Machine Learning-Driven Fall Detection Using Wearable Sensors for Enhanced Safety

OBI-OBUOHA ABIAMAMELA[1], NGIM N. EWEZU[2], ADJEROH E. PRINCEWILL[3], DR. AKINWUMI O. ADERONKE[4]

[1,2,3,4]*Department of Mechanical and Mechatronics Engineering, Afe Babalola University, Ekiti, Nigeria.*

Abstract- The goal of this study is to create an accurate and dependable fall detection system utilizing machine learning methods. By merging accelerometer measurements from wearable sensors that covered motion patterns related with falls and regular activities, a huge dataset was constructed. To extract useful characteristics from sensor data, feature engineering approaches were used. After the training of the model, upon testing an accuracy value of 86.25% was attained, alongside a recall value of 90.68%, precision value of 83.16% and a f1 score of 85.91%. Finally, this work proposes a novel approach to fall detection based on machine learning approaches. Our research shows significant progress in accurately detecting falls, outperforming existing threshold-based approaches. The purpose of building an effective fall detection system is to promote individual safety and well-being, particularly in healthcare settings. The proposed technique has enormous potential to transform how we respond to fall-related events and provide crucial help to the aging population and those who work jobs that may cause them to fall, such as construction workers. This opens up fascinating new possibilities for future study and real-world use of machine learning-based fall detection systems, which will directly touch people's lives.

Indexed Terms- Accelerometer, Fall detection, Machine learning-based, Wearable sensors.

I. INTRODUCTION

All humans lose their balance and fall from time to time, it is a phenomenon that affects all of us. Although we consider falls to be prevalent, not all of them are benign. In actual fact, in 1996, falls were responsible for over 14,000 fatalities and 22 million hospital and doctor visits. For individuals aged 79 and over, they are the primary cause of unintentional injury mortality and the second most common cause for people of all ages. Men always die from falls at a higher rate than women, and the rate rises progressively with age. The two most frequently identified locations for fatal falls are residences and residential institutions [1].

However, within the population as a whole, there are certain demographics of people at a greater risk of it and who stand to be more affected by it. One such group is the elderly, falling is a problem that affects older persons all too frequently and can have serious repercussions. Over one-third of senior citizens experience unintentional falls, which causes them to lose their independence and experience fear. Unintentional falls frequently happen indoors when moving about, as in restrooms and on stairs. For those 65 and older, the percentage of falls varies from 28 to 35, and for those 70 and older, it goes from 32 to 42 percent [2].

Another group of people whose falls might interest us include those who faint. In medical terminology, fainting is also known as 'syncope'. It occurs when the heart is unable to adequately pump blood to the brain. Underlying medical conditions, such as balance or gait issues, vision impairment, drug side effects, or cognitive impairment, may be present in patients who fall frequently. In these situations, healthcare professionals must do a complete assessment of the patient's health and create a personalized care plan to lower the risk of falls. Seeing the effects of falling and the groups of people it affects, it becomes necessary to develop a system that helps speed up response to falls.

II. LITERATURE REVIEW

The goal of detecting falls is to get the victim prompt medical attention by alerting a responder. Broadly speaking, there are two approaches for detecting falls in terms of the inference engine a controller uses. The methods are; by using a threshold-based engine and a Machine Learning-based engine.

A. Inference Engine

Threshold-based fall detection systems work by capturing certain data from sensor — such as acceleration, angular velocity, or orientation — exceeds predetermined thresholds, it is considered to be a fall event. This approach is predicated on the idea that falls can be reasonably accurately identified by setting proper thresholds since the motion patterns during a fall are different from regular daily activities. In order to assess whether a fall event has occurred, this method entails setting precise thresholds for several parameters, such as acceleration, orientation, or angular velocity, which are then evaluated against the sensor readings in real-time [3]. It is a very easy approach to detecting falls once the threshold has been decided. However, it's crucial to remember that the right threshold value may change depending on the particulars of the situation, such as the type of sensor being used, the population being observed, and the demands of the particular application. As a result, the threshold needs to be properly established through comprehensive investigation and validation in the particular environment where the fall detection system will be used.

B. Machine Learning

The research area of Machine Learning (ML) is what enables computers to learn without being explicitly taught [4]. ML is the branch of Artificial Intelligence (AI) that teaches computers to make decisions by helping them find the relationship between the inputs and outputs in a dataset.

Most people agree that the current version of machine learning was created by Cornell University psychologist Frank Rosenblatt. Rosenblatt oversaw a group that used theories about the workings of the human nervous system to create a method that would enable a machine to recognize the letters of the alphabet. With a threshold component that converted analog signals into discrete ones, the device, which its creator called the "perceptron," used both discrete and analog impulses. The way it learned was similar to the psychological models developed of both human and animal learning, and it served as the model for modern artificial neural networks (ANN) [5].

In order to learn from data, machine learning uses algorithms [6]. ML is an essential technique to fully exploit AI technology. Because of its ability to learn and make decisions, machine learning is commonly referred to as artificial intelligence (AI), but it is actually a subset of AI. It was a phase of AI development up until the late 1970s. After that, it broke off to carry on developing independently. Nowadays, machine learning is used in many cutting-edge technologies and is an essential response tool for eCommerce and cloud computing [7].

C. Neural Networks

Conventional Machine Learning algorithms have the disadvantage of still being machine-like, despite their seeming complexity. They need a great deal of subject knowledge and human assistance, and they can only do what they were designed to do. Deep learning has slightly greater potential for both the general public and AI developers in this regard [8]. It makes it possible for developers to tackle issues in areas in which they lack expertise. Deep learning is a part of machine learning that, in practice, learns to represent the world as a layered hierarchy of concepts, where each concept is defined in connection to more abstract representations that are calculated in terms of less abstract ones. Deep learning can attain enormous power and flexibility because to this learning process [8].

