Journal of
Sensor and Actuator Networks

MDPI

Article

A Study of Fall Detection in Assisted Living: Identifying and Improving the Optimal Machine Learning Method

Nirmalya Thakur * and Chia Y. Han

Department of Electrical Engineering and Computer Science, University of Cincinnati, Cincinnati, OH 45221-0030, USA; han@ucmail.uc.edu
* Correspondence: thakurna@mail.uc.edu

Abstract: This paper makes four scientific contributions to the field of fall detection in the elderly to contribute to their assisted living in the future of Internet of Things (IoT)-based pervasive living environments, such as smart homes. First, it presents and discusses a comprehensive comparative study, where 19 different machine learning methods were used to develop fall detection systems, to deduce the optimal machine learning method for the development of such systems. This study was conducted on two different datasets, and the results show that out of all the machine learning methods, the k-NN classifier is best suited for the development of fall detection systems in terms of performance accuracy. Second, it presents a framework that overcomes the limitations of binary classifier-based fall detection systems by being able to detect falls and fall-like motions. Third, to increase the trust and reliance on fall detection systems, it introduces a novel methodology based on the usage of k-folds cross-validation and the AdaBoost algorithm that improves the performance accuracy of the k-NN classifier-based fall detection system to the extent that it outperforms all similar works in this field. This approach achieved performance accuracies of 99.87% and 99.66%, respectively, when evaluated on the two datasets. Finally, the proposed approach is also highly accurate in detecting the activity of standing up from a lying position to infer whether a fall was followed by a long lie, which can cause minor to major health-related concerns. The above contributions address multiple research challenges in the field of fall detection, that we identified after conducting a comprehensive review of related works, which is also presented in this paper.

Keywords: fall detection; elderly; machine learning; assisted living; smart homes; artificial intelligence; human-computer interaction; internet of things; pattern recognition; pervasive computing

Citation: Thakur, N.; Han, C.Y. A Study of Fall Detection in Assisted Living: Identifying and Improving the Optimal Machine Learning Method. *J. Sens. Actuator Netw.* **2021**, *10*, 39. https://doi.org/10.3390/jsan10030039

Academic Editors: Antonio Coronato and Giovanni Paragliola

Received: 17 April 2021
Accepted: 21 June 2021
Published: 24 June 2021

1. Introduction

People live longer these days due to advanced healthcare facilities. The population of elderly people is increasing at a rate higher than any other age group, and there are currently 962 million elderly people across the world [1]. According to a recent study [2], by the year 2050, approximately 20% of the world's population will be aged 60 years or more. The process of aging is accompanied by a decline in physical, cognitive, and motor skills. Due to the slow degradation of these abilities, falls can happen unexpectedly at any moment of the day and at any place. The rate of falling increases with age. According to [3], a fall is an unanticipated incident of coming down on the ground or floor triggered by a push or pull, environmental features, unconsciousness, or any similar health-related limitations or disorders. It involves an involuntary change in an individual's posture, resulting in the person lying on the ground. The consequences of falls can affect the overall health, well-being, and quality of life of the elderly in multiple ways, including limiting their abilities to perform activities of daily living (ADLs). Although there are several definitions of ADLs, according to [4], ADLs may be broadly defined as the set of daily routine tasks and activities essential for one's sustenance, which a person usually performs in their living environment. ADLs may broadly be classified into the following five categories: personal hygiene, dressing, eating, maintaining continence, and mobility.

On a global scale, falls are the second most common cause of accidental deaths. Falls are also considered the most likely reason for older adults to develop traumatic brain injuries [5]. On an annual basis, around 30% of elderly people fall at least once a year, and the rate of falling in the elderly is expected to increase to 42% over the next few years [6,7]. Falls have severely affected the increasing aging population in the United States. In the United States, deaths due to falls have increased by 30% since 2009. Every 11 s, an elderly person who has fallen has to be taken to the emergency room. Every 19 min, an older adult dies from a fall in the US. There are over 3 million cases of emergency, 800,000 hospitalizations, and more than 32,000 deaths in the elderly from falls every year [8–10]. This rate of hospitalizations and deaths from falls is expected to increase. It is predicted that by 2030, every hour, seven older adults will die in the US after having experienced a fall. The medical and healthcare-related costs of falls in the elderly account for a total of USD 50 billion to the US economy on an annual basis [11].

The causes of a fall can be many, and they can be broadly categorized as internal and external [12]. Here, external causes are related to all the environment-based factors that can contribute to a fall. These include slippery surfaces, stairs, etc. Internal causes of a fall refer to the individual's factors, which include impairments in vision, cramps, weakness in muscular skeleton structure, chronic disorders, etc. In addition to consequences from physical injuries of minor to a major degree of severity, falls can also have a multitude of impacts on the various aspects of well-being, such as personal, social, emotional, and cognitive. These effects of a fall can be broadly summarized as individual—injuries, bruises, blood clots; social life—reduced mobility leading to loneliness and social isolation; cognitive or mental—fear of moving around, loss of confidence in carrying out ADLs; and financial—the cost of medical treatment and caregivers.

In addition to detecting falls when the falls occur, it is also important to detect whether a long lie follows a fall. An individual experiences a long lie when they have been lying on the ground after a fall for more than an hour without being able to get up [13]. According to [14], 47% of individuals who experience a fall cannot get up on their own after the fall. To add, around 50% of elderly who experience a long lie are likely to die within the next six months [15]. A long lie can also cause localized muscle injuries, tissue damage, pressure injuries, nerve issues, carpet burns, dehydration, hypothermia, pneumonia, and fear of falling, which can affect the quality of life, health, and well-being of the elderly [13]. Thus, it is essential to track whether a fall is being followed by a long lie, as a long lie can lead to both major and minor health-related concerns for the elderly.

Although there have been numerous recent works [16–43] related to fall detection over the last few years, as discussed in detail in Section 2, several research challenges still exist. These include (1) the lack of a comprehensive comparative study of different machine learning methods to deduce the optimal machine learning method for the development of fall detection systems and applications; (2) the high possibility for fall-like motions such as being on all fours (both hands and legs touching the ground at the same time) being classified as false positives, as most of the existing frameworks for fall detection have been binary classifiers; (3) the low accuracy of the systems that focused on addressing the issue of such false positives and studied fall-related activities; and (4) the inability of these fall detection systems to track whether the person is experiencing a long lie after the fall.

In the work presented in this paper, we address all the above challenges. This paper is presented as follows. In Section 2, we present a comprehensive review of recent works in fall detection and further discuss the research challenges that exist. In Section 3, we present our approach and the results for developing a k-nearest neighbor (k-NN)-based multilabel classifier that can detect falls and track fall-like motions. Section 4 presents the comprehensive comparison study where we developed, implemented, and compared the performance characteristics of 19 machine learning methods to deduce the optimal learning method for fall detection. In Section 5, we discuss and present the results of the proposed methodology for improving the performance accuracy of the k-NN-based machine learning-based approach for fall detection and fall-like motions proposed in

Section 3, to increase the trust and reliance on such systems. It is followed by Section 6, which includes a comparative discussion where we outline how our work addresses the research challenges in this field and outperforms similar works in this discipline. Section 7 concludes the paper, summarizing the contributions of the same and outlining the scope for future work.

2. Related Work

The recent research related to assisted living technologies and their related disciplines, with a specific focus on fall detection, is outlined in this section. According to [16], assisted living technologies may broadly be classified into three different generations. The first-generation technologies consisted of systems and gadgets that were assistive in nature only when they received a request or response from the user for help. An example of a first-generation assisted living technology would be a wearable device with a panic or help button to call for help. A user wearing that device could ask for help from a caregiver or medical professional by pressing a button in the event of an emergency, for instance, a fall. The major drawback of these technologies was that they were not functional without the user's response or request. Elderly people might sometimes be incapacitated to press the help button due to a fall, in which case such a wearable device serves no purpose at all. This created the need for developing the second generation of assisted living technologies. Second-generation technologies were characterized by their abilities to sense when the user needed any assistance. These devices tracked health-related, user behavior-related, and user interaction-related data to trigger alarms to alert caregivers or medical personnel. However, second-generation technologies also had limitations. First, they could not prevent emergencies, as the alarms were triggered after the user faced an emergency. Second, there were instances of false positives, and finally, elderly people felt the systems were too intrusive. The third generation of assisted living technologies has been emerging recently to address these limitations. These refer to smart assistive systems that use a myriad of technologies such as artificial intelligence, machine learning, sensor networks, and their related applications to detect and predict any assistive needs, for instance, in a fall.

Next, we review the recent advances in fall detection based on these different generations of assisted living technologies. Liu et al. [17] developed a system that used the k-NN classifier to study multimodal components of human postures to detect falls. The system studied the ratio and difference of human body silhouette bounding box height and width to detect falls during different activities. An inclinometer-based approach was proposed by Sun et al. [18] for fall detection. The approach studied the angle variations recorded by the inclinometer during walking and similar activities, including a fall. The authors developed a threshold condition to detect falls based on the degree of angle variation during different motions. Rafferty et al. [19] developed a thermal vision sensing framework to study human movements and postures to detect falls. The work involved setting up thermal vision sensors on the ceiling of the simulated IoT-based space and then using computer vision algorithms to detect falls. The use of activity theory and recent advances in the same were used by Castillo et al. [20] to develop a fall detection system. The system analyzed the video feed and interpreted the accelerometer data for detecting falls during different activities.

The work of Fu et al. [21] involved using image-processing principles and vector analysis concepts to study the human behavior and posture data to detect falls in a simulated IoT-based room. Willems et al. [22] proposed a grayscale video-data analysis-based approach for fall detection in real-time. The approach was equipped with multiple features, such as background subtraction, shadow removal, ellipse fitting, and calculating the posture angle and aspect ratio to perform fall detection. Feng et al. [23] used a combination of deep learning and other machine learning principles to develop a fall detection system. The concept involved combining the outputs of these learning approaches based on certain rules that the authors proposed. A PCA-based fall detection system was proposed by

Jokanovic et al. [24]. In the proposed approach, the authors used eigen images to classify different types of motions, including falls. The study revealed that their work outperformed the feature extraction methods for fall detection published until that time. In [25], Bian et al. proposed a support vector machine (SVM)-based machine learning classifier for fall detection. The approach used the data from depth cameras and the associated RGB data to train the SVM classifier to develop the machine learning model. In the fall detection approach proposed by Ozcan et al. [26], the camera had to be carried by the user instead of the same being mounted at strategic locations, as had been observed in previous studies. The system used a decision tree-based approach to detect falls based on studying multimodal components of the image data coming from the camera carried by the user.

Another decision tree-based learning approach for fall detection was proposed by Lai et al. [27]. The approach tracked and analyzed the data coming from triaxial accelerometers mounted on different body parts of the user to perform behavior analysis to detect any anomalies such as falls. The system also studied the accelerometer data to calculate the impact of the fall after a fall occurred. In [28], Hakim et al. proposed a methodology for fall detection that was smartphone data-driven. The authors developed an algorithm in Matlab that was SVM-based. The SVM classifier used the data from the built-in inertial measurement unit (IMU) of the user's phone to study characteristics of motion data to detect falls. Tomii et al. [29] used Doppler sensors to develop a fall detection system that was k-NN-based. The system used three such sensors and could detect falls during different activities in different directions. Espinosa et al. [30] developed a neural network-based approach for the detection of falls. The approach used the data from multiple cameras and extracted a range of features using an optical flow method. This optical flow method provided information about the relative motion between two consecutive images by interpreting the image data. Nakamura et al. [31] developed a fall detection system that was neural network-driven. The approach used spectrogram images and Wi-Fi CSI data to train this learning model to develop the fall detection classifier.

