Fall Detection System using XGBoost and IoT

DK Cahoolessur[a], B Rajkumarsingh[b]

Received 30 October 2019, in revised form 26 Nov 2019 and accepted 12 June 2020

Abstract: This project aims to design and implement a fall detection system for the elders using machine learning techniques and Internet-of-Things (IoT). The main issue with fall detection systems is false alarms and hence incorporating machine learning in the fall detection algorithm can tackle this problem. Therefore, choosing the right machine learning algorithm for the given problem is essential and several factors need to be considered in making that choice. For this project, the XGBoost algorithm is used and the machine learning model is trained on the Sisfall dataset. A wearable device that is worn on the waist is designed using an accelerometer, a microcontroller, a Global Positioning System (GPS) module and a buzzer. The acceleration data obtained is converted into features and fed into the machine learning model which will then make a prediction. If a fall event has occurred, the buzzer is activated and emergency contacts of the victim are notified immediately using IoT and Global System for Mobile Communications (GSM). This allows the fall victim to be attended quickly, thus reducing the negative consequences of the fall. The details of the fall are stored on the cloud so that they can be easily accessed by healthcare professionals. Testing the system concluded that the XGBoost machine learning algorithm is well suited for this problem due to the small percentage error obtained.

Additional keywords: Fall detection, machine learning, XGBoost , IoT.

1 Introduction

In the past decade, the world population has been ageing due to an increase in life expectancy and medical advances [1,2]. According to the United Nations, in 2017, there was approximately 962 million people aged 60 or above globally which accounts for 13 % of the world population [1]. The population of elders around the world is increasing by about 3 % each year [1]. Due to this ageing population, measures need to be taken to enhance the Quality of Life (QoL) of elders by ensuring that they have an active and independent life. Falls represent one of the main health risks to elders worldwide and the rapidly ageing population enhances this threat to public health [2]. As people get older, they become more fragile and their senses deteriorates, for example, their vision worsens and their ability to perceive the surroundings diminishes hence making them more prone to falls [3]. Moreover, some of the medicines prescribed to elderly people for medical problems may cause a decrease in mental alertness, hence increasing the probability of a fall. According to the World Health Organization (WHO), about 30 % of people above 65 are victims of accidental falls annually and for people above 80, the fall rate reaches about 50 % [4]. Falls more often occur indoor and are associated to activities of daily living (ADLs). In order to tackle this issue, fall detection systems have been designed which make use of sensors and other components to detect falls and allow timely assistance to the fall victims by alerting emergency contacts and healthcare personnel immediately after a fall event [4-7].

A robust fall detection system monitors the fall and alerts caregivers, hence alleviating their burden and diminishes the workload on resource constrained healthcare systems [8]. Fall detection systems can be classified into two main categories: context-aware systems and wearable device based approaches [9]. Context-aware systems use networks of sensors deployed in the environment to detect falls. They include ambient sensors as infrared, floor, radar, microphones, and pressure sensors as well as vision-based devices [10]. Wearable device based approaches consist of various sensors embedded in a fabric that is worn by the user which can detect the latter's position and motion. Wearable sensors with accelerometers and gyroscopes are frequently used in fall detectors [10]. Smart phones equipped with accelerometer sensors have also been used as fall detection systems [11-14]. A mobile application is used to analyse the data from the sensor and if a threshold value is detected, a fall is detected. A smart home monitoring unit that uses the ZigBee wireless sensors has been developed which can monitor the daily activities of the elder in real time by the use of sensors placed on the household appliances used by the elder [15,16]. This method reduces false alarms and can also predict unusual behaviour. Systems using omni-directional cameras have been developed which use image and audio processing techniques coupled with machine learning techniques to detect if a fall event has occurred. These systems have proven to be robust systems, but they have numerous drawbacks including high cost, lack of privacy as they are located indoors.

For wearable devices, the most common used sensors are tri-axial accelerometers and gyroscopes. These systems use a threshold based fall detection algorithm to detect falls [17,18]. The wearable devices have numerous advantages including low power consumption, low weight, small size and ease of operation. One main drawback is that being a wearable device, the elderly can often forget to wear it. IoT is being incorporated in wearable fall detection systems to help decrease the workload of the device and prolong its operational longevity [19,4,20]. When a fall is detected, the IoT gateway is able to notify the emergency contacts and healthcare professionals about the details of the fall. The data about each fall event is stored on the cloud and new models can be built using this information.

a. Faculty of Engineering, University of Mauritius, dashil.cahoolessur@umail.uom.ac.mu
b. Department of Electrical and Electronic Engineering, University of Mauritius, b.rajkumarsingh@uom.ac.mu

When a fall detection system is being designed, the following aspects need to be considered: obtrusiveness, privacy, energy consumption, computational cost, noise distorting the signal being monitored and defining a threshold for threshold-based systems [21]. One of the main issues when designing a fall detection system is to implement a fall detection algorithm that is reliable in detecting falls and has a low probability of false alarms.

Nowadays, machine learning (ML) techniques are being implemented in fall detection algorithms so as to increase accuracy of predicting a fall and decreasing the probability of false alarms in a fall detection system. They include Support Vector Machine (SVM), Decision Tree (DT), Naive Bayes (NB), Least Squares Method (LSM), K-Nearest Neighbour (K-NN), and Artificial Neural Networks (ANNs), XGBoost, Random Forest (RF), LightGBM [17,19,21-30]. The related studies and results are given in the following paragraphs.

In [22], the issue of false alarms in a fall detection system was tackled using the one-class SVM machine learning algorithm. Here, the sensor used to get the acceleration data was the tri-axial accelerometer, MMA7260Q. The 12 volunteers, eight males and four females aged from 10 to 70 years old with height from 1.36 m to 1.80 m were selected to attend the experiments. The fall training data was obtained from young adult volunteers. The ADLs data was obtained from both young adults and elders. This system had a fall prediction accuracy of 96.7 %. However, it does not include any method of updating the machine learning model and alerting the emergency contacts if a fall event occurs. The false alarm rate was also not reported.

In [23], activity recognition coupled with machine learning was used to distinguish between fall and non-fall events. Four approaches were used to be able to detect falls; first being the use of accelerometers, secondly, the use of gyroscopes, then, visual detection and finally, using video to reconstruct posture of the users. Three persons attended the experiments. Recordings of walking, falling, standing, sitting down and lying were used to obtain the coordinates of 12 different body parts during each activity. These coordinates were used to obtain the three attributes used for machine learning namely, reference, body and angle attributes. These attributes were used to train eight machine learning models among C4.5 decision trees, RIPPER decision trees, NB, K-NN, SVM, RF, Bagging and Adaboost M1 boosting with an accuracy of 94.1 %, 93.1 %, 89.5 %, 97.1 %, 97.7 %, 97.2 %, 97.2 %, 96.3 % and 93.7 % respectively. SVM was found to have the highest accuracy. The false alarm rates were not mentioned.