D. Wearable Sensors

The term 'wearable fall detection' refers to the use of wearable devices that have sensors to detect and alert for falls in real-time. The wearable devices can be watches, wristbands, pendants, or belts; they typically have accelerometers, gyroscopes, or other sensors that can detect changes in movement and orientation. When a fall is detected, the wearable device sends an alert to a caregiver, family member, or emergency services.

In [9], the suggested system extracts characteristics of fall events from a mix of accelerometer and gyroscope sensors and classifies them using machine learning methods. In order to assess the effectiveness of the approach, the authors also suggest a brand-new dataset of fall events that includes activities that take place throughout the fall.

The study's findings demonstrate that the suggested system detects and recognizes falls with a high degree of accuracy. With a sensitivity of 97.92% and a specificity of 91.88%, the system detected falls with an overall accuracy of 94.44%. The authors concluded that their suggested system, which uses wearable sensors to detect falls and distinguish them from regular activities, is a promising method for detecting falls in actual environments.

E. Non-Wearable Sensors

While wearable devices have traditionally been utilized for fall detection, such as accelerometers and gyro sensors included in smart watches and pendants, there is growing interest in non-wearable fall detection systems that do not require people to wear any equipment.

Non-wearable fall detection systems employ a variety of sensing methods, including pressure, acoustic, and vision-based sensors. By examining the movements and poses of the people, vision-based sensors that use cameras can detect falls. Acoustic-based sensors employ microphones to record the fall's sound and recognize the distinctive acoustic patterns that are related to falls. Pressure-based sensors use floor sensors to identify pressure changes brought on by a fall's impact. One of the popular non-wearable methods that makes use of computer vision techniques for fall detection is camera-based fall detection. In these systems, cameras are installed in the surrounding area to record video feeds, which are then analysed to detect falls, such as in homes or healthcare institutions. A single camera and an image processing algorithm were utilized in a study [10] to develop a vision-based fall detection system. The program was created to identify falls by examining changes in the position, size, and form of the objects in the camera view. They got sensitivity and specificity values of at least 92 percent.

An acoustic-based fall detection system was suggested in a different study [11], which employed a smartphone to record and assess the sound of falls. The suggested approach analyses the sound of a fall using characteristics including energy, spectral centroid, spectral entropy, and spectral flatness. To create a classifier using the support vector machine (SVM) algorithm, the system was trained using a dataset of simulated falls and non-fall activities. The results showed that the proposed system achieved a high detection rate of 90.6% with a low false positive rate of 0.2 per hour.

A pressure-based fall detection system that used pressure sensors embedded in the floor to detect falls was suggested in a study [12]. The technology they used offers advantages over existing fall detection systems, such as having a passive sensor (no power source is required) and being fully undetectable because the sensor is built into the floor. The outcomes are contrasted with cutting-edge categorization methods. With a True Positive Rate of 94.4% and a False Positive Rate of 2.4%, fall detection performed well on the database.

F. Research Gaps

There is still a void in our understanding of fall detection notwithstanding notable breakthroughs. Although it has been the subject of substantial investigation, the kind of data sufficient to effectively distinguish between falls and activities of daily living (ADLs) to an almost certain degree of accuracy has not received as much attention. Therefore, additional research is required to close this gap and gain a better understanding of what parameters are closely linked to understanding falls.

In [13], their model was fed data from a gyroscope only. However, because it cannot distinguish between a fall and other activities that produce similar acceleration patterns, an accelerometer alone is insufficient for fall detection. It is challenging to distinguish between falls and other activities, such as sitting down, lying down, or activities of daily life that may cause comparable acceleration profiles according to [3]. Additional sensors, like gyroscopes or barometers, may be required to provide complementary data on changes in body position and altitude in order to effectively detect falls.

Additionally, [14] noted that "the complexity of human activities and the diversity of acceleration patterns make it challenging to develop a reliable fall detection algorithm using only acceleration data".

When deriving insights from data, it is important to get context. In order to fully comprehend the data, the interactions between variables, and the potential influences on the data, context is essential while evaluating data. Data without context may be incomplete, deceptive, or difficult to understand. Take a dataset, for instance, that reveals a sharp rise in sales of a specific product. Without context, one would infer that this growth is the result of an effective marketing effort when, in fact, it might be the result of a seasonal pattern or a shortage of a rival product.

It is quite common to find papers studying fall detection that forget about context. In [15], only 10 participants had data collected, which may not be enough to fully assess the proposed system's efficacy, especially in identifying falls in various settings and among diverse populations. In fall detection, where the model must be able to detect falls in various surroundings, with various people, and in various situations, a model that has been trained on a big, but more importantly, diverse dataset is more likely to generalize to new situations and settings.

Secondly, larger dataset may offer more variety in the kinds of recorded falls and non-fall events, enabling a more thorough knowledge of fall patterns and behaviours. By doing this, the model's precision and adaptability to novel circumstances can both be enhanced.

III. RESEARCH METHODOLOGY

A. Hardware

As the physical interface between the program and the outside world, hardware plays a crucial role in embedded systems initiatives. The capabilities of the software are heavily reliant on the hardware, for optimal function of our system we chose a microcontroller and peripherals that complied with the project’s power consumption, memory, ease of use, size, processing and budgetary restrictions. Additionally, hardware design is essential to ensuring the system complies with the project's power consumption, size, and budgetary restrictions. Hardware constraints may occasionally even influence the choices made for software design and execution.

i. Choosing microcontroller

A microcontroller is a miniaturized microcomputer designed to perform certain tasks for embedded devices, such accepting external signals and displaying radio data. It is essentially our project's brain. Peripherals are controlled by it.

Choosing our micro-controller left us with a lot of dilemmas, the first criteria we used was processing power. Processing power, commonly referred to as computing power, describes a computer system's capacity to carry out calculations and handle data. It is often assessed in terms of how quickly and effectively a system can carry out actions like running software programs, managing input and output, and carrying out mathematical computations.