The fall detection framework proposed by Balli et al. [32] used a smartwatch for data collection. The human behavior and activity-related data collected by this smartwatch were interpreted and analyzed by a random forest model to detect falls. Dhole et al. [33] developed a prototype for collecting and analyzing EEG data for fall detection. The prototype resembled a helmet that the user wore, and the authors developed a random forest classifier to detect falls based on this EEG data. A k-NN-based machine learning model for tracking falls was proposed by Ramirez et al. [34]. The system extracted features from human poses during different activities and used computer vision principles to detect falls. Like [30], another neural network-based fall detection system was proposed by Tahir et al. [35], which improved upon the associated computational costs without compromising on the fall detection accuracy.

A deep learning-based approach was used by Jácome et al. [36] for the development of a framework for studying and detecting falls. The architecture of the framework consisted of two deep learning models and implemented concepts of virtualization to detect falls. A video data-based activity analysis approach for fall detection was proposed by Dhiraj et al. [37]. The work involved analyzing 360-degree video data of the user to analyze activities to track falls. Ngu et al. [38] used a naïve bayes approach to develop a fall detection system that used the data from a smartwatch. The work of Khan et al. [39] involved a wearable device for fall detection. The device consisted of a camera, gyroscope, and accelerometer that communicated with a microcomputer. The system involved using a naïve bayes classifier to develop a binary classification model to detect falls. A gradient boosted trees approach was proposed by Ning et al. [40] for fall detection. The system studied human behavior and posture data to detect falls during different activities. In [41], Cahoolessur et al. proposed another gradient boosted trees-based learning approach that was primarily a binary classifier that could detect falls and other user activities in a simulated IoT-based environment. The authors developed a wearable device that the user had to wear on their waist and the system used a cloud computing-based architecture to

implement the machine learning model. Cai et al. [42] used a single-layer decision tree that used accelerometer data and an AdaBoost approach to detect falls. Lee et al. [43] used a similar threshold-based approach to distinguish falls from other daily activities. The approach used the acceleration data recorded by the user's phone and the threshold criteria to detect falls. Table 1 summarizes these related works in terms of their source of data or the datasets that were used for the development of the associated methodologies, as well as whether the data came from wearables or non-wearable sensors.

Table 1. Summary of recent works in fall detection along with their source of data.

Work	Data	Use of Wearables	Use of Non-Wearables
Liu et al. [17]	Human body silhouette bounding box dimensions	-	✓
Sun et al. [18]	Angle variations from inclinometer recordings	-	✓
Rafferty et al. [19]	Thermal vision sensing	-	✓
Castillo et al. [20]	Video feed and accelerometer data	✓	✓
Fu et al. [21]	Behavior and posture data	-	✓
Williams et al. [22]	Grayscale video	-	✓
Feng et al. [23]	Postural orientations	-	✓
Jokanovic et al. [24]	Eigen images during motion	-	✓
Bian et al. [25]	Depth cameras and the associated RGB information	-	✓
Ozcan et al. [26]	Video data from a movable camera	✓	✓
Lai et al. [27]	Triaxial accelerometers	✓	-
Hakim et al. [28]	Inertial measurement unit (IMU) of the user's phone	✓	-
Tomii et al. [29]	Doppler sensor data	-	✓
Espinosa et al. [30]	Data from multiple cameras	-	✓
Nakamura et al. [31]	Spectrogram images and Wi-Fi CSI data	-	✓
Balli et al. [32]	Activity and movement data from a smartwatch	✓	-
Dhole et al. [33]	EEG data	✓	
Ramirez et al. [34]	Analysis of images of different postures	-	✓
Ahmad et al. [35]	Accelerometer data	✓	-
Jácome et al. [36]	Data from resource-constrained devices (fog nodes)	✓	-
Dhiraj et al. [37]	360-degree video data	-	✓
Ngu et al. [38]	Smartwatch activity data	✓	
Khan et al. [39]	Camera, gyroscope, and accelerometer data	✓	✓
Ning et al. [40]	Human behavior and posture data	✓	-
Cahoolessur et al. [41]	Human behavior data	✓	-
Cai et al. [42]	Three-axis acceleration, three-axis angular acceleration	✓	-
Lee et al. [43]	Acceleration data collected from a smartphone, GPS data, and WiFi signals	✓	-

Despite the above works in this field, several research challenges need to be addressed. They are as follows:

1. A range of machine learning methods such as random forest, artificial neural network, decision tree, support vector machine, k-NN, gradient boosted trees, naïve bayes, and deep learning have been used by researchers to develop fall detection systems. Still, the best machine learning method in terms of performance accuracy for the development of such systems and applications is unknown. Thus far, none of the works in this field have compared the performance characteristics of different machine learning approaches to deduce the optimal learning approach.
2. Most of the fall detection frameworks developed so far have been binary classifiers where the framework would classify a motion or a posture as fall or not a fall. There is a high possibility of fall-like motions such as being on all fours leading to false positives. Such false positives can cause alert fatigue [44] in caregivers or medical personnel. Alert fatigue can lead to caregivers or medical personal becoming desensitized to alarms for falls, causing a decrease in quality of care or even no timely care. Therefore, it is necessary to detect falls and such fall-like motions when monitoring human activities.

3. Some of the works [19–24] that focused on detecting false positives and studying fall-like motions did not achieve high levels of performance accuracy in terms of detecting falls. Thus, it is the need of the hour that fall detection systems be developed in a way that minimizes false positives and can detect fall-like motions while achieving high levels of performance accuracy so that the underlining systems are considered reliable and can be trusted.
4. Falls can be fatal if the fall is associated with a long lie [13–15]. A long lie can be detected by tracking whether the person who experienced a fall could get up after the fall. None of the above works have focused on implementing such an approach. Therefore, it is necessary for the future of fall detection systems to detect falls as well as long lies.

Addressing the above challenges by integrating the latest advancements and technologies in human–computer interaction, artificial intelligence, machine learning, internet of things, pattern recognition, and pervasive computing serves as the main motivation of this work.

3. Machine Learning-Based Approach for Detecting Falls and Fall-Like Motions

This section presents the system architecture, methodology, and results of our proposed k-NN-based machine learning approach for detecting falls and fall-like motions. The motions and behavioral patterns that our system can detect include falling, lying, being on all fours, standing up from lying, and other activities. Here, "lying" refers to the person lying on the ground after having experienced a fall, and "other activities" refers to all kinds of activities related to ADLs, which does not include falling, lying, standing up from lying, or being on all fours. This approach's methodology and system architecture is outlined in Section 3.1, and the associated results are presented in Section 3.2.

3.1. Methodology and System Architecture of the Machine Learning-Based Approach for Detecting Falls and Fall-Related Motions

The system architecture of this approach consists of four primary modules—posture recognition, data collection and data preprocessing, learning module, and performance module. Next, we outline each of these modules. The system architecture is shown in Figure 1. The posture recognition module is associated with the functionality to detect the user's posture at each time instant by interpreting the motion and movement-related data during different activities. One methodology for developing such a posture recognition approach is by analysis of the accelerometer data (obtained from a wearable sensor) in the X-, Y-, and Z-directions during different activities at different time instants [45].

This is done by calculating the acceleration vector and the orientation angles of the user's acceleration recorded for each of the three axes. Traditionally, such inertial sensors had a dedicated system of working where a proof mass, m, was suspended on a mechanical frame by use of a spring, km, that responded to an input force, F, that represented the desired quantity to be measured by the sensor. This applied force would create a displacement, x, of the present proof mass, and this displacement was measured by the device to sense the associated force. For instance, the input force could result from the acceleration of the proof mass, which would happen if the device were an accelerometer. In a different scenario, the input force could be the result of Coriolis acceleration related to the angular rotation of the proof mass, which would happen if the device were a gyroscope. Thus, the outputs of such sensors depend on the movements, position, and orientations of the device and the person who holds the sensor [46]. Our proposed system is based on advanced versions of such inertial sensors, such as [47], which are sophisticated and consist of three-axis accelerometers. A three-axis accelerometer sensor outputs the acceleration values measured along the X-, Y-, and Z-directions, and by using the same the velocity and orientation angle can also be computed.

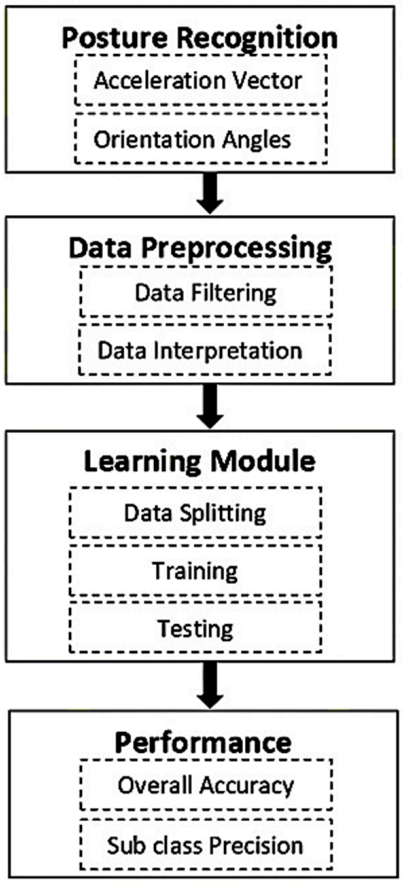

Figure 1. System architecture of the machine learning-based approach for detecting falls and fall-like motions.

In addition to measuring the acceleration value, such sophisticated sensors also output the relative vector projections of the acceleration vector represented in a three-dimensional coordinate system by using their own coordinate system, which is built into the sensor at the time of its design and development. Because of Earth's acceleration due to gravity, any object on the surface of the Earth gets pulled towards the center of the Earth, with force referred to as "g". When the accelerometer is at rest, it only measures Earth's gravity. These accelerometers are affixed to different locations on the person's body, based on the specific part of the body whose acceleration is to be measured. The "g" value is important for the working of such sensors. Such sensors analyze this "g" value to compute the sensor orientation angle and to differentiate between different body postures. The orientation angle in this context refers to the angle between the acceleration vector and each of the axes—X, Y, and Z. The formula to compute this orientation angle for the X-axis is shown in Equation (1). In a similar manner, the orientation angles for the Y- and Z-axes can be computed.

$$\cos(An(x)) = \frac{An(x)}{\sqrt[2]{An(x)^2 + An(y)^2 + An(z)^2}} \quad (1)$$

where:

$\cos(An(x))$ = cosine value of the angle between the acceleration vector and the X-axis
$An(x)$ = acceleration in the X-direction
$An(y)$ = acceleration in the Y-direction
$An(z)$ = acceleration in the Z-direction

By computing these orientation angles, every movement, posture, and motion can be studied in the form of a unique spatial orientation. The position on the body where

the sensor is affixed also plays a crucial role in this scenario. For example, the position of an accelerometer placed on the user's chest is always perpendicular to the ground for movements such as standing and sitting but parallel to the ground for other categories of movements related to falls, such as lying on the ground and being in a position where both arms and legs are touching the ground (on all fours). Based on the above system configurations of the sensors and the associated sensor data that were used, our proposed framework did not require that we include any forms of coordinate transformations. The calculation of the orientation angles along the three axes helps in studying the postures, both static and dynamic, associated with different ADLs. Here, dynamic postures refer to the postural orientations of the user when the user is in some form of motion. Falls or fall-related motions can be detected by interpreting the sudden variations in the values of Equation (1), and likewise for the other two axes, with respect to time. The acceleration pattern during a fall event is characterized by a decrease in the magnitude of acceleration followed by a rapid increase in the same, which can be tracked and recorded. The data collection and data preprocessing module consist of the methodology for collecting the data and for preprocessing the same. Here, the processing steps include (1) removing the attributes from the data that are not necessary for the development of the next module—the learning module, which consists of the machine learning classifier, and (2) filtering relevant information from the data using a data filter. We set up an IoT-based data collection framework in our lab space that can be used to collect real-time Big Data associated with different user interactions with context parameters during different ADLs [48]. The setup consists of wireless sensors and wearables and can be used to track different activities such as using a microwave, reading, working, relaxing, and studying, usually performed within the premises of a smart home environment. We also obtained Institutional Review Board (IRB) approval from our institution to carry out real-time experiments with human subjects within this IoT-based simulated environment in our lab. However, due to the COVID-19 situation and the associated "work from home" recommendations by various government sectors in the United States [49], we could not collect real-time data using this data collection framework.