The SVM machine learning algorithm was used in [24] to implement the fall detection algorithm. The device was worn on the waist of the subject and consisted of tri-axial accelerometer, a microprocessor and a Bluetooth module. Four features were used to train the model using two-fold cross-validation on the dataset that was split in 50 % training data and 50 % test data. The system achieved an accuracy of 99.14 %, a sensitivity of 99.60 %, and a false positive rate of 1.3 %. The activities were performed by two groups of young volunteers (20 subjects) and one group of elderly volunteers (five subjects). Here, the model was trained mostly with data from young volunteers.

In [19] it is proposed to use only the streaming accelerometer data from a commodity-based smartwatch device to detect falls. The smartwatch is paired with a smartphone as a means for performing the computation necessary for the prediction of falls in real time without incurring latency in communicating with a cloud server while also preserving data privacy. Both SVM and NB machine learning algorithms for the creation of the fall model were experimented. They showed that using a NB machine learning model, an accuracy of 93.33 % and a false positive rate of 8% can be obtained.

A novel and truly unobtrusive detection method was proposed based on the advanced wireless technologies, WiFall [25]. WiFall employs the time variability and special diversity of Channel State Information (CSI) as the indicator of human activities. As CSI is readily available in prevalent in-use wireless infrastructures, WiFall withdraws the need for hardware modification, environmental setup and worn or taken devices. WiFall is implemented on laptops equipped with commercial 802.11n NICs. As demonstrated by the experimental results, WiFall with SVM algorithm yielded 94 % detection precision with false positive rate of 13 % on average.

In [4], a 6LoWPAN (IPv6 over Low -Power Wireless Personal Area Networks) wearable fall detection device consisting of a tri-axial accelerometer was designed and implemented. The fall detection algorithm was implemented using (DT) algorithm and the model was created from the Sisfall dataset using the BigML Data Analysis tool [26] and was stored in the cloud using MongoDB. A smart IoT gateway was set up which allowed the data from the accelerometer to be sent to a Big Data Analyser which will process and analyse the data from the sensor to decide if a fall has occurred or not. The model is updated each time new acceleration data is fed to it in order to increase its predictive accuracy. Whenever a fall is detected, the emergency contacts are notified through a MQ Telemetry Transport (MQTT) broker and the GPS information of the fall victim is also included in the notification. The data of each fall event is stored on the cloud so that medical professionals can easily access the data. The system had a fall prediction accuracy of 91.67 %, precision of 93.75 % and a false positive rate of 8 %. The fall detection system designed in [27] makes use IoT and an ensemble machine learning algorithm and consists of a tri-axial accelerometer on a 6LowPAN wearable device. This device is quite similar to that in [4] apart from the machine learning algorithm being used. The ensemble machine learning algorithm yielded an accuracy of 98.72 % as compared to 91.67 % in [4].

Vallabh *et al.* [28] investigated five different classification algorithms for fall detection using the MobiFall dataset. The K-NN algorithm obtained an overall accuracy of 87.5 % with a sensitivity of 90.70 %, and a false positive rate of 16.22 %.

In [17], the IoT based fall detection system is a smartwatch with a tri-axial accelerometer embedded in it. The smartwatch is used in parallel with a smartphone containing an Android application that uses the accelerometer data for fall prediction. The fall detection algorithm was implemented using SVM, NB and deep learning algorithms. For SVM and NB, four features were used to create the model

http://dx.doi.org/10.17159/2309-8988/2020/v36a2

whereas deep learning does not require feature extraction. Whenever a fall is detected, emergency contacts are alerted and the fall data is stored on the cloud so that it is available to medical professionals. When predicting falls in real-time, SVM obtained an accuracy of 64 %, NB got 56 % and deep learning achieved 70 %. The respective false positive rates were 14 %, 18 % and 46 %. The deep learning model was the better choice, but the accuracy of its predictive performance dropped from 99 % with an offline dataset to 70 % in real-time. The drop in performance can be explained as follows. These systems are tested in a controlled environment and are optimized for a given set of sensor types, sensor positions, and subjects. Validation tests with elderly people significantly reduce the fall detection performance of the tested features [17].

In [29], the long-term fall detection sensitivity and false alarm rate of a fall detection prototype was studied in real-life use using an automatic accelerometric fall detection system. The fall detection system detected 12 out of 15 real-life falls, having a sensitivity of 80.0 %, with a false alarm rate of 0.049 alarms per usage hour with the implemented real-time system. It was also found that by some slight variation of data analysis the false alarm rate was reduced to 0.025 false alarms per hour, equating to 1 false fall alarm per 40 usage hours.

A low-power fall detector using triaxial accelerometry and barometric pressure sensing was proposed that minimizes its power consumption using co-design of hardware and firmware and threshold optimization technique [30]. Additionally, the thresholds of the fall detection algorithm were optimized to achieve a balance between sensitivity and false alarm rate. The fall detector achieved a high sensitivity (91 %) with a low false alarm rate (0.1149 alarms per hour).

XGBoost [31] is short for eXtreme Gradient Boosting package. XGBoost is an open source, efficient and popular implementation of gradient boosted decision trees. It belongs to a broader umbrella of Distributed Machine Learning Community (DMLC). It was made by the contributions of many developers with main contributions from Tianqi Chen and Carlos Guestrin [31,32]. It has been used by several Kaggle competition winners. XGBoost strictly prioritizes computational speed and model performances [32]. Advantages of XGBoost include ease of use, efficiency, feasibility, easy to install, highly developed R/Python interface for users, automatic parallel computation on a single machine and good accuracy for most data sets [33-35].

In [36], a self-adaptive system was designed so that it is able to adapt the fall detection algorithm to the current position of the wearable using XGBoost [31] and LightGBM [37]. These algorithms are known to perform better than the RF in numerous machine learning applications. These algorithms are tree-based ensemble methods, but differ significantly from the RF in the sense that the RF relies on bagging which decreases the variance of the prediction where XGBoost and LightGBM use boosting which should reduce the bias.