The central processing unit (CPU), which is in charge of carrying out computations and carrying out commands, is mostly responsible for determining the processing power of a computer system. The CPU's clock speed, which is expressed in gigahertz (GHz), determines how quickly it operates. Faster processing power is typically associated with higher clock speeds, while other elements like the number of cores and the architecture's efficiency can also impact performance. We considered the Raspberry Pi with its 1.5 GHz quad-core 64-bit ARM Cortex-A72 processor, up to 8 GB of RAM, and up to 256 GB of microSD storage. It is perfect for uses like home automation and multimedia streaming because it supports technologies like Wi-Fi and Bluetooth networking. Raspberry Pi can be used for various ML jobs even though its processing power is lower than that of more potent desktop or server computers. Running small to medium-sized ML models and even Neural Networks, such speech or picture classification, on the Raspberry Pi is a good idea. These models can be developed on more powerful platforms, trained, and then deployed for inference on the Raspberry Pi. In contrast, due to their constrained processing speed and memory, Arduino boards are not normally intended for demanding machine learning (ML) workloads. Nevertheless, there are some situations in which Arduino boards can be utilized for machine learning, such as for straightforward categorization jobs or simple data pre-processing.

ii. Memory

When it comes to embedded systems, memory is a crucial factor. Embedded systems are frequently utilized in areas with limited power and space since they are typically built to carry out specific functions. The precise needs of the system and the tasks it must complete determine how much memory is needed for an embedded system. For instance, the system can need extra memory to retain intermediate findings if it must process vast volumes of data or run complicated algorithms. Similarly, the system can need extra non-volatile memory if it wants to store a lot of data or program instructions.

Raspberry Pi has greater RAM and non-volatile memory than most Arduino boards ranging from 2GB-8GB vs 32KB -512 KB of flash memory and 2KB - 32KB of RAM. Tilting the suitability for ML in Raspberry Pi's favour.

iii. Ease of Use

Given the time available to complete this project, it was very important to find out which microcontroller would be easier to use and set-up. Needing some knowledge of Linux to get the Raspberry Pi running was a bit discouraging. Meanwhile, Arduino had its simplicity and the abundance of user-friendly tutorials and materials going for it.

iv. Portability

It is important for the wearable device to be comfortable for the user to wear. That means it should be portable.

Due to their smaller size and reduced power needs, Arduino boards are generally more portable than Raspberry Pi. The tiny and lightweight designs of Arduino boards make them perfect for projects that need to be carried around. On the other hand, Raspberry Pi is typically bigger and uses more power to run. Even while certain Raspberry Pi versions, like the Raspberry Pi Zero, are small and portable, they still need a power source like a micro-USB connection and possibly other accessories like a monitor and keyboard. Arduinos meanwhile, use a battery of USB.

v. Availability and Support

One very important criterion is availability, project delays and cost overruns may occur if a microcontroller is not easily accessible. It is vital to choose a microcontroller that is widely available and easy to purchase from trustworthy sources. It can also ensure that any necessary replacement parts or components are readily available.

Another important factor is the availability of support materials such as manuals, tutorials, and forums. They can have a significant impact on the ease and speed with which development can be completed. It is vital to choose a microcontroller that includes useful information and recommendations. applying new features, understanding how to swiftly identify problems, and applying best practices and optimizations can all benefit developers. Furthermore, having a large and active development community has advantages. Developers can use this to contribute code, discover answers to questions, and learn from others' experiences. It may also ensure that the microcontroller is compatible with a wide selection of software libraries, tools, and add-ons.

vi. Cost

The cost of a microcontroller is an important consideration for many design projects. One of the most obvious reasons to consider the cost of a microcontroller is to ensure that the project remains within budget.

Another important reason is if the project is intended for mass production, the cost of each microcontroller can add up quickly. By selecting a cost-effective microcontroller, designers can keep the production costs low and increase their profit margins.

As a result, many design projects consider a microcontroller's price. The Arduino and Raspberry Pi platforms differ from one another in terms of price in some ways.

Generally speaking, Arduino boards are less expensive than Raspberry Pi boards. Depending on the type and features it delivers, an Arduino board might cost anywhere from $5 to $60. Using these criteria after comparing the two microcontrollers, the Arduino was decided to be the best option, due to its adequate performance in ML applications, cost-effectiveness, availability and support, and ease of use.

B. Selecting Peripherals

Peripherals in embedded systems are hardware components that are added to the main microcontroller or microprocessor to expand its capabilities. These peripherals can be either built into the microcontroller or added externally, and they provide a way to interface with the outside world, such as sensors, displays, or communication devices.

Choosing the peripherals was dependent on how we wanted the overall system to function and factors like the pinout, voltage levels and communication protocols. We envisioned a system whereby based on motion data, a ML model detects whether a fall has occurred or not, and when it had, a call for help is made. To get this motion data, we decided to use an accelerometer (specifically the ADXL345) and for the signal, a GSM module. The accelerometer reveals details about an object's motion or orientation by measuring change in velocity. The operational principle can be summarized in Figure 1.

i. Sensor data

The most important criteria for picking peripherals are what they are being used for.

In fall detection, the most important variables to consider are acceleration in all three axes, tilt and the direction of the magnetic field. So, we need the peripherals to include the following; accelerometer, gyroscope and magnetometer. Gyroscopes track rotational motion or angular velocity around several axes. They can be used to detect changes in orientation and offer information on the rate of rotation of an item.