Therefore, we used two datasets to validate and evaluate our proposed approach and its associated functionalities. These datasets were specifically chosen because their associated attributes were very similar to the real-time data that our data collection framework [48] could have collected. The first dataset was developed by Kaluža et al. [50]. This dataset consists of 164,860 rows of data collected from five individuals, denoted as A to E, performing different activities, including falls, in an IoT-based environment. The data capture process involved tracking the motion and behavior-related data using accelerometer sensors placed on four locations—left ankle, right ankle, chest, and belt—of each of these participants. The accelerometer sensors on these four locations were represented in the dataset with unique IDs: 010-000-024-033, 020-000-032-221, 010-000-030-096, and 020-000-033-111, respectively. The absolute counts of the data that were obtained from these respective sensors were 43,526, 42,973, 42,560, and 35,801, respectively. The average value of the acceleration in the X-, Y-, and Z-directions were recorded as 2.811, 1.697, and 0.418, respectively. The data from these sensors were tracked in real-time as the participants performed a range of activities in the simulated IoT-based space. The different activities present in this dataset include walking, falling, lying, lying down, being on all fours, sitting, sitting down, sitting on the ground, standing up from lying, standing up from sitting, and standing up from sitting on the ground. The number of instances these respective activities were found in the dataset were 32,710, 2973, 54,480, 6168, 5210, 27,244, 1706, 11,779, 18,361, 1381, and 2848, respectively. Here the activity of "lying" refers to the person being in a lying position after having experienced a fall. In addition to these activities, the other attributes present in this dataset included the tag numbers of the sensors, the tag identifiers, the date and time, and the acceleration values in the X-, Y-, and Z-directions. As we were interested in detecting falls and fall-related activities, the activities that were not related to falls were labeled as "other activities".

The second dataset, developed by Tabbakha et al. [51], contains activity and human behavior-related data collected from both wireless sensors and wearables in an IoT-based environment. The data attributes present in this dataset include the accelerometer data, gyroscope data, and the received signal strength indicator (RSSI) data obtained from Bluetooth low energy (BLE) beacons and BLE scanners. The simulated smart-home environment in which these data were collected consisted of four rooms—kitchen, bedroom, office, and bathroom. To collect the data as presented in this dataset, the authors developed a wearable device by using the Linkit 7697 and the MPU6050 sensors. This wearable device tracked the behavior-related information of the user and collected position-related data with respect to the user's location in each of the four rooms. A BLE beacon was incorporated into this wearable, and Raspberry Pi-based BLE scanners were installed at different locations of the IoT-based space. These scanners tracked the position of the user by sensing the BLE beacon and interpreting the associated RSSI data. The authors set the advertising interval of the BLE beacon to 100 ms with a transmitting power of up to −30 dBm. The activities performed in each of these rooms included sleeping, changing clothes, relaxing, moving around, cooking, eating, working, defecating, and experiencing an emergency. The respective activities were found to occur 25, 24, 24, 47, 26, 26, 21, 31, and 71 times, respectively. Here, as per the definition of the authors, an emergency constituted detecting the user in a lying position (from a fall) in an environment where a user is not supposed to lie down, for instance, in the bathroom. The data from the accelerometer and gyroscope from the wearable were sampled at a rate of 20 Hz. A total of 20 volunteers (10 males and 10 females) had participated in the experimental trials. The average values of the acceleration along the X-, Y-, and Z-directions were recorded as 2244, 16,708, and 15,148, respectively, after the completion of these experimental trials. The average value of the gyroscope data along the X-, Y-, and Z-directions were recorded as −590.790, 250.417, and 49.963, respectively, after the completion of the data collection.

For the data preprocessing steps for [50], we referred to the findings of [45]. According to [45], placing one accelerometer on the chest region is considered the best placement for an accelerometer for posture recognition, as compared to (1) placing an accelerometer on other locations on the human body, or (2) combinations of two or more accelerometers placed at different locations on the human body. These findings helped to ensure that we did not have to perform permutation and combination by selecting two or more sensor arrangements out of sensors placed on the left ankle, right ankle, chest, and belt to evaluate the optimal approach. Therefore, we filtered the data [50] to work with the readings obtained from the chest accelerometer that recorded the motion-related information during the different activities each of the five users performed. In other words, the data obtained from the accelerometers placed on the left ankle, right ankle, and belt were removed from the dataset and were not considered for this study. This data filtering and unwanted data elimination were done using a data filter. After this process, the dataset consisted of 35,801 rows. After that, we performed the data labeling as discussed above, where all other activities that were not falling, lying, standing up from lying, or being on all fours were labeled as "other activities". After the data labeling, we had 18,138 instances of other activities, 614 instances of falls, 11,930 instances of lying, 3968 instances of standing up from lying, and 1151 instances of being on all fours.

To preprocess the data present in [51], we removed the attributes from the data that were not associated with the acceleration data of the user. These attributes included bedroom, living room, kitchen, office, bathroom, location, gyroscope-X, gyroscope-Y, gyroscope-Z, and activity. After completing the data preprocessing steps on this dataset [51], the remaining attributes were acceleration-X, acceleration-Y, acceleration-Z, and action, where the "action" attribute consisted of falls and other activities. As per the data description of this dataset, we labeled "emergency" as "fall" and all other actions in the "action" attribute as "not a fall" .

When working with each of these datasets, we split the data into the training set and test set for the learning module by using a data splitter. A total of 80% of the data was

used for training, and 20% of the data was used for testing the proposed k-NN-based machine learning classifier. We evaluated our approach with several other ratios between the training set and the test set and found that when 80% of the data was used for training and the remaining 20% was used for testing, the learning method achieved the highest performance accuracy, so we used this ratio to develop and evaluate our approach. As per the system architecture shown in Figure 1, we implemented all the steps associated with this methodology in RapidMiner [52]. RapidMiner is a software application development tool that allows for the development, implementation, and testing of various data science and machine learning-related algorithms. We used the RapidMiner Studio, version 9.9.000, on a Microsoft Windows 10 computer with an Intel (R) Core (TM) i7-7600U CPU @ 2.80 GHz, two core(s), and four logical processor(s). The free version of RapidMiner Studio has a data processing limit of 10,000 rows, so we used the education license of RapidMiner to overcome this data processing limit. With the education license, there is no limit on the number of rows in the dataset that can be processed.

As we used RapidMiner to develop this proposed methodology, we will define two terminologies specific to RapidMiner here in a broad manner. They are "process" and "operator". In RapidMiner, an "operator" refers to one of the multiple building blocks of an application associated with specific functionalities that can be changed or modified either statically or dynamically, both by the user and by the application based on the specific need. In RapidMiner, certain operators are already developed in the software tool, which can be customized or updated. The tool also allows for the development of new operators based on any specific need. A continuous, logical, and operational collection of "operators" linearly or hierarchically representing a working application with one or more output characteristics is referred to as a "process". The RapidMiner process that represents this methodology is shown in Figure 2. The "dataset" operator was used to import the given datasets into the RapidMiner platform. Then we developed the "posture recognition" operator as per the characteristics of the posture recognition module of the system architecture, shown in Figure 1. This posture recognition operator was connected to the "data preprocess" operator, which was developed to perform the necessary data filtering and data preprocessing as outlined above. After that, the built-in "split data" operator was used to split the data into the training set and test set, where 80% of the data was used to train and 20% of the data was used to test the k-NN-based multilabel classification model. This k-NN classifier was implemented using the built-in "k-NN" operator in RapidMiner. The following is the pseudocode of the k-NN learning method:

1. Calculate "d(x, xi)" i = 1, 2, ..., n; where d denotes the Euclidean distance between the points.
2. Arrange the calculated n Euclidean distances in non-decreasing order.
3. Let k be a +ve integer, take the first k distances from this sorted list.
4. Find those k-points corresponding to these k-distances.
5. Let ki denotes the number of points belonging to the ith class among k points, i.e., $k \geq 0$.
6. If ki >kj $\forall$ i $\neq$ j, then put x in class i.

To evaluate the characteristics of the learning method, we used the built-in "performance" operator in RapidMiner. This operator can be used to calculate the overall accuracy and the individual class precision values of a learning method, which is represented in the form of a confusion matrix. The results of the same are further discussed in Section 3.2. We developed two different processes in RapidMiner corresponding to the two different datasets [50,51]. The only difference in these two processes was that the dataset operator consisted of a different dataset each time.

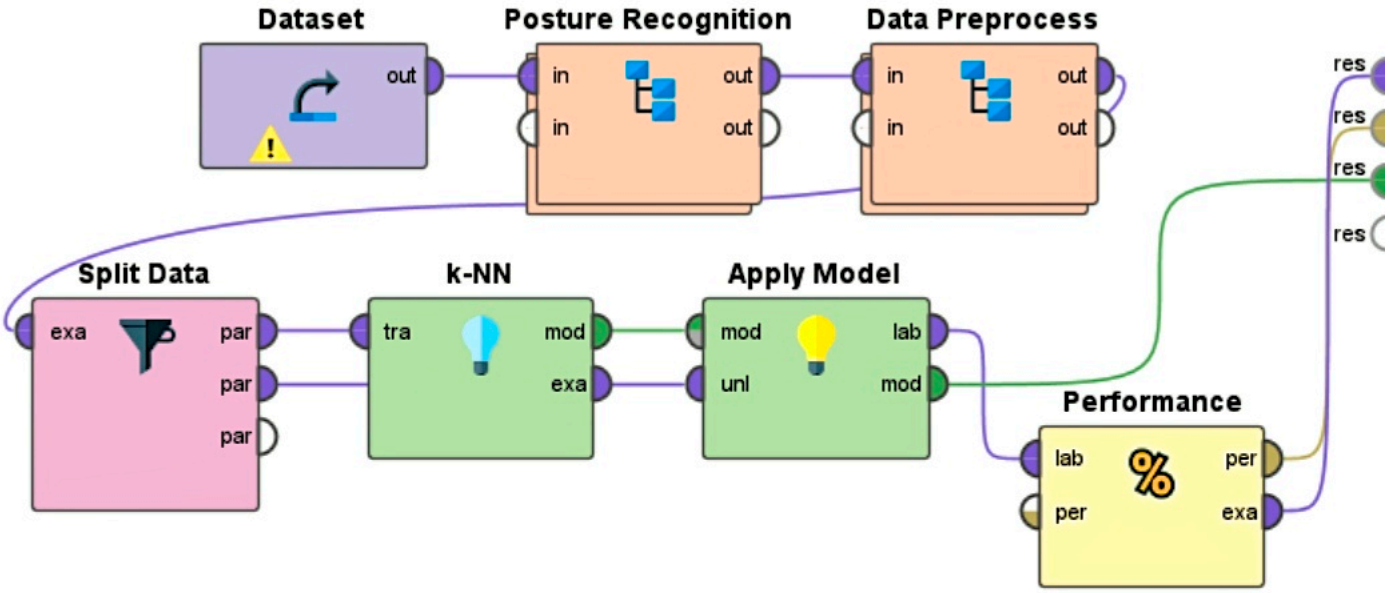

Figure 2. The process that was developed in RapidMiner as per the system architecture of the machine learning-based approach for detecting falls and fall-related motions.