A fall detection system should not miss a single fall due to the medical implications every fall may carry on which implies a fall detection model with a high Recall or Sensitivity. A missed fall is represented in our evaluation experiments as a false negative (FN). We also do not want to have too many false alarms, which in our evaluation as represented as false positives (FPs), and thus, we want to achieve a high sensitivity and a low false alarm rate. In this respect, an IOT-based fall detection system is designed and implemented with machine learning techniques that will be able to detect falls accurately and minimize the negative consequences of a fall allowing elderly people to live an active and independent life. The objectives are to design and implement an accurate and reliable fall detection system, to detect falls and alert emergency contacts instantaneously for immediate assistance, to store the data about the fall event on the cloud to allow healthcare professionals to access the data easily, to be able to distinguish between real falls and false alarms using machine learning techniques, to use the data stored on the cloud to train the system so as to increase accuracy of detection and to provide GPS location of the fall victim in case of a fall.

Section 2 describes the methodology used. Section 2 also describes how the machine learning model was designed together with the electrical and logical design of the fall detection system. The implementation and testing of the fall detection system being designed are also mentioned in this section. Section 3 provides an analysis of the final system by performing experiments and analysing the results. Section 4 concludes the research work.

2 Methodology

2.1 Training Data from Sisfall

The training data for the model was acquired from the Sisfall: A Fall and Movement Dataset [38,39] which contains data from both young and elderly participants. The dataset consists of 4505 files of which 1798 are related to 15 types of fall and 2707 files are related to 19 types of ADL. The movement data was measured and collected at a sample frequency of 200 Hz using two triaxle accelerometers (ADXL345 and ITG3200) and gyroscope (MMA8451Q).

The participants were 23 young adults between the age of 19 to 30 and 15 elders between the ages 60 to 75 years old. In this project, the data obtained from the elderly between the ages 60 to 75 years old is used to train our model as it is a better representation of the movement of elders for whom this fall detection system is being built. The types of falls and ADLs are given in tables 1 and 2 respectively.

2.2 Choice of Machine Learning Algorithm

Choosing the appropriate machine learning algorithm for a given problem depends on the following factors: the size of the dataset being used, whether we are dealing with labelled or unlabelled data, the amount of features in the dataset, the type of output the model is expected to generate, prediction accuracy of the algorithm, the amount of time required to train the model, whether the algorithm is prone to overfitting, the number of parameters available within the algorithm to tune it and whether the algorithm is able to handle sparse data.

The algorithm cheat sheet from Microsoft Azure machine learning [40] was used in order to have a better understanding of how to proceed to find the most appropriate machine learning algorithm for the given problem.

For the fall detection algorithm in this project, the Sisfall dataset being used is a labelled dataset having four features based on the components of the acceleration as described in

Table 1 Types of falls in Sisfall dataset

Code	Activity	Trials	Duration
F01	Fall forward while walking caused by a slip	5	15 s
F02	Fall backward while walking caused by a slip	5	15 s
F03	Lateral fall while walking caused by a slip	5	15 s
F04	Fall forward while walking caused by a trip	5	15 s
F05	Fall forward while jogging caused by a trip	5	15 s
F06	Vertical fall while walking caused by fainting	5	15 s
F07	Fall while walking, with use of hands in a table to dampen fall, caused by fainting	5	15 s
F08	Fall forward when trying to get up	5	15 s
F09	Lateral fall when trying to get up	5	15 s
F10	Fall forward when trying to sit down	5	15 s
F11	Fall backward when trying to sit down	5	15 s
F12	Lateral fall when trying to sit down	5	15 s
F13	Fall forward while sitting, caused by fainting or falling asleep	5	15 s
F14	Fall backward while sitting, caused by fainting or falling asleep	5	15 s
F15	Lateral fall while sitting, caused by fainting or falling asleep	5	15 s

Table 2 Types of ADLs in Sisfall dataset

Code	Activity	Trials	Duration
D01	Walking slowly	1	100 s
D02	Walking quickly	1	100 s
D03	Jogging slowly	1	100 s
D04	Jogging quickly	1	100 s
D05	Walking upstairs and downstairs slowly	5	25 s
D06	Walking upstairs and downstairs quickly	5	25 s
D07	Slowly sit in a half height chair, wait a moment, and up slowly	5	12 s
D08	Quickly sit in a half height chair, wait a moment, and up quickly	5	12 s
D09	Slowly sit in a low height chair, wait a moment, and up slowly	5	12 s
D10	Quickly sit in a low height chair, wait a moment, and up quickly	5	12 s
D11	Sitting a moment, trying to get up, and collapse into a chair	5	12 s
D12	Sitting a moment, lying slowly, wait a moment, and sit again	5	12 s
D13	Sitting a moment, lying quickly, wait a moment, and sit again	5	12 s
D14	Being on one's back change to lateral position, wait a moment, and change to one's back	5	12 s
D15	Standing, slowly bending at knees, and getting up	5	12 s
D16	Standing, slowly bending without bending knees, and getting up	5	12 s
D17	Standing, get into a car, remain seated and get out of the car	5	25 s
D18	Stumble while walking	5	12 s
D19	Gently jump without falling (trying to reach a high object)	5	12 s

section 2.4.2 and one label each consisting of 200 097 data points meaning the size of the dataset is quite large. Here, we can eliminate the K-NN and SVM algorithms as they exhibit poor performance as the dataset gets larger. One suitable algorithm will be the XGBoost algorithm due to its good handling of large datasets.

As the dataset consists of labelled features, supervised machine learning methods must be used. Moreover, the output will be a number, either 0 or 1, whereby, 0 denotes no fall has occurred and 1 denotes a fall. As the output is a 0 or 1, we will be dealing with a classification problem. From these observations, the suitable algorithms will be the XGBoost and DT.

For a fall detection system, fast execution time and accuracy of prediction are *sine qua non* factors and hence an algorithm that satisfies these conditions must be selected. Furthermore, the training time should be fast in so far as the predictions are obtained instantaneously. It is also desired for the machine learning algorithm to be flexible to tuning of hyper parameters in order to optimise the performance of the model created.

From literature [31-36], the machine learning algorithm that mostly satisfies the numerous criteria is the XGBoost algorithm. The latter can handle large datasets and sparse data, has good performance with numerical features, possesses several hyper parameters that can be tuned to obtain optimal performance and lastly, it has fast processing speed and accurate predictive performance.

2.3 Model Training

The XGBoost model was created in the Python programming language using the Spyder software. The dataset was converted into a csv file and loaded in the Python IDE. The dataset contains four features and each row of features has a label, either '0' or '1', whereby, '0' denotes no fall and '1' denotes a fall event. The features and label are extracted from the dataset as X and Y respectively ensuring that the headers of the data are not selected. Then the dataset is split into the training and test datasets using the train test split function from the sklearn.model selection library. Initially the hyperparameters of the XGBoost algorithm are set to default values and the training process is started. Afterwards, the model is tested using the test dataset and the prediction accuracy and confusion matrix are obtained. Using a trial and error process, the hyperparameters' values are modified until the best prediction accuracy is achieved. The tree booster hyperparameters used in the model are described below [41].