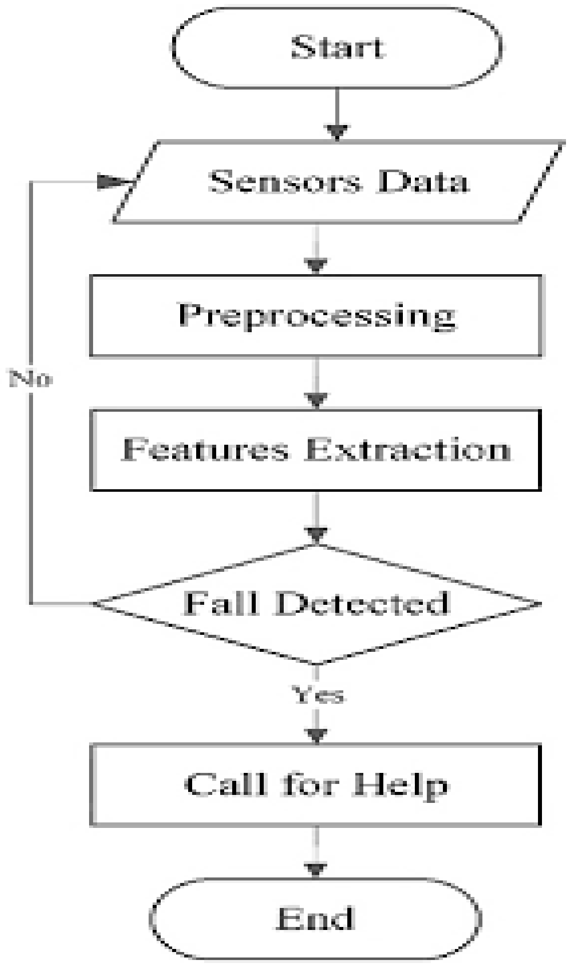

Figure 1: Flowchart for the development of the Fall detection system

Gyroscopes are essential for activities like stabilizing a camera or managing an aircraft's attitude.

For precise measurements of an object's motion in three dimensions, the IMU combines data from multiple sensors. The IMU can follow changes in location, velocity, and orientation by continually measuring acceleration and rotation. Applications requiring accurate motion tracking and control, such as navigation, robotics, virtual reality, motion capture, and many more areas, all depend on this data.

When training our model, we decided to use an established dataset called the SisFall dataset on Kaggle. We did this instead of gathering data ourselves from ourselves because of the following reasons; challenges with removing noise, the time taken to gather enough data and complexities involved in storing the data. Our approach gave us access to a standard benchmark for our model training, enabled us to develop the system faster and access to proven quality data. As a result of the data we were using for training we decided to use an ADXL345 accelerometer as our sensor. The ADXL345 is a small, thin, fully functioning 3-axis accelerometer with signal-conditioned voltage outputs. The full-scale range of the device's acceleration detection is ±16g.

Fixed plates and moving plates make up the accelerometer's fundamental construction. The capacitance between stationary plates and movable plates changes as acceleration is applied along an axis. As a result, the output voltage amplitude of the sensor increases in proportion to the acceleration. This module's specifications and features include an on-board LDO voltage regulator, a 3V–6V DC supply voltage, integrated MOSFET-based voltage level converter, it has the ability to interact with both 3.3V and 5V microcontrollers. Ultra-Low Power: SPI and I2C interfaces, Tap/Double Tap Detection, Free-Fall Detection, 40uA in measurement mode, and 0.1uA in standby at 2.5V The measurement range is ±16g, and the values are -235 to +270 for X, -240 to +260 for Y, and -240 to +270 for Z. An image can be found in Figure 2.

Call for help

The other function the controller is not capable of performing is the communication with the outside world. Peripherals that enable this are called communication modules, alongside them a buzzer, LED and charging module were included to sound an alarm, call bystanders through light and charge the system respectively. The HC-05 Bluetooth module was the first communication module that was taken into consideration for the project.

Figure 2: ADXL345 accelerometer

The research (included in the planning stage) revealed that previously created systems lacked 'fast' communication methods as a result, which was discovered later on. This observation was made in the sense that the device's primary goal, to immediately alert first responders in the event of a fall, was not really achieved as the Bluetooth module needed to send data to a mobile device first and foremost before the user could call the medical staff. This finding led to the selection of the SIM808 GSM/GPRS module as a new communication module.

The SIM808 Module from SIMCOM serves as the basis for this GSM and GPS modem. The SIM808 uses the Global Positioning System (GPS) for satellite navigation and supports GSM/GPRS Quad-Band networks. It is based on the most modern GSM/GPS module from SIMCOM. It contains a built-in charging circuit for lithium-ion batteries and a very extended standby life. It is therefore ideal for uses that require rechargeable Li-Ion batteries. In addition, it has 66 acquisition channels and 22 tracking receiver channels, both of which improve the sensitivity of its GPS reception. Additionally, A-GPS, which can be utilized for indoor localization, is supported. AT instructions are used to control the module using UART. module (ElectronicsComp.Com)., 2023). The SIM808 is shown in Figure 3.

Figure 3: SIM808 Module

We also included a buzzer to sound an alarm whenever a fall happens and an LED. A buzzer or beeper is an auditory signalling device that can be mechanical, electromechanical, or piezoelectric (The Free Dictionary, 2015). They can be active or passive. For the project, an active buzzer was used. They are commonly offered in voltage ranges between 1.5V to 24V. To generate a sound, all that is needed to do is supply a DC voltage to the pins. Active buzzers are polarized. Similar to an LED and a capacitor, the longer pin is connected to the positive terminal.

i. Charging Module

The TP4056 charging module was employed. For lithium-ion and LIPO batteries, this is a linear charger. The module can recharge single-cell batteries, and because of its ability to create 4.2V, it is particularly well suited for charging 18650 cells and other 3.7V batteries. The charger can be powered by any source that provides a 5V and 1A supply, whether a USB source, wall adapter, or other.

When operating at high power levels or in environments with high ambient temperatures, thermal feedback automatically modifies the charge current to control chip temperature. A resistor can be used to externally control the charging current, while the charging voltage is fixed at 4.2V. The TP4056 module automatically switches to a low current mode, reducing the battery leakage current to less than 2uA, when the input voltage (from the AC adapter or USB power) is interrupted.

