NÖHÜ Müh. Bilim. Derg. / NOHU J. Eng. Sci., 2024; 13(2), 582-592

Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences

Araştırma makalesi / Research article

www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

Exploring the performance of PySpark and Scikit-Learn libraries in developing fall detection systems

Düşme algılama sistemlerinin geliştirilmesinde PySpark ve Scikit-Learn kütüphanelerinin performansının araştırılması

Erhan Kavuncuoğlu[1,*]

[1] *Cumhuriyet Üniversitesi, Bilgisayar Teknolojileri Bölümü, 58840, Sivas Türkiye*

Abstract

Falls pose a significant risk, often resulting in serious injuries and reduced quality of life for the elderly population. Accurate and effective fall detection systems can play an important role in reducing these risks. This study presents a comparative analysis of the performance of PySpark and Scikit-Learn libraries in the development of fall detection models. Using both libraries, fall detection models were built using five popular machine learning algorithms, including logistic regression, gradient boosting classifier, random forest, support vector machine and decision tree. The models were evaluated using comprehensive metrics (accuracy, sensitivity, specificity, confusion matrix). In the study, 26 different features were extracted from the Sisfall dataset consisting of falls and activities of daily living data in five main categories: basic statistical features, frequency domain features, time series features, motion features and relational features. These features were incorporated into the fall detection models to increase their ability to recognise falls. The findings show that both PySpark and Scikit-Learn offer powerful and effective results in fall detection. The highest performance rates of both libraries were achieved by logistic regression. Furthermore, PySpark exhibited slightly longer training times than Scikit-Learn, which performed better in the test. In conclusion, this study contributes to the development of fall detection systems to improve the safety and well-being of the elderly and contributes to the literature by providing a new feature extraction method.

Keywords: Fall detection, Artificial intelligence, Machine learning, PySpark, Scikit-Learn

Öz

Düşmeler, genellikle ciddi yaralanmalara ve yaşlı nüfusun yaşam kalitesinin azalmasına neden olan önemli bir risk oluşturur. Doğru ve etkili düşme tespit sistemleri, bu riskleri azaltmada önemli bir rol oynayabilir. Bu çalışma, düşme tespit modellerinin geliştirilmesinde PySpark ve Scikit-Learn kütüphanelerinin performansını karşılaştırmalı bir analiz sunmaktadır. Her iki kütüphane de kullanılarak, lojistik regresyon, gradyan arttırma sınıflandırıcısı, rastgele orman, destek vektör makinesi ve karar ağacı dahil olmak üzere beş popüler makine öğrenme algoritması kullanılarak düşme tespit modelleri oluşturuldu. Modeller, kapsamlı metrikler (doğruluk, duyarlılık, özgüllük, karışıklık matrisi) kullanılarak değerlendirildi. Çalışmada düşme ve günlük yaşam aktivite verilerinden oluşan Sisfall veri setinden 26 farklı özellik beş ana kategoride çıkarıldı: temel istatistiksel özellikler, frekans alanı özellikleri, zaman serisi özellikleri, hareket özellikleri ve ilişkisel özellikler. Bu özellikler, düşme tespit modellerine düşmeleri tanıma yeteneklerini artırmak için dahil edildi. Bulgular, hem PySpark hem de Scikit-Learn'ün düşme tespitinde güçlü ve etkili sonuçlar sunduğunu göstermektedir. Her iki kütüphane de en yüksek performans oranlarına lojistik regresyon ile ulaşılmıştır. Ayrıca, PySpark, testte daha iyi performans sergileyen Scikit-Learn'e göre biraz daha uzun eğitim süreleri sergilemiştir. Sonuç olarak, bu çalışma, yaşlıların güvenliğini ve refahını artırmak için düşme tespit sistemlerinin geliştirilmesine katkıda bulunduğu gibi yeni bir özellik çıkarma yöntemi sunarakta literatüre katkıda bulunuyor.

Anahtar kelimeler: Düşme Algılama, Yapay zeka, Makine öğrenmesi, PySpark, Scikit-Learn

1 Introduction

Fall detection is an important research area, particularly for the well-being and safety of the elderly population in the domains of fall sensing, machine learning, and data analysis. This study aims to compare the performance of fall detection models using PySpark and Scikit-Learn libraries utilizing the Sisfall [1] dataset.