3.2. Results of the Machine Learning-Based Approach for Detecting Falls and Fall-Related Motions

This section presents the results of the RapidMiner processes as shown in Figure 2, which was developed as per the system architecture of this approach, represented in Figure 1. To calculate the performance accuracy of the model, the overall performance accuracy and the respective class precision values were calculated by using the performance operator in RapidMiner, in the form of a confusion matrix, as per Equations (2) and (3).

$$\text{Overall\_Accuracy} = \frac{True\_P + True\_N}{True\_P + True\_N + False\_P + False\_N} \quad (2)$$

$$\text{Class\_Precision} = \frac{True\_P}{True\_P + False\_P} \quad (3)$$

where:

Overall_Accuracy = overall accuracy of the machine learning method
True_P = true positive
True_N = true negative
Class_Precision = sub class precision

Figure 3 shows the confusion matrix that shows the performance accuracy of the RapidMiner process shown in Figure 2 when we imported the first dataset [50] using the dataset operator. As shown in Figure 3, this k-NN-based multilabel machine learning approach for detecting falls and fall-related motions achieved an overall performance accuracy of 98.32%. It could detect the motions that correspond to falling, lying, being on all fours, and standing up from lying, with respective class precision values of 87.72%, 98.42%, 96.09%, and 96.71%. Here, "lying" refers to the person lying on the ground after having experienced a fall. It could also detect "other activities" with an accuracy of 99.09%. "Other activities" comprises all activities other than falls or fall-related motions that were present in this dataset. Figure 4 shows the confusion matrix that shows the performance accuracy of the RapidMiner process shown in Figure 2 when we imported the second dataset [51] using the dataset operator. It achieved an overall performance accuracy of 81.36%, with the class precision values for falls and other activities being 76.92% and 84.85%, respectively.

accuracy: 98.32%

	true other activity	true falling	true lying	true standing up from lying	true on all fours	class precision
pred. other activity	3593	16	3	10	4	99.09%
pred. falling	7	100	7	0	0	87.72%
pred. lying	7	7	2362	19	5	98.42%
pred. standing up from lying	14	0	12	765	0	96.71%
pred. on all fours	7	0	2	0	221	96.09%
class recall	99.04%	81.30%	98.99%	96.35%	96.09%	

Figure 3. Confusion matrix (tabular view) of the RapidMiner process shown in Figure 2, which is a k-NN-based multilabel machine learning approach for detecting falls and fall-related motions when the first dataset [50] was used.

accuracy: 81.36%

	true Not a Fall	true Fall	class precision
pred. Not a Fall	28	5	84.85%
pred. Fall	6	20	76.92%
class recall	82.35%	80.00%	

Figure 4. Confusion matrix (tabular view) of the RapidMiner process shown in Figure 2, which is a k-NN-based multilabel machine learning approach for detecting falls and other activities when the second dataset [51] was used.

After that, we performed a comprehensive comparison study of different machine learning-based methods to deduce the best machine learning method, in terms of performance accuracy, for the development of such fall detection systems and applications. This study is outlined in Section 4.

4. Comparative Study of Different Machine Learning Methods to Deduce the Best Machine Learning Method for Fall Detection Systems

As discussed in detail in Section 2, a range of machine learning approaches have been used by previous researchers in this field. These include random forest, artificial neural network, decision tree, support vector machine, k-NN, gradient boosted trees, naïve bayes, and deep learning [53]. Table 2 summarizes the works that have focused on implementing these machine learning approaches to develop fall detection systems.

Table 2. Summary of recent works in fall detection that used different kinds of machine learning approaches.

Learning Approach	Work (s)
Random forest	Balli et al. [32], Dhole et al. [33]
Artificial neural network	Espinosa et al. [30], Nakamura et al. [31], Tahir et al. [35]
Decision tree	Ozcan et al. [26], Lai et al. [27], Cai et al. [42]
Support vector machine	Bian et al. [25], Hakim et al. [28]
k-NN	Liu et al. [17]
Gradient boosted trees	Ning et al. [40], Cahoolessur et al. [41]
Deep learning	Feng et al. [23], Jácome et al. [36]
Naïve Bayes	Ngu et al. [38], Khan et al. [39]

Even though a range of machine learning approaches have been investigated by researchers in this field, there has not been any work done so far that compares the performance characteristics of these approaches to deduce the best learning method in terms of performance accuracy for the development of fall detection systems. Because of multiple reasons such as differences in the data source, variations in the preprocessing steps, different ratios of training and test data, and different methods of evaluating the performance characteristics, the performance characteristics of these existing works cannot be directly compared. There is a need for a comprehensive study that compares the performance characteristics of different machine learning approaches by using the same data source, the same data preprocessing steps, the same training-to-test ratios, and the same methods of evaluating the performance characteristics to deduce the best machine learning approach for the development of fall detection systems. We address this research challenge in this section by performing a comprehensive comparison study by developing, implementing, and testing the performance of different machine learning methods for fall detection. The machine learning methods that we developed, implemented, and tested for this study include random forest, artificial neural network, decision tree, multiway decision tree, support vector machine, k-NN, gradient boosted trees, ID3, decision stump, CHAID, AutoMLP, linear regression, vector linear regression, random tree, naïve bayes, naïve bayes (kernel), linear discriminant analysis, quadratic discriminant analysis, and deep learning [53]. We performed this study using both datasets [50,51]. The methodology is outlined in Section 4.1, and the results of the study are presented and discussed in Section 4.2.

4.1. Methodology for the Comparative Study of Different Machine Learning Approaches

The methodology outlined in this section is based on the system architecture shown in Figure 1. Therefore, a separate system architecture diagram is not provided in this section. The following are the steps associated with this study:

1. Use the dataset operator to import the datasets [50,51] into RapidMiner to develop a process where one dataset is used each time for a process.
2. Develop an operator that can track and analyze different postural orientations of the user by using Equation (1) and likewise calculate the orientation angles along the other two axes to detect falls and fall-related motions.
3. Develop an operator to preprocess the data, which consists of functionalities to (a) filter out and use the readings of the accelerometer data collected from the accelerometer placed on the user's chest and (b) remove any unnecessary attributes that are not necessary for the development of the classifier.
4. Use the "split data" operator to split the data into the training set and test set, with 80% data for training and the rest for testing.
5. Develop and implement a machine learning approach out of random forest, artificial neural network, decision tree, multiway decision tree, support vector machine, k-NN, gradient boosted trees, ID3, decision stump, CHAID, AutoMLP, linear regression, vector linear regression, random tree, naïve bayes, naïve bayes (kernel), linear discriminant analysis, quadratic discriminant analysis, and deep learning [53] each time for the development of this RapidMiner process. Separate RapidMiner processes were developed for each of these learning approaches.
6. Train the learning method using the training data and implement it on the test data by using the "apply model" operator.
7. Evaluate the performance characteristics of the learning approach in terms of overall accuracy and subclass precision values as per Equations (2) and (3) by using the performance operator.
8. Compare the findings of Step (7) for each of the learning methods mentioned in Step (5) to deduce the best machine learning approach in terms of accuracy.
9. Repeat Step 8 for both datasets to detect and track the consistency in the findings.

Even though the machine learning methods of multiway decision tree, ID3, decision stump, CHAID, AutoMLP, linear regression, vector linear regression, random tree, naïve bayes (kernel), linear discriminant analysis, and quadratic discriminant analysis [53] have not been very popular for the development of fall detection systems, we included these learning approaches in our study for the comprehensiveness of the same. The results and the comparison of the associated performance characteristics of these respective RapidMiner processes are presented and discussed in Section 4.2.

4.2. Results of the Comparative Study of Different Machine Learning Approaches

In this section, we present and discuss the results of this study, which we performed on both datasets [50,51]. A separate RapidMiner process was developed to implement each of the machine learning approaches as mentioned in Section 4.1, as per the system architecture shown in Figure 1, for each of the two datasets. The respective RapidMiner processes were similar to the RapidMiner process shown in Figure 2, with the primary difference being in the learning model operator that was used for each of these processes. Similar to Figure 3, which evaluates the performance characteristics of the RapidMiner process shown in Figure 2 via a confusion matrix when the first dataset [50] was used, we used the same approach to evaluate the performance characteristics of the respective RapidMiner processes that were developed to implement each of these learning methods on the two datasets. Table 3 summarizes the performance characteristics (both the overall accuracy and the respective class precision values) of the different learning methods that we developed to detect falls and fall-related motions by using the first dataset [50]. In this table, for any cell that shows 0.00% as the precision, it means that that specific class or subclass did not consist of any true detections in the test set by the specific learning method under consideration.

Table 3. Summary of the performance characteristics of the different machine learning methods that we developed in RapidMiner to detect falls and fall-related motions by using the first dataset [50].

Learning Approach	Overall Accuracy	Precision (Falling)	Precision (Lying)	Precision (Standing Up from Lying)	Precision (On All Fours)	Precision (Other Activity)
Random forest	81.97%	66.67%	80.17%	90.00%	100.00%	83.12%
Artificial neural network	86.33%	14.29%	87.24%	71.12%	75.29%	87.99%
Decision tree	81.87%	50.00%	80.51%	50.66%	76.92%	84.60%
Multiway decision tree	83.12%	0.00%	83.36%	63.34%	49.32%	85.90%
Support vector machine	76.50%	0.00%	67.34%	0.00%	0.00%	84.45%
k-NN	98.32%	87.72%	98.42%	96.71%	96.09%	99.09%
Gradient boosted trees	84.95%	0.00%	84.52%	67.20%	51.24%	88.61%
ID3	68.89%	0.00%	66.47%	66.47%	59.55%	70.39%
Decision stump	50.96%	0.00%	44.36%	0.00%	0.00%	51.20%
CHAID	68.87%	0.00%	66.44%	60.00%	59.55%	70.39%
AutoMLP	86.30%	30.43%	85.23%	70.72%	52.75%	91.45%
Linear regression	77.80%	0.00%	71.56%	0.00%	0.00%	82.58%
Vector linear regression	50.66%	0.00%	0.00%	0.00%	0.00%	50.66%
Random tree	77.29%	0.00%	76.04%	0.00%	0.00%	78.10%
Naïve Bayes	27.23%	5.69%	69.85%	12.95%	5.53%	0.00%
Naïve Bayes (kernel)	80.28%	4.00%	80.90%	47.87%	23.97%	85.08%
Linear discriminant analysis	50.66%	0.00%	0.00%	0.00%	0.00%	50.66%
Quadratic discriminant analysis	50.66%	0.00%	0.00%	0.00%	0.00%	50.66%
Deep learning	82.13%	33.33%	79.23%	56.75%	61.11%	86.61%

From Table 3, which shows the comparison of the performance accuracies of all the machine learning methods that we developed, implemented, and tested on the first dataset [50] for detecting falls and fall-like motions, the following can be concluded:

1. In order of decreasing overall performance accuracy, these machine learning methods can be arranged as k-NN > artificial neural network > AutoMLP > gradient boosted

trees > multiway decision tree > deep learning > random forest > decision tree > naïve bayes (kernel) > linear regression > random tree > support vector machine > ID3 > CHAID > decision stump > vector linear regression > linear discriminant analysis > quadratic discriminant analysis > naïve bayes.

2. In order of decreasing class precision value for the detection of a falling motion, these machine learning methods can be arranged as k-NN > random forest > decision tree > deep learning > AutoMLP > artificial neural network > naïve bayes > naïve bayes (kernel) > gradient boosted trees > multiway decision tree > linear regression > random tree > support vector machine > ID3 > CHAID > decision stump > vector linear regression > linear discriminant analysis > quadratic discriminant analysis.
3. In order of decreasing class precision value for the detection of a lying motion, these machine learning methods can be arranged as k-NN > artificial neural network > AutoMLP > gradient boosted trees > multiway decision tree > naïve bayes (kernel) > decision tree > random forest > deep learning > random tree > linear regression > naïve bayes > support vector machine > ID3 > CHAID > decision stump > vector linear regression > linear discriminant analysis > quadratic discriminant analysis.
4. In order of decreasing class precision value for the detection of standing up from a lying motion, these machine learning methods can be arranged as k-NN > random forest > artificial neural network > AutoMLP > gradient boosted trees > ID3 > multiway decision tree > CHAID > deep learning > decision tree > naïve bayes (kernel) > naïve bayes > random tree > linear regression > support vector machine > decision stump > vector linear regression > linear discriminant analysis > quadratic discriminant analysis.
5. In order of decreasing class precision value for the detection of being on all fours, these machine learning methods can be arranged as random forest > k-NN > decision tree > artificial neural network > deep learning > ID3 > CHAID > AutoMLP > gradient boosted trees > multiway decision tree > naïve bayes (kernel) > naïve bayes > random tree > linear regression > support vector machine > decision stump > vector linear regression > linear discriminant analysis > quadratic discriminant analysis.
6. In order of decreasing class precision value for the detection of other activities, these machine learning methods can be arranged as k-NN > AutoMLP > gradient boosted trees > artificial neural network > deep learning > multiway decision tree > naïve bayes (kernel) > decision tree > support vector machine > random forest > linear regression > random tree > ID3 > CHAID > decision stump > vector linear regression > linear discriminant analysis > quadratic discriminant analysis > naïve bayes.