C. Circuit Design

The circuitry of the device must be clearly defined in order for it to operate as intended. Tinker CAD was used for the circuit design. This Autodesk-developed free online 3D modelling tool is accessed using a web browser (Herrman, n.d.). Proteus 8 was initially supposed to be utilized, but it was abandoned in favour of Tinker CAD since it lacked libraries for some of the components, such as those for the sensors (the ADXL345 accelerometer and the LSM303 IMU module).

The HC-05 Bluetooth module served as the communication module in the initial circuit design. The circuit design for the system is shown in Figure 4.

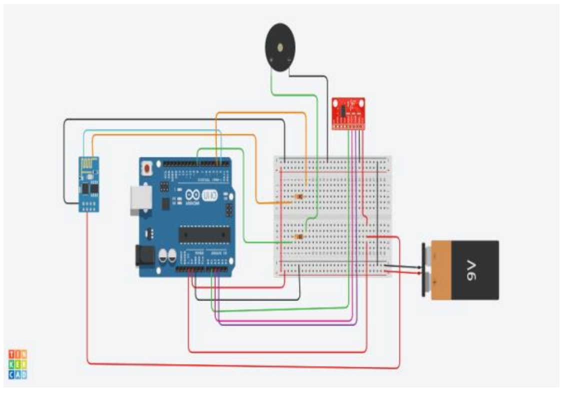

Figure 4: Circuit design for the connection of the components involved in the fall detection system

A new circuit was created after the modifications and improvements made to the HC-05 Bluetooth module to accommodate for those factors.

The selection of the design and component steps was followed by the implementation phase. Components were ordered and delivered on time to avoid project delays. At this stage, a test version of the fall detection and alarm system was developed. The creation of programming code or algorithms for the microcontroller—in this case, the ATMega328PU from the Arduino UNO—is another thing that happens. To make it easier to spot and address issues, it was made sure that the algorithm was clear-cut and simple to understand.

D. Software Implementation

The success of this project is massively dependent on the ML model. This is because the regular detection of falls is a function of the accuracy of the model. A high model accuracy on new data means that it has successfully learnt the relationship between the features and output. Training a model is essential to its accuracy because it enables the model to learn patterns, relationships, and representations from the available data.

The model was trained with access to a lot of labelled data. This is to help it derive pertinent features and comprehend underlying patterns and trends by watching and examining this data. High accuracy is achieved by the generalization of the model's information from training instances to new ones. The objective is to identify the underlying patterns that may be applied to fresh, comparable situations rather than simply memorizing the training data.

The Jupyter Notebook was used for the majority of the model's training, testing, and evaluation. Creating and sharing documents with live code, equations, pictures, and narrative text is possible with this open-source web application. It is a popular tool for data analysis, scientific computing, and machine learning.

Its compatibility with the Python programming language was key in its choice. This is because of Python's simplicity, abundance of ML libraries, adaptability, strong community support, simplicity in integrating with other technologies, and ability to*iii.* balance performance and productivity. *iv.*

i. Import dataset

The lifeblood of ML is the data. Quality data is needed to make good inferences. The dataset the model was trained on was the SisFall dataset from Kaggle, as stated earlier. To provide enough information to the model about a 1000 sets of data points were fed into the model.

Also, to enable it distinguish between falls and not falls, an equal number of samples were used.

The pandas library which is useful for manipulating data was imported to help read the dataset into the code.

ii. Filter data

The SisFall dataset has over a million samples. This is good because the more samples fed into the model, the better it can generalize. However, larger datasets tend to take longer to train. To save training time, the model was reduced to 2000 samples. 1000 samples of falls and another 1000 samples of non-falls.

An equal number of affirmative and negative samples were used to prevent a class imbalance. Classification models frequently perform poorly for the minority class when there is a class imbalance because they are biased toward the dominant class. So, bias was reduced by giving the model an equal representation of both classes by balancing the amount of affirmative and negative samples. This makes the model more accurate.

After filtering the data, the dataset was segmented into the features and dependent variable vectors. Features are the characteristics or properties that reveal details about the instances or data points in a dataset. They record various data facets or qualities that are pertinent to the learning activity. With regards to this project, the features are the data from the sensor. We also removed data our sensors wouldn't have like rotational data.

Whereas, the dependent variable vector is the model's output i.e. whether the individual has fallen or not. Generally, it could either be a set of continuous numbers or a category.

Pre-process data

In machine learning, the term "data pre-processing" refers to the procedures and methods used to organize and change raw data into a format appropriate for analysis and model training. It plays a crucial role in the machine learning pipeline and entails a number of tasks, such as normalization, transformation, and data cleansing. Improving data quality, managing missing values, getting rid of inconsistencies, and preparing the data for machine learning algorithms are its objectives. Encoding categorical data, dividing the data into training and test sets, and feature scaling were the data pre-processing techniques used in this study.
Categorical data refers to data that is not in numerical form. Data used to name or categorize objects is referred to as categorical data since it reflects several categories or groups. It is made up of qualitative variables that describe the traits or qualities of the subjects under investigation. Nominal or qualitative data are other names for categorical data. In categorical data, each observation is categorized into a single category or class; the categories themselves have no inherent numerical value or order. However, computers cannot understand qualitative data e.g. 'tall', 'short', 'fall', 'not fall' etc. Therefore, it has to be converted into a form the model would understand. The majority of statistical models and machine learning algorithms operate on numerical data. We make it possible for these models to effectively process and interpret the data by encoding categorical variables into numerical form. Algorithms may compare data, carry out mathematical operations, and find patterns in the data by converting category data to

numerical values. The output values fall and not fall are converted to 0s and 1s.

The main goal of separating the dataset is to assess a model's performance on unobserved data. We use one part, usually the majority of the samples, to train the model. That portion of the data set is called the training set, meanwhile, the ‘unobserved data’ is the test set.