1. **Learning rate [default=0.3]**
   - Makes the model more robust by shrinking the weights on each step.
   - Typical final values to be used: 0.01-0.2
2. **min_child_weight [default=1]**
   - Defines the minimum sum of weights of all observations required in a child.
   - Used to control over-fitting.
3. **max_depth [default=6]**
   - The maximum depth of a tree.
   - Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
   - Typical values: 3-10
4. **gamma [default=0]**

- A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split.

5. **subsample [default=1]**
   - Denotes the fraction of observations to be randomly sampled for each tree.
   - Typical values: 0.5-1
6. **colsample_bytree [default=1]**
   - Denotes the fraction of columns to be randomly samples for each tree.
   - Typical values: 0.5-1
7. **colsample_bylevel [default=1]**
   - Denotes the subsample ratio of columns for each split, in each level.
8. **lambda [default=1]**
   - This used to handle the regularization part of XGBoost. Though many data scientists don't use it often, it should be explored to reduce overfitting.
9. **alpha [default=0]**
   - Can be used in case of very high dimensionality so that the algorithm runs faster when implemented
10. **n_estimators [default=100]**
    - Number of boosted trees to fit.

The design process shown in figure 1.

2.3.1 Performance Metrics

The sensitivity (SE), specificity (SP), accuracy (AC) and false positive rate (FPR) are computed as follows [42]:

$$SE = \frac{TP}{TP+FN} \quad (1)$$

It measures the capacity of the system to detect falls:

$$SP = \frac{TN}{TN+FP} \quad (2)$$

It is the capacity of the system to detect falls only when they occur:

$$AC = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

It is the ability of the system to differentiate between falls and no-falls:

$$FPR = 1 - SP \quad (4)$$

False positive rate (*FPR*) is the number of incorrect fall predictions divided by the total number of ADLs, where *TP* and *TN* are the true positives and negatives; *FP* and *FN* the false positives and negatives, respectively.

2.4 Experimental Setup for Testing the Model in Real Time

Figure 2 below illustrates the block diagram of the fall detection system designed.

In this work, the issue of fall detection is tackled using an IoT based system consisting of a micro-controller and a network of sensors including a tri-axial accelerometer embedded in a wearable device that is able to take advantage of fog and cloud computing. The wearable device is worn around the waist as it has been found as the optimal position for detecting falls [43,44,45]. Whenever a fall is detected, an alarm is activated by means of a buzzer and the emergency contacts are notified immediately through the IoT gateway and also by SMS with the aid of a GSM and GPRS module. A GPS module is also used so that the location of the person is available. Moreover, the details about the fall are stored on the cloud in order for healthcare professionals to access the data easily. The issue of false alarms is tackled by implementing ML techniques in the fall detection algorithm. Compared to other systems, here the fall detection system algorithm uses the XGBoost machine learning algorithm which is known for its fast execution and better predictive performance when compared to other machine learning algorithms. This helps to increase the accuracy of detecting a fall and reduce the probability of false alarms by a greater extent.

The main limitation of this system is that the Sisfall dataset used to build the machine learning model is mostly data acquired from falls of young or elderly people as it is unfeasible to subject elderly people to simulated falls in order to form more accurate datasets. Hence the model may not be an accurate representation of the movement of older people.

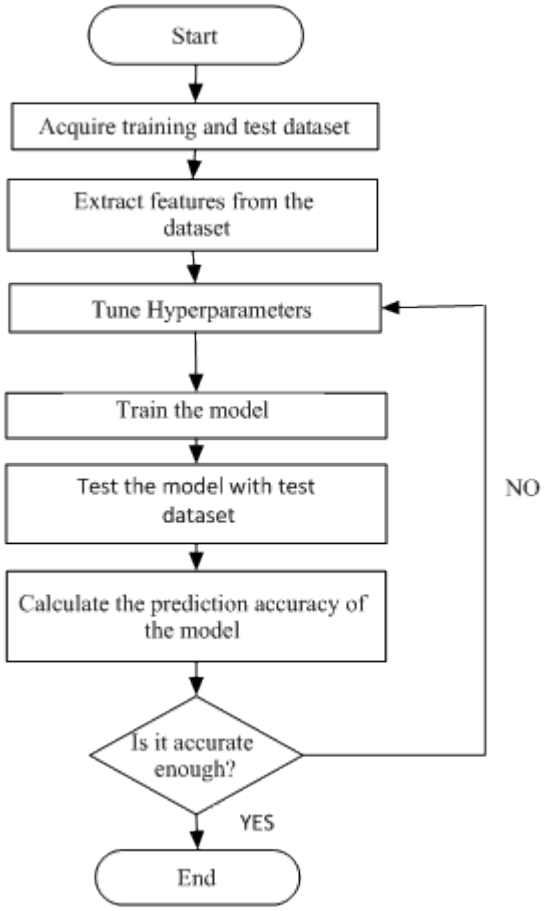

Figure 1 Flowchart showing the design process of the machine learning model

2.4.1 The Prototype

The prototype of the wearable device as shown in figure 3 was built using a micro-controller, a tri-axial accelerometer, a GPS and GSM module and a buzzer all embedded on an elastic belt. The microprocessor being used is a Raspberry Pi 3B running the Linux-based Raspbian operating system. The Raspberry Pi was chosen over the Arduino boards due to its faster processor and better memory which allows it to run the

http://dx.doi.org/10.17159/2309-8988/2020/v36a2

XGBoost machine learning algorithm. Also, it has integrated WiFi allowing easier and more rapid communication with the cloud services for IoT.