From the above, it can be inferred that the k-NN learning approach outperformed all the other learning approaches in terms of overall accuracy, class precision value for the detection of falling, class precision value for the detection of lying, class precision value for the detection of standing up from a lying motion, and for the detection of other activities. For the detection of being on all fours, the random forest method achieved a higher performance accuracy. However, upon comparison of the respective accuracies of these two learning approaches for the detection of being on all fours, as presented in Table 3, it can be seen that the difference was less than 4%. In addition, for all the other motions and behavioral patterns, the random forest model's accuracy was much lower than that of the k-NN approach. Thus, based on the findings from this dataset [50], it may be concluded that the k-NN approach is the best machine learning method for the development of fall detection systems and related applications. To further uphold these findings, we repeated this study of comparing the performance accuracies of the different learning methods, as mentioned above [53], using the second dataset [51]. Table 4 summarizes the performance characteristics (both the overall accuracy and the respective class precision values) of these different learning methods that we developed to detect falls using the second dataset [51].

Table 4. Summary of the performance characteristics of the different machine learning methods that we developed in RapidMiner to detect falls and other activities using the second dataset [51].

Learning Approach	Overall Accuracy	Precision (Fall)	Precision (Not a Fall)
Random forest	71.19%	60.53%	90.48%
Artificial neural network	66.10%	66.67%	64.71%
Decision tree	74.58%	62.50%	100.00%
Multiway decision tree	74.58%	62.50%	100.00%
Support vector machine	66.10%	66.67%	64.71%
k-NN	81.36%	76.92%	84.85%
Gradient boosted trees	74.58%	67.86%	80.65%
ID3	57.63%	0.00%	57.63%
Decision stump	74.58%	62.50%	100.00%
CHAID	57.63%	0.00%	57.63%
AutoMLP	66.10%	64.71%	66.67%
Linear regression	66.10%	64.71%	66.67%
Vector linear regression	57.63%	0.00%	57.63%
Random tree	74.58%	62.50%	100.00%
Naïve Bayes	66.10%	64.71%	66.67%
Naïve Bayes (kernel)	71.19%	66.67%	74.29%
Linear discriminant analysis	66.10%	64.71%	66.67%
Quadratic discriminant analysis	74.58%	62.50%	100.00%
Deep learning	74.58%	62.50%	100.00%

From Table 4, which shows the comparison of the performance accuracies of all the machine learning methods that we developed, implemented, and tested on the second dataset [51] for detecting falls and other activities (labeled as "not a fall"), the following can be concluded:

1. In order of decreasing overall performance accuracy, these machine learning methods can be arranged as k-NN > decision tree > multiway decision tree > gradient boosted trees > decision stump > random tree > quadratic discriminant analysis > deep learning > random forest > naïve bayes (kernel) > artificial neural network > support vector machine > AutoMLP > linear regression > naïve bayes > linear discriminant analysis > ID3 > CHAID > vector linear regression.
2. In order of decreasing class precision value for the detection of falls, these machine learning methods can be arranged as k-NN > gradient boosted trees > naïve bayes (kernel) > artificial neural network > support vector machine > AutoMLP > linear regression > naïve bayes > linear discriminant analysis > decision tree > multiway decision tree > decision stump > random tree > quadratic discriminant analysis > deep learning > random forest > ID3 > CHAID > vector linear regression.
3. In order of decreasing class precision value for the detection of other activities (labelled as "not a fall" in the dataset), these machine learning methods can be arranged as decision tree = multiway decision tree = decision stump = random tree = quadratic discriminant analysis = deep learning > random forest > k-NN > gradient boosted trees > naïve bayes (kernel) > AutoMLP > linear regression > naïve bayes > linear discriminant analysis > artificial neural network > support vector machine > ID3 > CHAID > vector linear regression.

From the above, it can be inferred that the k-NN learning approach outperformed all the other learning approaches in terms of overall accuracy and in terms of the class precision value for the detection of falls. In terms of the class precision for the detection of other activities as present in the dataset [51], the k-NN approach ranked third compared to all the other learning methods, and its performance was not much less compared to the learning methods with a better class precision value for the detection of other activities. Thus, it can be concluded that the k-NN approach outperformed all the other learning approaches for the detection of falls and other activities when the second dataset [51] was

used. The above discussion of the results from the second dataset, as far as the performance of the k-NN learning method is concerned, is consistent with our findings from the first dataset [50]. Thus, based on the consistent findings from two different datasets [50,51] and the associated discussions, it can be concluded that the k-NN classifier is best suited for the development of fall detection systems and related applications, and it outperforms all the other machine learning methods.

5. Improving the Performance Accuracy of the k-NN Approach for Detection of Falls and Fall-Related Motions

It is essential that fall detection systems have high accuracy so that the underlining detections are perceived as accurate, and the systems are deemed reliable by the users of these systems, the caregivers, and the associated medical personnel. For the development of any machine learning classifier, it is important to reduce false positives and the overfitting of data so that the same does not impact the performance accuracy. Previous research studies [19–24] that focused on addressing false positives and studying behavioral patterns related to falls did not achieve high levels of performance accuracy. Thus, the challenge is to address the issue of false positives and overfitting of data while investigating approaches to improve the underlining system's performance accuracy. We present a novel approach for the same in this section, which is a combination of k-folds cross-validation with the AdaBoost algorithm. Here, the AdaBoost algorithm uses the k-NN learning approach with certain specifications that we defined in terms of how the classifier should work. In Section 5.1, we present the associated methodology, and the results from both datasets [50,51] are presented and discussed in Section 5.2.

5.1. Methodology for Improving the Performance Accuracy of the k-NN Approach for the Detection of Falls and Fall-Related Motions

This section outlines the methods and associated steps for improving the performance accuracy of the k-NN-based multilabel classification approach for the detection of falls and fall-related motions. The k-NN approach was selected for the development of this methodology because, as in Section 4, where we used two different datasets [50,51] to test our approach, we found that the k-NN classifier outperformed all other machine learning approaches to detect falls and fall-related motions as well as other activities. We developed this approach as a RapidMiner process. It is shown in Figure 2, and the performance characteristics of the same for the two datasets are shown in Figures 3 and 4, respectively. This process was developed by building on the system architecture shown in Figure 1. We improved this system (Figure 2) by introducing and implementing the k-folds cross-validation [54] approach and the AdaBoost algorithm [55,56] and incorporating certain specifications in the k-NN classifier in terms of how it should work, which are outlined next. Adaptive Boosting [55,56], abbreviated as AdaBoost, is a machine learning algorithm that can be combined with other learning approaches to improve or "boost" the performance of those respective classification models. The pseudo-code for the AdaBoost algorithm [55,56] is presented in Algorithm 1. The methodology also helps remove overfitting issues in machine learning-based classifications. Unlike other machine learning classifiers, the algorithm uses specific features from the training set that would improve the predictive performance of the specific classifier being developed. Such an approach also reduces dimensionality and contributes to improving the execution time, as features of the data that are not related to the classification problem at hand are not computed anymore. Upon application of AdaBoost to a machine learning classifier, such as k-NN, the boosted classifier can be represented as shown in Equations (4) and (5):

$$B_N(x) = \sum_{n=1}^{N} b_n(x) \tag{4}$$

$$D_t = \sum_{m} D(B_{n-1}(x_m) + \mathrm{a(n)h}(x_m)) \tag{5}$$

where:

$b_n(x)$ = machine learning classifier that uses (x) as input
h(x_m) = output hypothesis of the machine learning classifier
n = variable that represents each iteration
D_t = sum of training error of the final t-stage boosted classification model
$B_{n-1}(x_m)$ = the boosted classifier upon application of AdaBoost
$D(f)$ = error function of "f"
$b_n(x)$ = a(n)h(x_m) = the machine learning classifier that is being boosted

Algorithm 1 AdaBoost algorithm

Input: examples, set of N labeled examples (x1, y1), . . . ,(xN, yN)
Local Variables: w, a vector of N example weights, initially 1/N
h, a vector of K hypotheses
z, a vector of K hypothesis weights
Output: weighted-majority hypothesis
function AdaBoost (examples, L, K) returns a weighted-majority hypothesis
L, a learning algorithm
K, the number of hypotheses in the ensemble
1: **for** k = 1 to K, **do**
2: h[k] ← L(examples, w)
3: error ← 0
4: **for** j = 1 to N, do
5: **if** h[k](xj) ≠ yj **then** error ← error + w[j]
6: **for** j = 1 to N, **do**
7: **if** h[k](xj) = yj **then** w[j] ← w[j] · error/(1 − error)
8: w ← Normalize(w)
9: Z[k] ← log(1 − error)/error
10: **return** Weighted-Majority(h, z)

In a k-folds cross-validation approach [54], the dataset is randomly split into "k" number of subsets. For each iteration, a total of (k-1) number of subsets is taken to train the machine learning method, and it is tested on the remaining subset. The iteration continues for "k" number of times, hence the name "k-folds". The results from all these iterations are then averaged or combined to obtain the final performance characteristics of the machine learning method. This procedure is as follows:

i. Shuffle the dataset randomly.
ii. Split the dataset into k groups.
iii. For each unique group:
    a. Take the group as a holdout or test data set.
    b. Take the remaining groups as a training data set.
    c. Fit a model on the training set and evaluate it on the test set.
    d. Retain the evaluation score and discard the model.
iv. Summarize the skill of the model using the sample of model evaluation scores.

The primary advantage of this approach is that all data points are considered for both testing and training the learning method, and each data point is taken up for evaluation of the model only once. Several works in machine learning, pattern recognition, and their interrelated disciplines [57–60] have shown that the cross-validation approach helps to eliminate overfitting and reduces false positives. These works [57–60] justify the relevance of using the k-folds cross-validation approach in our learning method to minimize overfitting and false positives.

We developed and implemented the k-folds cross-validation and the AdaBoost algorithm in the RapidMiner process shown in Figure 2 as per the following steps. For these steps, the specific number of iterations for the AdaBoost, the specific number of folds for the cross-validation operator, and the specifications of the k-NN learning method were

determined by implementing all the available training and testing methods and exploring different steps, followed by finalizing the specific steps that yielded the best performance.

1. Disable or delete all the other operators in the process (Figure 2) other than the dataset, posture recognition, and data preprocess operators.
2. Import the cross-validation operator into the process.
3. Customize the functionality of the cross-validation operator to define the number of folds and sampling type.
4. Customize the training subprocess of the cross-validation operator.
    a. The cross-validation operator has 10 folds.
    b. Import the AdaBoost operator into the subprocess with the AdaBoost consisting of 10 iterations.
    c. Customize the AdaBoost operator by defining the number of iterations.
    d. Define the AdaBoost operator's work by developing the process that it must iterate.
    e. The definition in (d) above consists of developing the RapidMiner process shown in Figure 2, in partial form, using only the dataset, posture recognition, data preprocess, and k-NN operators.
5. Add specifications to the k-NN learning method.
    a. The value of "k" = the number of labels in the attribute being predicted from the test data.
    b. Use the weighted vote approach.
    c. The measure type is numerical measures, with the specific numerical measure being Camberra Distance.
6. Customize the testing subprocess of the cross-validation operator.
    a. Import the "apply model" operator to apply the learning method.
    b. Include the performance operator and customize the same so that the performance is evaluated using a confusion matrix that shows the overall accuracy and the respective class precision values.
    c. Update the functionality of the performance operator to use example weights.