The main reason this is done is to measure how effectively the model generalizes to new, untested situations by using a subset of the data as a test set. The model's performance in real-world situations is estimated by this evaluation, which also helps determine the model's predictive or categorical accuracy. Lastly, but not least dividing the dataset is done to avoid leakage. Usually, before the training set is fed into the model it undergoes several processing stages. To avoid information from the test set mistakenly affecting the outcome. It is separated from the dataset. Also, by using this test set. We can compare the accuracies of different models to find out which is more suitable for the problem at hand. The entire dataset was split 80/20 into training and test set. Afterwards, we scaled our features. Feature scaling is a data pre-processing method used in machine learning to standardize or normalize a dataset's numerical features. This is done eliminate biases and make sure that no one aspect predominates the learning process, it seeks to scale all features to a similar level.

The size of the features can have a major impact on the performance of the model in many Machine Learning techniques, including those based on distance computations or gradient descent.

v. Training the Model

There are several kinds of Machine Learning (ML) models. The factors considered in choosing a model were; the problem type, interpretability, robustness to noise, model complexity and performance metrics.

a. Problem Type

Broadly speaking, ML can be grouped into two types of problems namely; regression and classification. Regression involves the prediction of a continuous numerical variable. The target variable is a number with actual meaning to the computer.

Meanwhile, classification involves the prediction of categories. Its object is to use labelled to data to predict the label of unseen data.

The aim of the model in this project is to ‘label’ movement as falls or not falls. Therefore, classification ML models were decided upon.

b. Interpretability

It was of utmost importance that the model used could represent the relationship between the features and output as a mathematical equation. This is because implementing the model on Arduino becomes a problem of simple computation and not one of intense programming. Arduino is not a native language for training ML models, so to avoid any difficulty. We’d get the equation after building the model with Python elsewhere, using the Arduino just for inference.

c. Robustness to Noise

A very desirable characteristic of machine learning (ML) models and systems is robustness to noise. It speaks to a model's capacity for sustaining reliable predictions or actions in the face of noise or other disturbances in the input data. Measurement mistakes, outliers, missing data, and irrelevant features are just a few of the many potential sources of noise that can have a big impact on how well ML models perform and how reliable they are.

d. Model Complexity

Simpler models are frequently simpler to understand, train, and troubleshoot. Though more accurate, complex models can be computationally expensive and prone to overfitting.

e. Performance Metrics

This involves choosing models based on their ability to make accurate predictions on unseen data. Two metrics we considered are accuracy and f1 score. Two metrics that show how well a machine learning model can categorize observations into classes are accuracy score and F1 score. How many of our predictions were accurate is indicated by the straightforward metric known as accuracy. Another statistic applied to multi-class classification models is the F1 score. It is preferred over accuracy because it offers trustworthy results for a variety of datasets, whether or not they are imbalanced.

After considering the above criteria, it was decided that the Support Vector Classifier would be the most appropriate for this project. However, it has two different types namely; linear and kernel Support Vector Classifier. To determine which to use, their performance metrics were evaluated. The linear SVM had an accuracy and f1 score of 0.8888 and 0.886 respectively meanwhile, the RBF/Kernel SVM had an accuracy and f1 score of 0.87 and 0.86. Therefore, the linear SVM performed better on new observations.

vi. Arduino Implementation

On Arduino, the model was implemented alongside the operations of the sensors, GSM modules and buzzers. The following steps were undertaken to achieve this; importing libraries, configure the GSM module, set weights and biases, build model, get data from sensors, pre-process data and parse new data into the model.

a. Importing Libraries

For software development, library imports are crucial. They make code organization, work simplification, time savings, code sharing, reliability, and learning materials all possible. They provide pre-made functions for typical operations, bundle reusable code, and simplify low-level details.

Libraries help to add new functionality to sketches, such as the ability to interact with hardware or manipulate data. They simplify the process of interacting with hardware. The libraries imported were:

- Wire.h: To aid serial communication with external devices.
- Adafruit_Sensor.h: To help control all sensors from Adafruit
- GSM.h: To help control the GSM module.
- Adafruit_ADXL345: To help control the accelerometer specifically.

b. Configure GSM module

To configure the GSM module, an instance of the GSM library was created, providing functions and APIs to interact with GSM modules for enabling data connections, placing calls, and sending and receiving SMS messages.

c. Configure the accelerometer

The accelerometer was configured using an instance of the Adafruit_ADXL345 library to operate the ADXL345 sensor module, which is a three-axis accelerometer capable of measuring acceleration along multiple directions. This library offers methods for interacting with the sensor, retrieving acceleration data, configuring power-saving modes, enabling interrupts, and setting measurement ranges. The getEvent function was employed to obtain real-time motion data.

d. Configure buzzer

Additionally, the buzzer was set up by defining the pin it was connected to and configuring it as an output using pinMode. A condition was established to send a HIGH signal to the buzzer, triggering it as needed.

e. Pre-process data

The data from the sensor needs to have a similar distribution to the one the model was trained on.
The mean and standard deviation of the training set is obtained and used to scale the sensor data before it is passed into the model.

f. Inference/ Prediction

Finally, the model was implemented as a prediction function, utilizing the weights and biases derived from the model trained in Jupyter. These parameters were stored in an array and applied within an equation to determine the best fit for classification. Based on the output values, the sensor data was classified as either a fall or a non-fall event.

g. Activate buzzer

When a fall is detected, a 'HIGH' signal is sent to the buzzer and a message is sent by the GSM module to a number chosen by the designer.

IV. RESULTS AND DISCUSSION

The findings of a thorough examination into fall detection using machine learning approaches are presented in this chapter. The goal of this project was to create a fall detection system that was precise and trustworthy while utilizing AI algorithms. The effectiveness of various machine learning models was assessed through rigorous experimentation and analysis with the goal of enhancing the detection abilities of such systems.

A diversified dataset made up of actual fall events and non-fall activities was used for the studies. The dataset was meticulously chosen to guarantee its variety and representativeness, including a range of scenarios and environments. On the basis of this dataset, machine learning models were trained to identify the patterns and traits connected to falls while separating them from other activities.