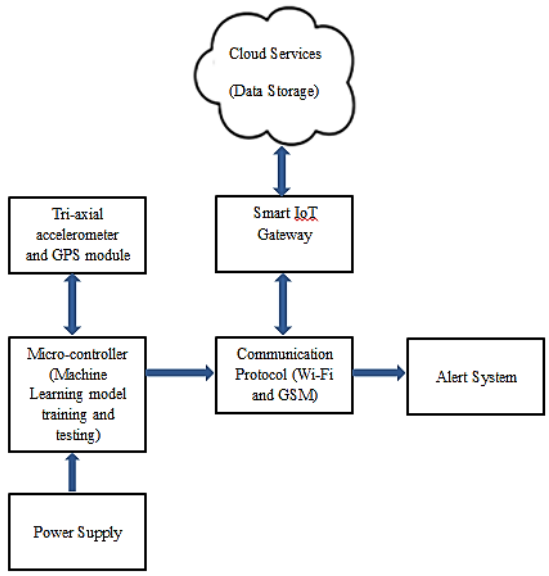

Figure 2 Block diagram for fall detection system

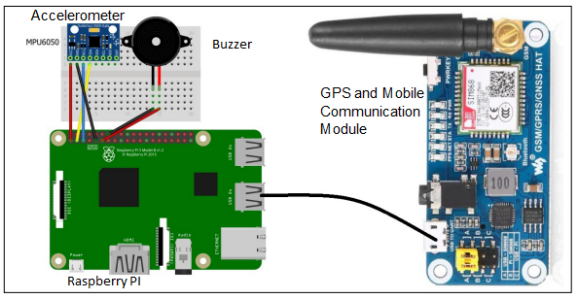

Figure 3 Components of the prototype

To obtain the acceleration values, the MPU6050, which is a tri-axial accelerometer and gyroscope, was used. This sensor was used due to its low power consumption, cheap price and high performance. The Waveshare SIM868 Development Board was used to obtain location through GPS and sent fall notification via SMS as it is easy to use, consumes low power and has multiple communication features such as GPRS, GSM, GNSS and Bluetooth. Finally, a buzzer was connected to the Raspberry Pi to sound an alarm whenever a fall has been detected so that nearby people can attend the fall victim. As per [44] and [45], the optimal location for the wearable device is on the waist of the user as this body position is near to the centre of gravity of the person. Hence, the acceleration values obtained from this location will provide more accurate distinction between falls and ADLs.

2.4.2 Feature Extraction

This process is one of the most important steps in the design of the machine learning model. Here, the data obtained from the accelerometer is converted into useful data which is characteristic to the problem to be solved. From [18] and [19], the features giving the best performance with fall detection systems are indicated. From these features, four were selected and are given as follows:

1. Feature 1: Sum vector magnitude.

$$Feature\ 1 = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (5)$$

2. Feature 2: Sum vector magnitude on horizontal plane.

$$Feature\ 2 = \sqrt{a_x^2 + a_z^2} \quad (6)$$

3. Feature 3: Angle between z-axis and vertical plane.

$$Feature\ 3 = \text{atan2}\sqrt{a_x^2 + a_y^2}, a_z \quad (7)$$

4. Feature 4: Orientation angles with respect to gravity.

$$Feature\ 4 = \cos^{-1}\left(\frac{a_z}{\sqrt{a_x^2+a_z^2}}\right) \quad (8)$$

In the above equations,
a_x: x-axis accelerometer reading,
a_y: y-axis accelerometer reading,
a_z: z-axis accelerometer reading,
atan2(a, b) function returns value of atan(a/b) in radians.

2.4.3 Fall Detection Algorithm

The fall detection algorithm uses the XGBoost model that has been trained on the Sisfall dataset to predict if a fall has occurred or not. The values of the accelerometer are read and converted into the features that required to obtain a prediction. The features are then used to test the model and get a prediction. The functioning of the fall detection algorithm is fully illustrated in the flowchart of figure 4.

2.4.4 Design of IoT Platform

The fall detection system used the Blynk IoT platform to connect to the cloud services. The communication between the device and the IoT platform was set up using the Transmission Control Protocol (TCP). The Blynk IoT platform was chosen because it is fast, reliable and relatively easy to set up. The fall event data was stored in the cloud using Google sheets which are easily accessible by healthcare professionals willing to see and analyse the data.

2.4.5 Feature Extraction using MPU6050 Sensor

The MPU6050 was connected to the Raspberry Pi as shown in figure 5. The appropriate libraries were required to be installed on the Raspberry Pi for the proper functioning of the accelerometer.

After making the connections, the Python code was run. The values of acceleration in the x, y and z axes were retrieved from the sensor. Then equations 5 to 8 were used to calculate the features required to test the machine learning model and each feature was displayed on the Raspberry Pi's terminal.

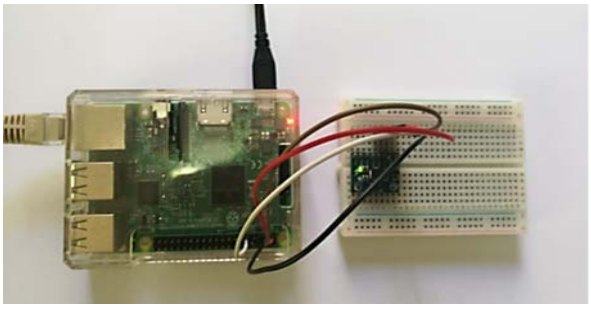

Figure 5 Connection of MPU6050 with Raspberry Pi

In the fall detection algorithm, the features extracted from the MPU6050 are tested with the XGBoost model for each interval of five seconds. Hence, the features were obtained from the sensor for 5 seconds and then stored in a csv file from which they were fed to the model for prediction using the Python pandas library.

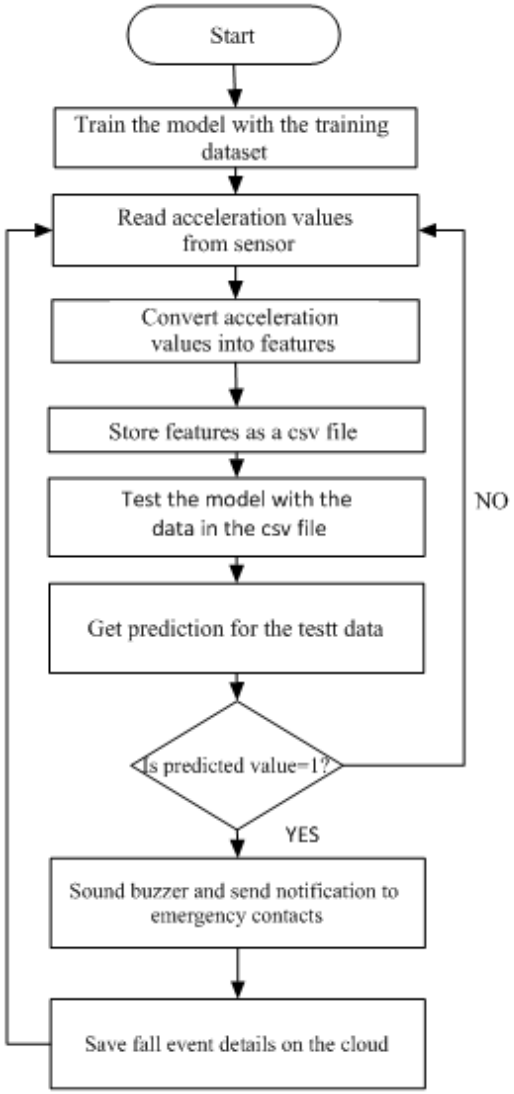

Figure 4 Fall detection flowchart

2.4.6 GPS Location and Fall Alert

To obtain the GPS location, the Python code was run. If a fall event occurs, the following SMS is sent to emergency contacts as shown in figure 6.