5.2. Results of the Methodology for Improving the Performance Accuracy of the k-NN Approach for the Detection of Falls and Fall-Related Motions

To implement the above approach, we used 10 folds for the k-folds cross-validation process (Step 3) and used 10 iterations for the AdaBoost operator (Step 4b). When the first dataset [50] was used, the weighted k-NN model contained 35,801 examples with seven dimensions of the classes—falling, lying, standing up from lying, being on all fours, and other activity. When the second dataset [51] was used, the weighted k-NN model contained 295 examples with three dimensions of the following classes: "fall" and "not a fall". The process and its associated subprocesses that we developed in RapidMiner by following the above steps are shown in Figures 5 and 6.

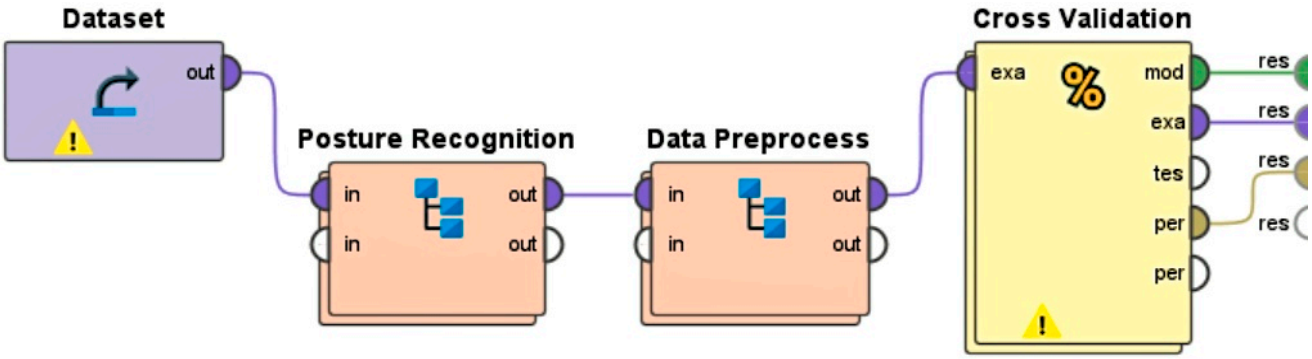

Figure 5. The process that we developed in RapidMiner using the cross-validation approach and the AdaBoost algorithm. The AdaBoost algorithm was a subprocess of the cross-validation operator, as shown in Figure 6a.

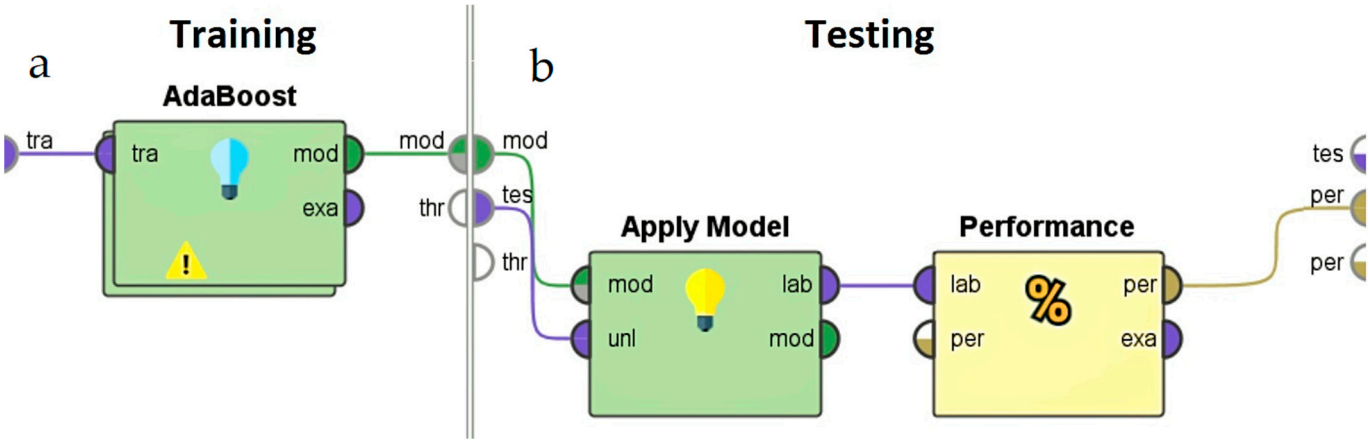

Figure 6. (**a**) The training and testing modules of the cross-validation operator from the process shown in Figure 5. (**b**) The iterative subprocess that was developed by customizing the AdaBoost operator is shown in Figure 6a.

The performance characteristics of this approach were studied and analyzed using a confusion matrix to determine the overall performance accuracy and the subclass precision values for detection of falls and fall-related motions—falling, lying, being on all fours, and standing up from lying for the first dataset [50]. The confusion matrix that outlined the performance accuracy of our approach when tested on the first dataset [50] is presented in tabular form in Figure 7.

accuracy: 99.87% +/- 0.08% (micro average: 99.87%)

	true other activity	true falling	true lying	true standing up from lying	true on all fours	class precision
pred. other activity	18123	8	0	6	0	99.92%
pred. falling	4	602	2	0	0	99.01%
pred. lying	2	4	11919	4	0	99.92%
pred. standing up from lying	8	0	8	3958	0	99.60%
pred. on all fours	1	0	1	0	1151	99.83%
class recall	99.92%	98.05%	99.91%	99.75%	100.00%	

Figure 7. Confusion matrix representing the overall performance accuracy as well as the subclass precision values for the different activities—falling, lying, standing up from lying, being on all fours, and other activities, of the RapidMiner process shown in Figure 5 when the first dataset [50] was used.

As shown in Figure 7, the use of the k-folds cross-validation approach with the AdaBoost algorithm boosted the performance accuracy of the k-NN-based learning approach considerably. The improved overall performance accuracy of this approach was now 99.87%, with the subclass precision values for the activities of falling, lying, standing up from lying, being on all fours, and other activities being 99.01%, 99.92%, 99.60%, 99.83%, and 99.92%, respectively. The confusion matrix that outlined the performance accuracy of our approach when tested on the second dataset [51] is presented in tabular form in Figure 8. As can be seen from Figure 8, the use of the k-folds cross-validation approach with the AdaBoost algorithm boosted the performance accuracy of the k-NN based learning approach considerably yet again. The improved overall performance accuracy of this approach was now 99.66%, with the subclass precision values for falls and other activities being 100% and 99.42%, respectively.

accuracy: 99.66% +/- 1.09% (micro average: 99.66%)

	true Not a Fall	true Fall	class precision
pred. Not a Fall	172	1	99.42%
pred. Fall	0	122	100.00%
class recall	100.00%	99.19%	

Figure 8. Confusion matrix representing the overall performance accuracy as well as the subclass precision values for the different activities, including falls, of the RapidMiner process shown in Figure 5 when the second dataset [51] was used.

Next, to further discuss the relevance and significance of these results, we compared these performance accuracies to similar works [17–43] in this field, which we reviewed in Section 2. The performance characteristics of these works [17–43] are summarized in Table 5. Table 5 also shows the comparison of the performance accuracies of our learning approaches, shown in Figures 7 and 8, with the performance accuracies of the existing approaches [17–43].

Table 5. Summary of recent works in fall detection along with their performance accuracies.

Work(s)	Performance Accuracy
Liu et al. [17]	84.44%
Sun et al. [18]	85.40%
Rafferty et al. [19]	68.00%
Castillo et al. [20]	79.57%
Fu et al. [21]	84.00%
Williams et al. [22]	80.00%
Feng et al. [23]	81.70%
Jokanovic et al. [24]	83.00%
Bian et al. [25]	97.60%
Ozcan et al. [26]	89.80%
Lai et al. [27]	92.92%
Hakim et al. [28]	90.00%
Tomii et al. [29]	93.30%
Espinosa et al. [30]	95.64%
Nakamura et al. [31]	90.00%
Balli et al. [32]	98.50%
Dhole et al. [33]	98.00%
Ramirez et al. [34]	98.84%
Ahmad et al. [35]	92.33%
Jácome et al. [36]	98.75%
Dhiraj et al. [37]	90.00%
Ngu et al. [38]	92.33%
Khan et al. [39]	95.00%
Ning et al. [40]	96.00%
Cahoolessur et al. [41]	96.00%
Cai et al. [42]	99.08%
Lee et al. [43]	96.00%
Thakur et al. [this work]—dataset 1	99.87%
Thakur et al. [this work]—dataset 2	99.66%

Upon arranging these performance accuracies in decreasing order, the following can be observed:

Thakur et al. [this work]—dataset 1 > Thakur et al. [this work]—dataset 2 > Cai et al. [42] > Ramirez et al. [34] > Jácome et al. [36] > Balli et al. [32] > Dhole et al. [33] > Bian et al. [25]

> Ning et al. [40] > Cahoolessur et al. [41] > Lee et al. [43] > Espinosa et al. [30] > Khan et al. [39] > Tomii et al. [29] > Lai et al. [27] > Tahir et al. [35] > Ngu et al. [38] > Hakim et al. [28] > Nakamura et al. [31] > Dhiraj et al. [37] > Ozcan et al. [26] > Sun et al. [18] > Liu et al. [17] > Fu et al. [21] > Jokanovic et al. [24] > Feng et al. [23] > Williams et al. [22] > Castillo et al. [20] > Rafferty et al. [19].

As per the above, it can be concluded that our work outperformed all similar works in this field [17–43] in terms of the overall performance accuracy. Our work used a methodology based on the k-fold cross-validation method and the AdaBoost algorithm applied to a k-NN-based multilabel classifier that minimizes false positives and reduces overfitting while detecting falls. The fact that when we tested our approach using two different datasets, we achieved performance accuracies higher than any of the prior works in this field upholds the potential, superiority, and relevance of this methodology. In addition to the above, our approach can also detect fall-related motions and overcomes the limitations of binary classification that have been previously used in this field. Furthermore, our methodology can detect the activity of "standing up from lying" with high accuracy. This serves as a means to infer whether a person who has experienced a fall also experienced a long lie. This can be done in real-time by tracking whether the "fall" event was followed by a "standing up from lying" event within a timeframe of one hour.