The performance of our models for the fall detection system are then discussed. By giving a complete examination of the investigation's findings, this chapter offers insightful information on fall detection utilizing machine learning techniques.

a. Learning Curve

A conventional learning curve's x-axis reflects the number of training samples or iterations, while the y-axis represents a performance parameter like accuracy or error rate. The learning curve depicts how the model's performance improves or stabilizes as more data or iterations are added.

Per the learning curve, our model performed quite well on the training and test set without overfitting. The accuracy on the training set started at 100 percent and that of the test set at about 0, but over more epochs both converged at 91 and 86 percent respectively. This shows that the model stopped memorizing the training data as time went on and began to generalize better on unseen data. Showing that ML is viable approach to detecting falls. The curve is shown in Figure 5.

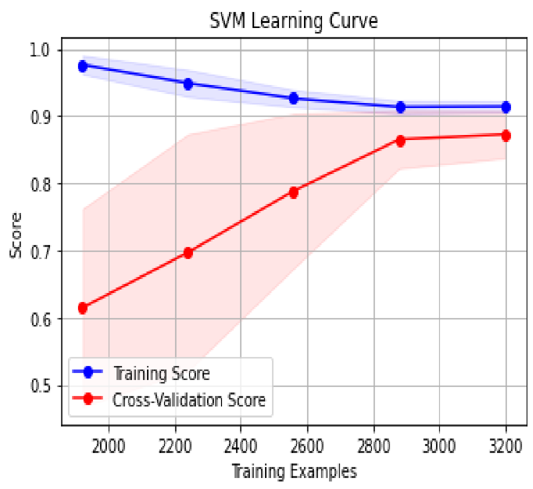

Figure 5. Learning Curve of the model

b. Colour Map

When building models, it is important to have a feel for how it makes decisions. To this end, we employed a colour map to understand how our model classifies an action as a fall or an ADL. The colour map is used to show the decision boundary of a machine learning model. The decision boundary is the dividing line or surface that separates multiple classes or categories in the input space. It is the moment when the model switches from predicting one class to predicting another. Though most data points are correctly classified, the boundary between both classes is a straight line due to our use of a linear model. Consequently, you would notice that it doesn't define both classes appropriately because many wrongly classified data points can be found on either side of it. Another thing is, our dataset was of a high dimensionality and had to be reduced to 2D through Principal Composition Analysis (PCA) to make the visualization, as a result, it is hard to understand which combination of our features PC1 and PC2 are. The colour map can be seen in Figure 6.

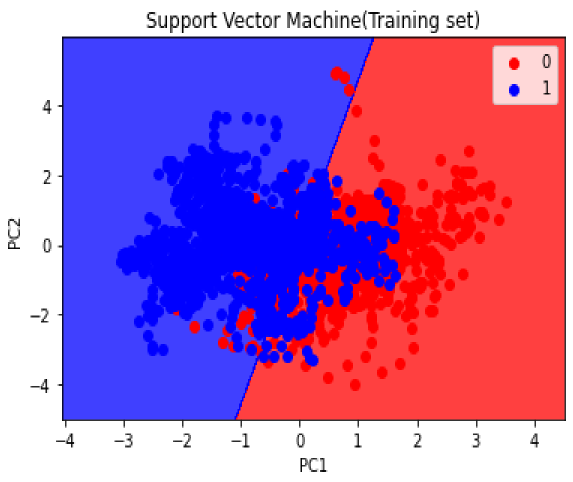

Figure 6: Decision Boundary of the model

c. *ROC curve*

A graph called the ROC curve shows how well a classification model can distinguish between two classes. It displays the true positive rate (sensitivity) in comparison to the false positive rate (1-specificity) at different threshold levels. The curve of a successful model will be toward the upper-left corner, signifying low false positives and high sensitivity.

By contrasting sensitivity (true positive rate) against false positive rate (1-specificity) at various threshold levels, the ROC curve illustrates a classification model's capacity to discriminate between two classes. In the context of fall detection, it is critical to comprehend how sensitivity and specificity are balanced. High sensitivity guarantees that most falls are recognized, but it may increase false positives, resulting in unneeded alarms or actions. Improving specificity (cutting false positives) may reduce sensitivity, risking missing falls, which could have serious consequences. Given the various contexts in which this technology could be used it is important to understand the trade-offs at play here.

An ideal AUC (Area under the curve) tends to 1, indicating a high ability to tell apart both events (falls and ADLs). However, AUC does not show performance at individual thresholds, it is useful to focus on specific locations on the ROC curve where sensitivity is preferred. The curve is shown in Figure 7.

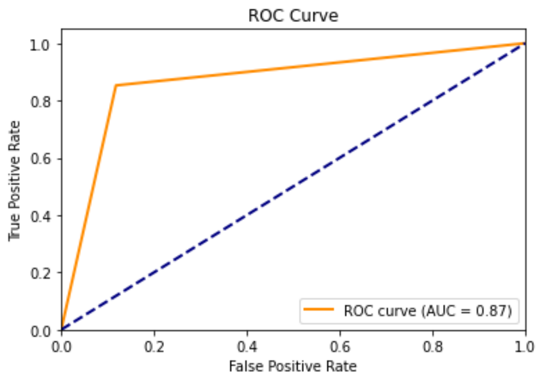

Figure 7: Area under the ROC curve

d. *Confusion Matrix*

A classification model's performance is assessed using a table called the confusion matrix. It is extensively used in statistics and machine learning to understand the prediction accuracy and error rates of a model. A confusion matrix is often a square matrix that contains both the actual and expected classes or labels. It is divided into four major sections:

True Positives (TP) occur when the model correctly predicted the positive class. The model detects falls so the positive class is ‘fall’ and the negative class is ‘not fall’.

True Negatives (TN) are situations in which the model correctly predicted the negative class.