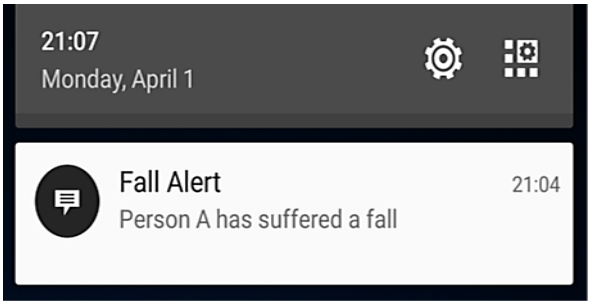

Figure 6 Fall Alert received upon fall event.

2.4.7 Prototype of Wearable Device

After all the sensors and modules had been tested, they were assembled on the elastic belt to make the prototype wearable device as shown in the figure 7.

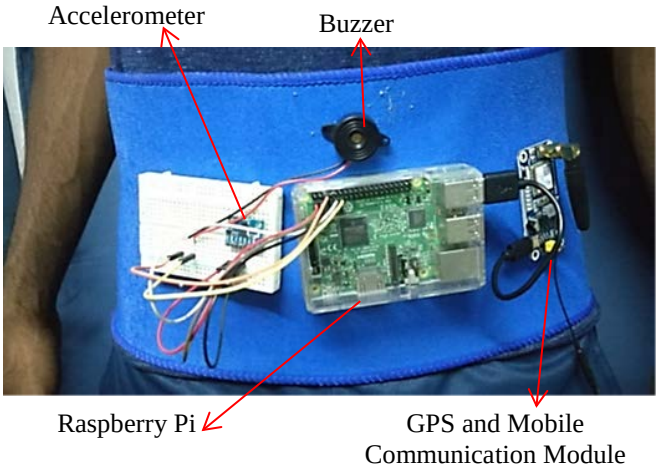

Figure 7 Prototype of wearable device

2.4.8 Fall Detection Algorithm

To initiate the fall detection algorithm, the wearable device was worn by a volunteer. If a fall event occurred, "WARNING: A FALL HAS OCCURED" is displayed on the Raspberry Pi's terminal, the buzzer is triggered and a notification is sent to emergency contacts. If no fall has been detected, "NO FALL HAS OCCURED" is displayed. When testing the algorithm, the buzzer was replaced by an LED light that turns on upon a fall event and remains off in case of no fall as shown in figure 8.

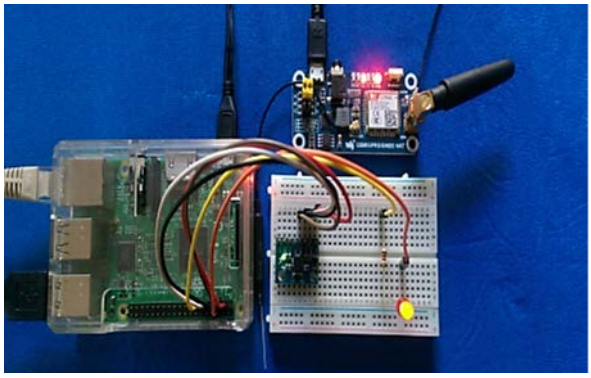

Figure 8 State of LED when fall occurs

2.4.9 Real Time Data

To evaluate the fall detection system, an experiment consisting of 15 fall events and 5 ADLs including walking, standing, running, sitting down and lying down was carried out on a volunteer of 23 years old. The individual wore the wearable device on his waist and carried out the different activities at random. Then, a confusion matrix was obtained and the accuracy of the system was evaluated.

3 Results

3.1 Training of Machine Learning Model

The Sisfall dataset was loaded in the Python IDE and 75 % was used as training data and 25 % for test data. Several trials

of tuning the different parameters were made until the best prediction accuracy was achieved.

The tree boosting parameters are adjusted to give the best results. In particular the colsample_bytree, colsample_bynode and colsample_bylevel were increased from 0.5 to 1. The performance metrics are shown in table 3. T1-T3 are the different trials.

Table 3 Boosting hyperparameters and performance metrics

Hyperparameter	**T1**	**T2**	**T3**
colsample_bytree	0.5	0.8	1
colsample_bynode	0.5	0.8	1
colsample_bylevel	0.5	0.8	1
subsample	1	1	1
learning rate	0.3	0.3	0.3
max_depth	1	5	10
alpha	0	0	0
gamma	2	2	2
Min_child_weight	1	1	1
n_estimators	15	15	15
TP	24960	25074	20182
FP	2315	1729	1348
FN	1297	1183	841
TN	21660	22246	17815
SE	95.1	95.5	**96**
SP	90.3	92.8	93
AC	92.8	94.2	94.6
FPR	9.7	7.2	**7**

From table 3, trial 3 (T3) yields the best predictive performance with an accuracy of 94.6 % and an FPR of 7 %. Therefore, the parameters from trial 4 were used to create the XGBoost model.

By tuning the tree-specific parameters like colsample_bytree, colsample_bynode, colsample_bytlevel, max_depth of the tree, we can change the sensitivity and specificity and hence the false positive rate. Increasing the value of these parameters tend to reduce the FPR and increase the SE. It is observed that tuning these parameters to the maximum limit gives the best results with SE = 96 %, SP = 93 %, AC = 94.6 % and FPR = 7% when only elderly volunteers were considered from the Sisfall dataset.

3.2 Discussion

In [27] an accuracy of 98.72 % was obtained with the Sisfall dataset which is more than the 96 % obtained by the XGBoost algorithm. Since we have trained the model on the data from the elderly volunteers only rather than the whole dataset including young volunteers, a direct comparison cannot be done. Liu [24] used data mostly from young volunteers to train their model. The high accuracy of 99.14 % cannot be compared to our work in respect to the above reasoning. Also, in [22], an accuracy of 96.7 % was reported. 12 subjects attended the experiments and the fall training data was obtained from young adult volunteers only. Previous works have demonstrated that the accuracy of the approaches where mainly young volunteers are used is significantly reduced when tested on institutionalized [47] and independent [17] elderly people [38].

