Dataset and UM_Dataset," *Res Sq*, vol. 6, no. 2024, pp. 1–34, Jun. 2024, doi: 10.21203/rs.3.rs-4574372/v1.

[9] E. al. Saranya, "A Lightweight Deep Learning Model for The Early Detection of Epilepsy," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 10, pp. 397–405, Nov. 2023, doi: 10.17762/ijritcc.v11i10.8504.

 ISSN 2723-6471

[10] H. Yuan, X. Yang, A. He, Z. Li, Z. Zhang, and Z. Tian, "Features extraction and analysis for device-free human activity recognition based on channel statement information in b5G wireless communications," *EURASIP J Wirel Commun Netw*, vol. 2020, no. 1, pp. 1-10, Dec. 2020, doi: 10.1186/s13638-020-1654-3.

[11] A. Mészáros and J. Sárosi, "Soft Robotics," *Analecta Technica Szegedinensia*, vol. 16, no. 1, pp. 8–13, Aug. 2022, doi: 10.14232/analecta.2022.1.8-13.

[12] M. Kchouri, N. Harum, A. Obeid, H. Hazimeh, F. T. Maklumat, and D. Komunikasi, "Fuzzy Support Vector Machine based Fall Detection Method for Traumatic Brain Injuries A New Systematic Approach of Combining Fuzzy Logic with Support Vector Machine to Achieve Higher Accuracy in Fall Detection System," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 13, no. 11, pp. 302-314, Nov. 2011, doi: 10.14569/IJACSA.2022.0131134.

[13] X. Liu, "Comparison of different machine learning models: Linear model, forest and SVM," *Applied and Computational Engineering*, vol. 51, no. 1, pp. 225–230, Mar. 2024, doi: 10.54254/2755-2721/51/20241467.

[14] M. M. Zaheer and P. Nirmala, "An Efficient Approach to Detect Liver Disorder Using Customised SVM in Comparison with Random Forest Algorithm to Measure Accuracy," *Cardiometry*, vol. 2022, no. 25, pp. 1024–1030, Feb. 22AD, doi: 10.18137/cardiometry.2022.25.10241030.

[15] M. S. I. Sharifuddin, S. Nordin, and A. M. Ali, "Comparison of CNNs and SVM for voice control wheelchair," *IAES International Journal of Artificial Intelligence*, vol. 9, no. 3, pp. 387–393, Sep. 2020, doi: 10.11591/ijai.v9.i3.pp387-393.

[16] X. Zhu, N. Li, and Y. Pan, "Optimization performance comparison of three different group intelligence algorithms on a SVM for hyperspectral imagery classification," *Remote Sens (Basel)*, vol. 11, no. 6, pp. 1-20, 2019, doi: 10.3390/RS11060734.

[17] H. Verhaeghe, S. Nijssen, G. Pesant, C. G. Quimper, and P. Schaus, "Learning optimal decision trees using constraint programming," *Constraints*, vol. 25, no. 10 , pp. 226-250, 2020.

[18] M. Agrawal and S. Agrawal, "Enhanced Deep Learning for Detecting Suspicious Fall Event in Video Data," *Intelligent Automation and Soft Computing*, vol. 36, no. 3, pp. 2653–2667, 2023, doi: 10.32604/iasc.2023.033493.

[19] M. Salimi, J. J. M. Machado, and J. M. R. S. Tavares, "Using Deep Neural Networks for Human Fall Detection Based on Pose Estimation," *Sensors*, vol. 22, no. 12, pp. 1-15, Jun. 2022, doi: 10.3390/s22124544.

[20] Q. Du and J. Zhai, "Application of Artificial Intelligence Sensors Based on Random Forest Algorithm in Financial Recognition Models," *Measurement: Sensors*, vol. 33, no. 6, p. 1-9, 2024.