False Positives (FP) occur when the model incorrectly predicted a class and the actual class is not that class.
False Negatives (FN) occur when the model incorrectly predicted the negative class.

The image of the table generated by the code is shown in Figure 8.

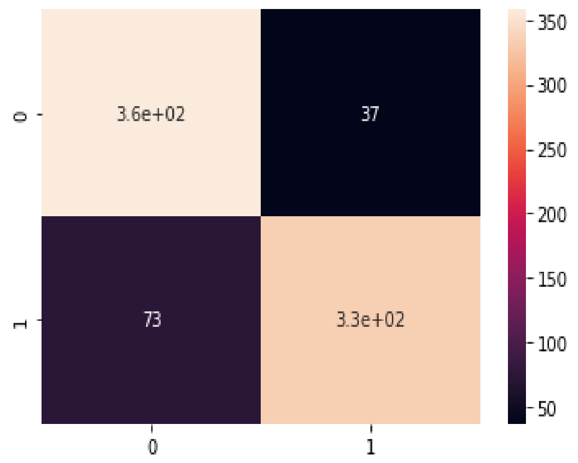

Figure 8: The Confusion Matrix

The confusion matrix shows that while our model properly identified 360 falls as falls, it misclassified 73 non-falls as falls, and it correctly identified 330 non-falls but misclassified 37 falls as non-falls. As per the model's predictions, there were 360 True Positives, 73 False Positives, 330 True Negatives, and 37 False Negatives.

Evaluation measures including the accuracy score, precision, recall, and f1 score can be calculated from these statistics. One performance metric that shows how well a model predicts the future overall is accuracy. It is computed as the sum of the true positives and true negatives divided by the total number of forecasts. The percentage of accurate positive forecasts among all positive forecasts is known as precision. It highlights how important positive forecasts are.

Recall, sometimes referred to as sensitivity or true positive rate, quantifies the percentage of correctly anticipated positive cases among all actual positive instances. It centres on how accurate optimistic predictions are.

The F1 score combines precision and recall into a single metric by considering their harmonic mean. It offers an impartial evaluation of the model's performance. Our model's f1 score was 85.91, its precision was 83.16, and its recall was 90.68. especially good at spotting real positives, (i.e. falls).

CONCLUSION

This project successfully designed and implemented a fall detection system using machine learning techniques. By employing the SisFall dataset for model training and testing, the system achieved an impressive accuracy of 86.25% and demonstrated strong performance metrics, including a recall of 90.68%, precision of 83.16%, and an F1 score of 85.91%. These results highlight the system's ability to reliably detect falls while minimizing false positives, making it suitable for real-world applications.

The fall detection system's innovative design—integrating robust hardware, a trained machine learning model, and communication peripherals—ensures timely alerts to caregivers, promoting safety and independence for 'at-risk' individuals. While the results are promising, further research is recommended to enhance the system's adaptability to diverse environments and populations, improve robustness against noise, and explore alternative data sources for increased accuracy. The development of this system contributes significantly to the field of wearable technology and machine learning, opening avenues for improved patient care and accident response.

ACKNOWLEDGMENT

We would like to thank God for giving us the opportunity to carry out this study and Afe Babalola University for providing the tools and assistance required for this investigation. Dr. A. O. Akinwumi deserves special recognition for her important advice and knowledge during the study. The SisFall dataset is also credited with making it easier to train and assess the machine learning model. Last but not least, we appreciate the support and helpful criticism we received from our peers and families during the study.

REFERENCES

[1] Hoskin, A. (1998, April 01). Europe PMC. Retrieved from https://europepmc.org: https://europepmc.org/article/med/9592923.

[2] World Health Organization. (2023, March 30). Retrieved from https://www.who.int/news-

room/fact-sheets/detail/falls#:~:text=A%20fall%20is%20defined%20as,though%20most%20are%20non%2Dfatal.

[3] Bourke, A. K. (2010). GamblEvaluation of waist-mounted tri-axial accelerometer-based fall-detection algorithms during scripted and continuous unscripted activities. Journal of Biomechanics, 3051-3057.

[4] Mahesh, B. (2018). Machine Learning Algorithms - A Review. International Journal of Science and Research (IJSR), 381-386.

[5] Bush, R. R. (1951). A mathematical model for simple learning. Psychological Review, 58(5),313–323.

[6] Robert Koch. (2023, March 2). https://www.clickworker.com/. Retrieved from clickworker.com: https://www.clickworker.com/customer-blog/history-of-machine-learning/

[7] Foote, K. (2021, December 3). dataversity. Retrieved from https://www.dataversity.net: https://www.dataversity.net/a-brief-history-of-machine-learning/

[8] Mahapatra, S. (2018, March 21). Towards Data Science. Retrieved from towardsdatascience.com: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063

[9] F. Hussain, F. H.-u.-H. (2019). Activity-Aware Fall Detection and Recognition Based on Wearable Sensors. IEEE Sensors Journal, vol. 19, no. 12, 4528-4536.

[10] Adrian Nunez-Marcos, G. A.-C. (2017). Vision-Based Fall Detection with Convolutional Neural Networks. Hindawi,Wireless Communications and Mobile Computing 2017(1), 1-16.

[11] Cheffena, M. (2016). Fall Detection Using Smartphone Audio Features. IEEE Journal of Biomedical and Health Informatics, vol. 20, no. 4, 1073-1080.

[12] Minvielle, L. M. (2017). Fall detection using smart floor sensor and supervised learning. 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). Jeju, Korea (South): IEEE.

[13] Tang, C.-H. O. (2015). Fall Detection Sensor System for the Elderly. International Journal of Advanced Computer Research.

[14] Su, Y. &. (2015). Fall detection in the elderly population by using wearable devices: A review. Sensors, 15(6), 11882-11910.

[15] Santoyo-Ramón, J. C.-G. (2018). Analysis of a smartphone-based architecture with multiple mobility sensors for fall detection with supervised learning. Sensors 18, 1155.