As it can be seen in table 4, the XGBoost algorithm outperforms the work done by Mauldin *et al*. [20] and Vallabh *et al*. [28] significantly. Also, the XGBoost algorithm has a better accuracy of 96 % as compared to 91 % [4] and 93 % [19] that used DT and NB respectively. The false positive rate and sensitivity of the XGBoost algorithm is better than that of [19].

Table 4 Comparison with past studies

	ML	**SE %**	**AC %**	**SP %**	**FPR %**
Mauldin et al. [20]	NB	66	64	62	38
	SVM	26	56	86	14
	Deep Learning	86	70	54	46
Yacchirema *et al*. [4]	DT	100	91	94	6
Ngu *et al*.[19]	NB	94	93	92	8
Wang *et al*. [25]	RF	n/a	n/a	82	18
Vallabh *et al*. [28]	K-NN	90.7	87.5	83.8	16.2
Proposed	XGBoost	97	96	93	7

3.3 Real Time Tests

It is also noted that when testing the algorithm with real time data, we obtained an SE of 100 % as shown in table 5. This demonstrates that the XGBoost model does very well in predicting falls. Two false negatives were obtained and the model has a higher false positive rate of 28 %. Our XGBoost model performs the best on detecting falls. However, the XGBoost is not very accurate on ADLs. Still, the results are still much better than that of [20].

Table 5 Real time tests with hyperparamters T4

TP	12
FP	2
FN	0
TN	5
SE	100
SP	71.4
AC	89.5
FPR	28.6

The fall detection sensitivity of 100 % with real-life falls presented in the present study is higher than with experimental falls (97 %), but the SP is lower by 23 %.

We recognize that our fall detection model is tested using data from a healthy and young volunteer, which might not reflect the actual fall data from elderly people. Furthermore,

http://dx.doi.org/10.17159/2309-8988/2020/v36a2

the real time falls database included less subjects and number of ADLs considered was limited as compared to the Sisfall dataset, which may explain the difference. However, the results from the Sisfall dataset with the XGBoost algorithm look promising in fall detection and should be investigated further. An extension of this work would be to test the model with a larger number of elder participants and ADLs. Additionally we should investigate the change in the position of the wearable device in the model. Also, in order to increase the effectiveness of the fall detection systems a false alarm button could be included to allow the user to cancel ADL detected as falls (false positives) [29].

4 Conclusion

Falls among elderly people is a prominent issue that needs to be addressed so as to increase the QoL of the elderly. This project has presented a method to tackle this issue by designing and implementing a fall detection system that incorporates machine learning in its decision making and IoT to store data and send alerts. The machine learning algorithm used was XGBoost which is known for its accuracy and speed. The machine learning model was trained using the online Sisfall dataset. For fall prediction, the features to be tested with the model are obtained from the accelerometer embedded on the wearable device which is worn on the waist by the user. The waist was found to be the optimal location for the wearable device and moreover, in this position, the elders can easily wear it. In case of a fall event, a buzzer is sounded to alert nearby people and emergency contacts and healthcare professionals are notified. The GPS location of the fall victim is also provided.

The XGBoost algorithm outperforms the work done by Mauldin [20] and Vallabh *et al*. [28] significantly when tested using the Sisfall dataset. The XGBoost algorithm has a better accuracy of 96 % as compared to 91 % [4] and 93 % [19] that used DT and NB respectively. The fall detection sensitivity of 100 % with real-life falls presented in the present study is higher than with experimental falls (97 %), but the SP is lower by 23 %. A false alarm button could be included in the system to prevent false alarms. Further tests with other XGBoost parameters need to be done to optimize the performance of the model. The number of participants should also be increased.

The details of the fall event is stored on the cloud to allow easy access to the data. The system was tested with a series of falls and ADLs and its accuracy and precision were evaluated. The error rate was small showing that the proposed system can detect falls with high accuracy and reduces the probability of false alarms.

To improve the system, the machine learning model can be designed and trained on a cloud platform such as Google Cloud which supports the XGBoost algorithm. This will allow more rapid data transfer and quicker response as all the processing will be done in cloud. Moreover, new machine learning algorithms such as neural networks, can be used to design the fall detection algorithm and try to improve the predictive performance.

References

[1] United Nations. World Population ageing. URL www.un.org/en/development/desa/population/publications/pdf/ageing/WPA2017_Highlights.pdf, 2017.

[2] World Health Organization. *Global Report on Falls Prevention in Older Age*. World Health Organisation, 2007.

[3] U.S. National Library of Medicine. Aging changes in the senses. URL medlineplus.gov/ency/article/004013.htm.

[4] D. Yacchirema, J. S. de Puga, C. Palau and M. Esteve. Fall detection system for elderly people using IoT and Big Data. *Procedia Computer Science*, 130:603-610, 2018.

[5] A. K. Bourke, P. W. J. van de Ven, A. E. Chaya, G. M. O'Laighin and J. Nelson. Testing of a long-term fall detection system incorporated into a custom vest for the elderly. *30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. pages 2844–2847, 2008.

[6] M. Belshaw, B. Taati, D. Giesbercht and A. Mihailidis. Intelligent vision-based fall detection system: Preliminary results from a real world deployment. *RESNA/ICTA 2011: Advancing Rehabilitation Technologies for an Aging Society*, Toronto, Canada, 5–8 June, 2011.

[7] M. Belshaw, B. Taati, J. Snoek and A. Mihailidis. Towards a single sensor passive solution for automated fall detection. *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. pages 1773–1776, 2011.

[8] A. Sixsmith and N. Johnson. A smart sensor to detect the falls of the elderly. *IEEE Pervasive Computing*, 3(2):42–47, 2004.

[9] T. Xu, Y. Zhou and J. Zhu. New advances and challenges of fall detection systems: A survey. *Applied Sciences*, 8(3):418, 2018.

[10] R. Igual, C. Medrano and I. Plaza. Challenges, issues and trends in fall detection systems. *Biomedical Engineering Online*. 12(1):66, 2013.

[11] L. Martínez-Villaseñor, H. Ponce, J. Brieva, E. Moya-Albor, J. Núñez-Martínez and C. Peñafort-Asturiano. UP-fall detection dataset: A multimodal approach. *Sensors*. 19(9):1988, 2019.

[12] G. Vavoulas, M. Pediaditis, E. G. Spanakis and M. Tsiknakis. The MobiFall dataset: An initial evaluation of fall detection algorithms using smartphones. *13th IEEE International Conference on BioInformatics and BioEngineering*, pages 1-4, 2013.

[13] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, P. Corsini and A. Vecchio. A smartphone-based fall detection system. *Pervasive and Mobile Computing*, 8(6):883-899, 2012.