6. Comparative Discussion

There have been several works [16–43] published over the last few years with the aim of detecting falls in the elderly to contribute to their assisted living and healthy aging in the future of technology-based pervasive living spaces, such as smart homes. Researchers have investigated multiple approaches to address this challenge. Despite several advances and multiple emerging technologies related to fall detection, these works have several limitations and drawbacks, as outlined in Section 2. In this section, we further elaborate on these challenges and the limitations of the existing systems while discussing how our work addresses the same. This is outlined as follows:

1. There is need for a study that deduces the optimal machine learning approach for fall detection: A range of machine learning methods such as random forest [32,33], artificial neural network [30,31,35], decision tree [26,27,42], support vector machine [25,28], k-NN [17], gradient boosted trees [40,41], naïve bayes [38,39], and deep learning [23,36] have been used by researchers in the last few years (Table 2). However, no study has been conducted in this field thus far that compares the performance characteristics of these machine learning approaches for fall detection to deduce the best learning approach for the same. Moreover, no prior work in this field (Tables 1 and 2) has been associated with the development, implementation, and evaluation of such a wide range of machine learning-based fall detection systems. This paper takes a comprehensive approach to address this challenge. We conducted a comprehensive study of 19 machine learning approaches for fall detection, which included random forest, artificial neural network, decision tree, multiway decision tree, support vector machine, k-NN, gradient boosted trees, ID3, decision stump, CHAID, AutoMLP, linear regression, vector linear regression, random tree, naïve bayes, naïve bayes (kernel), linear discriminant analysis, quadratic discriminant analysis, and deep learning. The approach was tested on two different datasets [50,51] and the results (Tables 3 and 4) were consistent from the two datasets. The results show that the k-NN based-machine learning method outperformed all the other machine learning methods for the detection of falls and fall-like motions.
2. Most of the recent works in this field, as discussed in Section 2, have been binary classifiers. Such classifiers tend to classify certain fall-like postures, such as being on all fours, as a fall. Such false positives can lead to alert fatigue [44] in caregivers or medical personnel, thereby causing a decline in the quality of care. It is highly necessary that machine learning-based fall detection systems detect falls and accurately detect fall-related motions or similar postures to address this limitation. We have addressed

this challenge in this work by proposing an interdisciplinary framework that can recognize postures (Figure 1) and interpret multimodal components of posture and motion-related data (Section 3.1) to train a k-NN-based multilabel machine learning method that can detect falls as well as fall-like motions (Figure 2) by tracking the data coming from an accelerometer placed on the user's chest, which is the optimal position for an accelerometer to track motions and behavioral data. We tested our approach on a dataset [50] that consisted of 164,860 rows of data collected from five individuals performing different activities, including falls. The results (Figure 3) of this k-NN-based multilabel classifier show that our proposed approach can detect not only falls but also fall-related motions with a high level of accuracy.

3. To address the challenge that fall detections must achieve much higher accuracies for increased trust and user acceptance, we have presented a novel approach (Section 5) to improve the k-NN-based multilabel classifier's performance for fall detection. We presented this approach for the k-NN-based classifier because the findings of Section 4 (Tables 3 and 4) show that out of all the machine learning methods, the k-NN approach is best suited for the development of fall detection systems and applications. The methodology for the development of this approach that uses k-folds cross-validation [54] and the AdaBoost algorithm [55,56] with a k-NN-based multilabel classifier with certain defined specifications is presented in Section 5.1. We evaluated this approach on two datasets, and the results presented and discussed in the form of a comparison study (Table 5) show that irrespective of the dataset, this approach for fall detection boosts the performance accuracy of the k-NN classifier and outperforms all prior works [17–43], thereby upholding the potential, relevance, and importance of this methodology.
4. To address the need for long lie detection after a fall, which can have minor to major health-related concerns for the elderly, we have proposed a k-NN-based posture and motion recognition approach that can detect the activity of standing up from lying. We tested this approach on a dataset [50], and the results (Figure 3) show that our approach achieves class precision and class recall values of 96.71% and 96.35%, respectively, for detecting the motion of standing up from lying. In addition to this, in Section 5, we have presented a novel methodology to improve the performance accuracy of our learning approach by use of the cross-validation approach [54] and AdaBoost algorithm [55,56]. The results (Figure 7) of this approach show that the class precision and class recall values for the motion of standing up from lying increased to 99.60% and 99.75%, respectively. To implement this approach in a real-time scenario, the temporal information would need to be tracked to detect whether the activity of standing up from lying took place within one hour of the person experiencing a fall.

Based on the above novel functionalities of our approach, which are supported by the results from two datasets [50,51], we expect that our approach would be able to detect falls due to to bodily impairments or limitations environmental factors, and/or other health-related concerns. One of the limitations of our work is that we used the datasets developed by Kaluža et al. [50] and Tabbakha et al. [51], on account of not being able to develop our own dataset due to the work-from-home guidelines in the United States on account of COVID-19 [49]. Therefore, upon relaxation of these guidelines in the United States [49], we plan on developing our own dataset by collecting motion-related data, including falls, from both healthy individuals and elderly people, as per IRB-approved protocols, to test and validate our approach in real-time.

7. Conclusions and Scope for Future Work

Falls are a critical issue in the constantly increasing aging population of the world. Falls are associated with multiple behavioral, social, emotional, mental, psychological, and health-related impacts to the elderly and can even lead to death. In recent times, the unprecedented increase of falls in the elderly population of the world, especially in the United States, has caused a huge burden on the world economy. Due to the acute

shortage of caregivers, it is the need of the hour that technology-based pervasive living environments, such as smart homes, be equipped with accurate, reliable, trustworthy, and robust systems and applications that can detect falls to contribute to healthy aging and improved quality of life of the elderly. In the work presented in this paper, we conducted a comprehensive review of recent works in the field of fall detection and identified multiple research challenges that still exist. Thereafter, we explored and integrated recent advances from the fields of human–computer interaction, artificial intelligence, machine learning, internet of things, pattern recognition, and pervasive computing to address these challenges. The work presented in this paper makes four scientific contributions to the field of fall detection and its interrelated disciplines.

First, to address the need for an optimal machine learning approach for the development of fall detection systems and applications, we performed a comprehensive study where 19 different machine learning-based fall detection systems were developed, implemented, and evaluated on two different datasets. The results of the study compared in terms of overall accuracy and subclass precision values show that the k-NN approach is best suited for the development of fall detection systems and related applications. Second, to address the challenge that most works in this field have been binary classifiers that have the limitation of classifying fall-related motions as falls, a novel posture recognition-based multilabel machine learning method was proposed in this paper that can detect falls and fall-related motions with high accuracy.

Third, we proposed a novel machine learning and pattern recognition-based methodology that uses the k-folds cross-validation and AdaBoost approaches applied to the k-NN learning method to address the need to develop highly accurate fall detection systems while minimizing false positives and reducing overfitting of data. The results presented and discussed show that such a methodology increased the k-NN-based multilabel classifier's performance to 99.87% for the first dataset and to 99.66% for the second dataset. Such high-performance accuracies have never been achieved before in any of the prior works in this field. Finally, our approach also consists of the methodology to accurately detect long lies, which are common after falls in the elderly and can have several health-related impacts.

As per the authors' best knowledge, no similar work has been done in this field so far. With the rapid advancement and urbanization in all parts of the world, more people, specifically the elderly, will start living in sensor-driven and IoT-based interconnected smart homes. The work presented in this paper can be of paramount importance in such smart homes to ensure that the future of intelligent living spaces can track, study, analyze, and detect multiple healthcare-related needs of the elderly. The results that show the reliability, high accuracy, trustworthiness, and robustness of our approach uphold the immense potential and relevance of this work to contribute to assisted living and healthy aging of the elderly in artificial intelligence-based living environments of the future that would consist of sensor and actuator networks working in tandem. In the future, we plan on conducting experiments with human subjects with IRB-approved protocols to evaluate the efficacy of this approach in real-time IoT-based pervasive living and working environments. As per the proposed system architecture, we plan on using the MetaMotion C wearable sensors developed by Mbient Labs [47], that provide triaxial acceleration data as well as other dynamics of a user's motion data, which can be suitably and seamlessly interpreted and integrated into our real-time data collection framework [48] for the real-time development and deployment of our approach. We will be designing the experiments in such a way that the safety of the human subjects is considered the top priority, as per the guidelines of IRB [61]. The robust system architecture of our framework, as presented and discussed in this paper, is expected to allow for the seamless development and implementation of the same in a myriad of wearable devices, products, and smart gadgets by both developers and/or researchers in the near future. However, as IRB guidelines [61] might not be applicable to some of these applications and scenarios, where our framework could be used, it is important that these real-world applications of our framework be developed in a way such that the safety of any possible category of user [62] is highly prioritized.

The authors recommend that such applications of our framework be developed as per the guidelines mentioned in ISO 14971:2019 [63] and as per the human-centered risk assessment (RA) and risk management (RM) methodologies mentioned in [64]. These guidelines [63] and the RA and RM methodologies [64] should be followed both at the design stage and during the post-market surveillance in such applications of our framework to minimize risks and to ensure the safety of the user.

Author Contributions: Conceptualization, N.T. and C.Y.H.; methodology, N.T.; software, N.T.; validation, N.T.; formal analysis, N.T.; investigation, N.T.; resources, N.T.; data curation, N.T.; visualization, N.T.; data analysis and results, N.T.; writing—original draft preparation, N.T.; writing—review and editing, N.T. and C.Y.H.; supervision, C.Y.H.; project administration, C.Y.H.; funding acquisition, not applicable. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found at https://www.kaggle.com/uciml/posture-reconstruction and https://data.mendeley.com/datasets/sy3kcttdtx/3, accessed on 13 February 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. World Population Ageing 2019. Available online: https://www.un.org/en/development/desa/population/publications/pdf/ageing/WorldPopulationAgeing2019-Highlights.pdf (accessed on 4 April 2021).
2. World Population Ageing: 1950–2050. Available online: http://globalag.igc.org/ruralaging/world/ageingo.htm (accessed on 4 April 2021).
3. Nahian, M.J.A.; Ghosh, T.; Banna, M.H.A.; Aseeri, M.A.; Uddin, M.N.; Ahmed, M.R.; Mahmud, M.; Kaiser, M.S. Towards an Accelerometer-Based Elderly Fall Detection System Using Cross-Disciplinary Time Series Features. *IEEE Access* **2021**, *9*, 39413–39431. [CrossRef]
4. Thakur, N.; Han, C.Y. An Improved Approach for Complex Activity Recognition in Smart Homes. In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2019; pp. 220–231.
5. Tinetti, M.E. Clinical Practice. Preventing Falls in Elderly Persons. *N. Engl. J. Med.* **2003**, *348*, 42–49. [CrossRef]
6. Wang, X.; Ellul, J.; Azzopardi, G. Elderly Fall Detection Systems: A Literature Survey. *Front. Robot. AI* **2020**, *7*, 71. [CrossRef]
7. Lezzar, F.; Benmerzoug, D.; Kitouni, I. Camera-Based Fall Detection System for the Elderly with Occlusion Recognition. *Appl. Med. Inform.* **2020**, *42*, 169–179.
8. CDC. Keep on Your Feet—Preventing Older Adult Falls. Available online: https://www.cdc.gov/injury/features/older-adult-falls/index.html (accessed on 4 April 2021).
9. The National Council on Aging. Available online: https://www.ncoa.org/news/resources-for-reporters/get-the-facts/falls-prevention-facts/ (accessed on 4 April 2021).
10. CDC Important Facts about Falls. Available online: https://www.cdc.gov/homeandrecreationalsafety/falls/adultfalls.html (accessed on 4 April 2021).
11. Cost of Older Adult Falls. Available online: https://www.cdc.gov/homeandrecreationalsafety/falls/data/fallcost.html (accessed on 4 April 2021).
12. El-Bendary, N.; Tan, Q.; Pivot, F.C.; Lam, A. Fall Detection and Prevention for the Elderly: A Review of Trends and Challenges. *Int. J. Smart Sens. Intell. Syst.* **2013**, *6*, 1230–1266. [CrossRef]
13. Long Lie. Available online: https://www.physio-pedia.com/Long_Lie (accessed on 5 April 2021).
14. Tinetti, M.E.; Liu, W.L.; Claus, E.B. Predictors and Prognosis of Inability to Get Up After Falls Among Elderly Persons. *JAMA* **1993**, *269*, 65–70. [CrossRef]
15. Wild, D.; Nayak, U.S.; Isaacs, B. How Dangerous Are Falls in Old People at Home? *Br. Med. J. (Clin. Res. Ed)* **1981**, *282*, 266–268. [CrossRef]
16. Nick, M.; Becker, M. A Hybrid Approach to Intelligent Living Assistance. In Proceedings of the 7th International Conference on Hybrid Intelligent Systems (HIS 2007), Kaiserslautern, Germany, 17–19 September 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 283–289.
17. Liu, C.-L.; Lee, C.-H.; Lin, P.-M. A fall detection system using k-nearest neighbor classifier. *Expert Syst. Appl.* **2010**, *37*, 7174–7181. [CrossRef]