[14] I. N. Figueiredo, C. Leal, L. Pinto, J. Bolito and A. Lemos.. Exploring smartphone sensors for fall detection. *mUX: The Journal of Mobile User Experience*, 5(1):2, 2016.

[15] N. K. Suryadevara, A. Gaddam, R. Rayudu and S. C. Mukhopadhyay. Wireless sensors network based safe

home to care elderly people: Behaviour detection. *Sensors and Actuators A: Physical*, 186:277-283, 2012.

[16] N. K. Suryadevara, M. T. Quazi and S. Mukhopadhyay. Intelligent sensing systems for measuring wellness indices of the daily activities for the elderly. *Eighth International Conference on Intelligent Environments*, pages 347-350, June 26, 2012.

[17] A. Sucerquia, J. López and J. F. Vargas-Bonilla. SisFall: A fall and movement dataset. *Sensors*, 17(1):198, 2017.

[18] D. Lim, C. Park, N. H. Kim, S.-H. Kim and Y. S. Yu. Fall-detection algorithm using 3-axis acceleration: Combination with simple threshold and hidden Markov model. *Journal of Applied Mathematics*, 2014.

[19] A. H. Ngu, P.-T. Tseng, M. Paliwal, C. Carpenter, W. Stipe. Smartwatch-based IoT fall detection application. *Open Journal of Internet of Things*, 4(1):87-98, 2018.

[20] T. R. Mauldin, M. E. Canby, V. Metsis, A. H. Ngu and C. C. Rivera. SmartFall: A smartwatch-based fall detection system using deep learning. *Sensors*, 18(10):3363, 2018.

[21] Y. S. Delahoz and M. A. Labrador. Survey on fall detection and fall prevention using wearable and external sensors. *Sensors*, 14(10):19806-19842, 2014.

[22] T. Zhang, J. Wang, L. Xu and P. Liu. Fall detection by wearable sensor and one-class SVM algorithm. *Intelligent Computing in Signal Processing and Pattern Recognition*, 858-863, 2006.

[23] M. Luštrek and B. Kaluža. Fall detection and activity recognition with machine learning. *Informatica*, 33(2):205-212, 2009.

[24] S. H. Liu and W. C. Cheng. Fall detection with the support vector machine during scripted and continuous unscripted activities. *Sensors*, 12(9):12301-12316, 2012.

[25] Y. Wang, K. Wu, K. and L. M. Ni. WiFall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing*, 16(2):581-594, 2016.

[26] BigML.com - Machine Learning made easy. URL https://bigml.com/, 2019.

[27] D. Yacchirema, J. S. de Puga, C. Palau and M. Esteve. Fall detection system for elderly people using IoT and ensemble machine learning algorithm. *Personal and Ubiquitous Computing*, 23(5-6):801-817, 2019.

[28] P. Vallabh, R. Malekian, N. Yea and D. C. Bogatinoska. Fall detection using machine learning algorithms. *24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1-9, 2016.

[29] M. Kangas, R. Korpelainen, I. Vikman, L. Nyberg and T. Jämsä. Sensitivity and False Alarm Rate of a Fall Sensor in Long-Term Fall Detection in the Elderly. *Gerontology*. 61(1):61-68, 2014.

[30] C. Wang, W. Lu, S. J. Redmond, M. C. Stevens, S. R. Lord and N. H. Lovell. A low-power fall detector balancing sensitivity and false alarm rate. *IEEE Journal of Biomedical and Health Informatics*. 22(6):1929-1937, 2017.

[31] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 785–794, August 2016.

[32] M. S. Siva Priya, B. K. Sahu, B. Kumar and M. Yadav. Network intrusion detection system using XG Boost. *International Journal of Engineering and Advanced Technology*, 9(1):4070-4073, 2019.

[33] T. He. XGBoosteXtreme Gradient Boosting. URL https://www.saedsayad.com/docs/xgboost.pdf

[34] H. Musa, A. Y. Gital, F. U. Zambuk, A. Umar, A. Y. Umar and J. U. Waziri. A comparative analysis of phishing website detection using XGBOOST algorithm. *Journal of Theoretical and Applied Information Technology*. 97(5):1434-1443, 2019.

[35] H. S. Choi, S. Kim, J. E. Oh, J. E. Yoon, J. A. Park, C. H. Yun and S. Yoon. XGBoost-based instantaneous drowsiness detection framework using multitaper spectral information of electroencephalography. *ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 111-121, August 2018.

[36] C. Krupitzer, T. Sztyler, J. Edinger, M. Breitbach, H. Stuckenschmidt and C. Becker. Beyond position-awareness—Extending a self-adaptive fall detection system. *Pervasive and Mobile Computing*. 58:101026, 2019.

[37] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 3146–3154, 2017.

[38] A. Sucerquia, J. D. López and J. F. Vargas-Bonilla. Real-life/real-time elderly fall detection with a triaxial accelerometer. *Sensors*, 18(4):1101, 2018.

[39] SisFall — SISTEMIC. URL sistemic.udea.edu.co/en/research/projects/english-falls/, 2018.

[40] Machine learning algorithm cheat sheet - Azure Machine Learning Studio. URL docs.microsoft.com/enus/azure/machine-learning/studio/algorithm-cheat-sheet, 2019.

[41] Complete Guide to Parameter Tuning in XGBoost with codes in Python. URL https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[42] Basic Evaluation Measures from the Confusion Matrix. URL https://classeval.wordpress.com/introduction/basic-evaluation-measures/

[43] A. Özdemir. An analysis on sensor locations of the human body for wearable fall detection devices: Principles and practice. *Sensors*, 16(8):1161, 2016.

[44] I. Cleland, B. Kikhia, C. Nugent, A. Boytsov, J. Hallberg, K. Synnes, S. McClean and D. Finlay. Optimal placement of accelerometers for the detection of everyday activities. *Sensors*, 13(7):9183-9200, 2013.

[45] J. D. López, C. Ocampo, A. Sucerquia and J. F. Vargas-Bonilla. Analyzing Multiple Accelerometer Configurations to Detect Falls and Motion. *VII Latin American Congress on Biomedical Engineering CLAIB*,

Bucaramanga, Santander, Colombia, 26-28 October 2016, pp. 169-172, 2017.

[46] N. Pannurat, S. Thiemjarus and E. Nantajeewarawat. Automatic fall monitoring: A review. *Sensors*, 14(7):12900-12936, 2014.

[47] F. Bagala, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra and J. Klenk. Evaluation of accelerometer-based fall detection algorithms on real-world falls. PLoS ONE 7(5), 2012.