18. Sun, J.; Wang, Z.; Chen, L.; Wang, B.; Ji, C.; Tao, S. A Plantar Inclinometer Based Approach to Fall Detection in Open Environments. In *Emerging Trends and Advanced Technologies for Computational Intelligence*; Springer International Publishing: Cham, Switzerland, 2016; pp. 1–13.
19. Rafferty, J.; Synnott, J.; Nugent, C.; Morrison, G.; Tamburini, E. Fall Detection through Thermal Vision Sensing. In *Ubiquitous Computing and Ambient Intelligence*; Springer International Publishing: Cham, Switzerland, 2016; pp. 84–90.
20. Castillo, J.C.; Carneiro, D.; Serrano-Cuerda, J.; Novais, P.; Fernández-Caballero, A.; Neves, J. A multi-modal approach for activity classification and fall detection. *Int. J. Syst. Sci.* **2014**, *45*, 810–824. [CrossRef]
21. Fu, Z.; Delbruck, T.; Lichtsteiner, P.; Culurciello, E. An Address-Event Fall Detector for Assisted Living Applications. *IEEE Trans. Biomed. Circuits Syst.* **2008**, *2*, 88–96. [CrossRef]
22. Willems, J.; Debard, G.; Vanrumste, B.; Goedemé, T. A Video-Based Algorithm for Elderly Fall Detection. In *IFMBE Proceedings*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 312–315.
23. Feng, P.; Yu, M.; Naqvi, S.M.; Chambers, J.A. Deep Learning for Posture Analysis in Fall Detection. In *2014 19th International Conference on Digital Signal Processing*; IEEE: Piscataway, NJ, USA, 2014; pp. 12–17.
24. Jokanovic, B.; Amin, M.; Ahmad, F.; Boashash, B. Radar Fall Detection Using Principal Component Analysis. In *Radar Sensor Technology XX*; Ranney, K.I., Doerry, A., Eds.; SPIE: Bellingham, WA, USA, 2016.
25. Bian, Z.-P.; Hou, J.; Chau, L.-P.; Magnenat-Thalmann, N. Fall Detection Based on Body Part Tracking Using a Depth Camera. *IEEE J. Biomed. Health Inform.* **2015**, *19*, 430–439. [CrossRef]
26. Ozcan, K.; Velipasalar, S.; Varshney, P.K. Autonomous Fall Detection with Wearable Cameras by Using Relative Entropy Distance Measure. *IEEE Trans. Hum. Mach. Syst.* **2016**, *47*, 1–9. [CrossRef]
27. Lai, C.-F.; Chang, S.-Y.; Chao, H.-C.; Huang, Y.-M. Detection of Cognitive Injured Body Region Using Multiple Triaxial Accelerometers for Elderly Falling. *IEEE Sens. J.* **2011**, *11*, 763–770. [CrossRef]
28. Hakim, A.; Huq, M.S.; Shanta, S.; Ibrahim, B.S.K.K. Smartphone Based Data Mining for Fall Detection: Analysis and Design. *Procedia Comput. Sci.* **2017**, *105*, 46–51. [CrossRef]
29. Tomii, S.; Ohtsuki, T. Falling Detection Using Multiple Doppler Sensors. In *2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*; IEEE: Piscataway, NJ, USA, 2012; pp. 196–201.
30. Espinosa, R.; Ponce, H.; Gutiérrez, S.; Martínez-Villaseñor, L.; Brieva, J.; Moya-Albor, E. A vision-based approach for fall detection using multiple cameras and convolutional neural networks: A case study using the UP-Fall detection dataset. *Comput. Biol. Med.* **2019**, *115*, 103520. [CrossRef]
31. Nakamura, T.; Bouazizi, M.; Yamamoto, K.; Ohtsuki, T. Wi-Fi-CSI-Based Fall Detection by Spectrogram Analysis with CNN. In *GLOBECOM 2020—2020 IEEE Global Communications Conference*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
32. Balli, S.; Sağbaş, E.A.; Peker, M. Human activity recognition from smart watch sensor data using a hybrid of principal component analysis and random forest algorithm. *Meas. Control* **2019**, *52*, 37–45. [CrossRef]
33. Dhole, S.R.; Kashyap, A.; Dangwal, A.N.; Mohan, R. A novel helmet design and implementation for drowsiness and fall detection of workers on-site using EEG and Random-Forest Classifier. *Procedia Comput. Sci.* **2019**, *151*, 947–952. [CrossRef]
34. Ramirez, H.; Velastin, S.A.; Fabregas, E.; Meza, I.; Makris, D.; Farias, G. Fall Detection using Human Skeleton Features. In Proceedings of the 11th International Conference on Pattern Recognition Systems—ICPRS-21, Curico, Chile, 17–19 March 2021, in press.
35. Tahir, A.; Ahmad, J.; Morison, G.; Larijani, H.; Gibson, R.M.; Skelton, D.A. Hrnn4f: Hybrid Deep Random Neural Network for Multi-Channel Fall Activity Detection. *Probab. Eng. Inf. Sci.* **2021**, *35*, 37–50. [CrossRef]
36. Sarabia-Jácome, D.; Usach, R.; Palau, C.E.; Esteve, M. Highly-efficient fog-based deep learning AAL fall detection system. *Internet Things* **2020**, *11*, 100185. [CrossRef]
37. Dhiraj; Manekar, R.; Saurav, S.; Maiti, S.; Singh, S.; Chaudhury, S.; Neeraj; Kumar, R.; Chaudhary, K. Activity Recognition for Indoor Fall Detection in 360-Degree Videos Using Deep Learning Techniques. In *Proceedings of the 3rd International Conference on Computer Vision and Image Processing*; Springer: Singapore, 2020; pp. 417–429.
38. Ngu, A.H.; Tseng, P.-T.; Paliwal, M.; Carpenter, C.; Stipe, W. Smartwatch-Based IoT Fall Detection Application. *Open J. Internet Things* **2018**, *4*, 87–98.
39. Khan, S.; Qamar, R.; Zaheen, R.; Al-Ali, A.R.; Al Nabulsi, A.; Al-Nashash, H. Internet of things based multi-sensor patient fall detection system. *Healthc. Technol. Lett.* **2019**, *6*, 132–137. [CrossRef]
40. Ning, Y.; Zhang, S.; Nie, X.; Li, G.; Zhao, G. Fall Detection Algorithm Based on Gradient Boosting Decision Tree. In *2019 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–4.
41. Cahoolessur, D.K.; Rajkumarsingh, B. Fall Detection System using XGBoost and IoT. *R&D J.* **2020**, *36*, 8–18. [CrossRef]
42. Cai, W.; Qiu, L.; Li, W.; Yu, J.; Wang, L. Practical Fall Detection Algorithm based on Adaboost. In *Proceedings of the 2019 4th International Conference on Biomedical Signal and Image Processing (ICBIP 2019)—ICBIP '19*; ACM Press: New York, NY, USA, 2019.
43. Lee, J.-S.; Tseng, H.-H. Development of an Enhanced Threshold-Based Fall Detection System Using Smartphones with Built-In Accelerometers. *IEEE Sens. J.* **2019**, *19*, 8293–8302. [CrossRef]
44. Cash, J.J. Alert fatigue. *Am. J. Health Syst. Pharm.* **2009**, *66*, 2098–2101. [CrossRef] [PubMed]
45. Gjoreski, H.; Lustrek, M.; Gams, M. Accelerometer Placement for Posture Recognition and Fall Detection. In *2011 Seventh International Conference on Intelligent Environments*; IEEE: Piscataway, NJ, USA, 2011.
46. Shaeffer, D.K. MEMS inertial sensors: A tutorial overview. *IEEE Commun. Mag.* **2013**, *51*, 100–109. [CrossRef]

47. Mbient Labs Metamotion C. Available online: https://mbientlab.com/metamotionc/ (accessed on 5 April 2021).
48. Chakraborty, S.; Han, C.Y.; Zhou, X.; Wee, W. A Context Driven Human Activity Recognition Framework. In Proceedings of the 2016 International Conference on Health Informatics and Medical Systems, Monte Carlo Resort, Las Vegas, NV, USA, 25–28 July 2016; pp. 96–102.
49. Axelrod, B. Ohio Gov. Mike DeWine Asks Employers to Continue Working Remotely Amid COVID-19. Available online: https://www.wkyc.com/article/news/health/coronavirus/dewine-woking-remotely-covid-19/95-54447569-e757-4eac-bbc3-4bab4652b764 (accessed on 5 April 2021).
50. Kaluža, B.; Mirchevska, V.; Dovgan, E.; Luštrek, M.; Gams, M. An Agent-Based Approach to Care in Independent Living. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 177–186.
51. Tabbakha, N.E.; Ooi, C.P.; Tan, W.H. A Dataset for Elderly Action Recognition Using Indoor Location and Activity Tracking Data. *Mendeley Data*. 2020. Available online: https://data.mendeley.com/datasets/sy3kcttdtx/3 (accessed on 13 February 2021).
52. Mierswa, I.; Wurst, M.; Klinkenberg, R.; Scholz, M.; Euler, T. YALE: Rapid Prototyping for Complex Data Mining Tasks. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD'06, Philadelphia, PA, USA, 20–23 August 2006; ACM Press: New York, NY, USA, 2006.
53. Mohammed, M.; Khan, M.B.; Bashier, E.B.M. *Machine Learning: Algorithms and Applications*; CRC Publishers: Boca Raton, FL, USA, 2016. [CrossRef]
54. Browne, M.W. Cross-Validation Methods. *J. Math. Psychol.* **2000**, *44*, 108–132. [CrossRef] [PubMed]
55. Schapire, R.E. Explaining AdaBoost. In *Empirical Inference*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 37–52.
56. Wikipedia Contributors. AdaBoost. Available online: https://en.wikipedia.org/w/index.php?title=AdaBoost&oldid=1015653726 (accessed on 5 April 2021).
57. Santos, M.S.; Soares, J.P.; Abreu, P.H.; Araujo, H.; Santos, J. Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches [Research Frontier]. *IEEE Comput. Intell. Mag.* **2018**, *13*, 59–76. [CrossRef]
58. Subramanian, J.; Simon, R. Overfitting in Prediction Models—Is It a Problem Only in High Dimensions? *Contemp. Clin. Trials* **2013**, *36*, 636–641. [CrossRef] [PubMed]
59. Jia, Z. Controlling the Overfitting of Heritability in Genomic Selection through Cross Validation. *Sci. Rep.* **2017**, *7*, 13678. [CrossRef] [PubMed]
60. Lim, C.; Yu, B. Estimation Stability with Cross-Validation (ESCV). *J. Comput. Graph. Stat.* **2016**, *25*, 464–492. [CrossRef]
61. U.S. Department of Health and Human Services; Office for Human Research Protections (OHRP); U.S. Department of Health and Human Services; Food and Drug Administration; Office of Good Clinical Practice (OGCP); Center for Drug Evaluation and Research (CDER); Center for Biologics Evaluation and Research (CBER); Center for Devices and Radiological Health (CDRH); Office of Regulatory Affairs (ORA). Institutional Review Board (IRB) Written Procedures: Guidance for Institutions and IRBs, Fda.gov. 2018. Available online: https://www.fda.gov/media/99271/download (accessed on 19 June 2021).
62. U.S. Food & Drug Administration. CFR—Code of Federal Regulations Title 21. 2020. Available online: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820&showFR=1 (accessed on 19 June 2021).
63. ISO 14971:2019, Medical Devices—Application of Risk Management to Medical Devices, Iso.org. 2019. Available online: https://www.iso.org/standard/72704.html (accessed on 19 June 2021).
64. Coronato, A.; Cuzzocrea, A. An Innovative Risk Assessment Methodology for Medical Information Systems. *IEEE Trans. Knowl. Data Eng.* **2020**. [CrossRef]