PySpark is an open-source cluster computing framework that provides a distributed computing environment for large-scale data processing. It is suitable for processing big data and complex computations by offering various machine learning algorithms and tools [2-6]. Patel et al. [7] focused on a real-time scalable data collection process for COVID-19 data and highlighted PySpark as a useful tool for this process. The authors emphasized the comprehensive and efficient capabilities of PySpark for processing large datasets in parallel clusters, specifically in machine learning applications. Another study by Gupta et al. [8] presented

* Sorumlu yazar / Corresponding author, e-posta / e-mail: ekavuncuoglu@cumhuriyet.edu.tr (E. Kavuncuoğlu)
Geliş / Recieved: 10.10.2023 Kabul / Accepted:12.02.2024 Yayımlanma / Published: 15.04.2024
doi: 10.28948/ngumuh.1388789

Apache Spark and a deep learning-based big data analysis framework using PySpark. The authors performed experimental analyses on real-world datasets and demonstrated the effectiveness of their framework compared with traditional big data analysis techniques. Additionally, Rothauge [9] discussed the use of PySpark as an interface for the Alchemist system. Alchemists have enabled Apache Spark to achieve better performance by providing interfaces with high-performance computing libraries for large-scale distributed computations. The author highlights the deployment options and data transfer times when using PySpark with Alchemist. These studies demonstrate the versatility and efficiency of PySpark in large-scale data processing and machine learning tasks, emphasizing its use in real-time data collection, integration with other data analysis frameworks, etc.

On the other hand, Scikit-Learn is a widely used machine learning library in Python that provides comprehensive tools for data analysis and modelling [3-6], [10]. Abraham et al. [11] focused on the use of Scikit-Learn for machine learning in neuroimaging. The authors demonstrated how Scikit-Learn can perform key analysis steps in functional neuroimaging applications, highlighting its versatility in brain studies. Another study by Buitinck et al. [12] discussed the design choices of Scikit-Learn Application Programming Interface (API). The authors explained that Scikit-Learn is designed to be simple, efficient, and accessible to non-experts, emphasizing its reusability in various contexts. Auti et al. [13] mentioned the use of Scikit-Learn for data preparation in data mining, specifically for normalization, standardization, and handling outliers or missing data. Furthermore, a review article by Hao and Ho [14] provides an overview of Scikit-Learn as a machine-learning package in Python. The authors highlight the comprehensive list of machine learning methods included in Scikit-Learn and their adherence to unified data and modelling procedure rules, emphasizing the ease of use for educators and behavioral statisticians. Overall, these studies showcase the varied applications and functionalities of Scikit-Learn in machine learning and data analysis across different domains, highlighting its versatility, simplicity, and efficiency.

In this study, several commonly used machine learning algorithms, such as logistic regression, gradient boosting classifier, random forest, support vector machine, and decision tree, were employed to develop the fall detection models. These algorithms were chosen for their effectiveness in classification tasks, ability to handle both numerical and categorical data, and high accuracy achievements in fall detection [15-19].

The performance of the models was evaluated using various metrics, such as accuracy, precision, recall, confusion matrix, training time, and testing time, obtained through PySpark and Scikit-Learn. By comparing the performance of PySpark and Scikit-Learn based on these metrics, this study aimed to determine which library is more successful in fall detection. The findings of this research will guide researchers and practitioners in selecting the most suitable library for developing fall detection systems.

The following sections explain the methodology used in this study, present the obtained results, discuss the implications of the findings, and provide a comprehensive analysis of PySpark and Scikit-Learn in the context of fall detection.

2 Materials and methods

In this study, the Sisfall dataset, which includes falls and activities of daily living (ADL) obtained from an experiment conducted by the Systemic research group, was utilized. The collaboration was established with 38 volunteers, consisting of elderly and young adults. The elderly group comprised 15 participants (8 males and 7 females) who were in good health and independent. The young adult group consisted of 23 participants (11 males and 12 females). Each participant repeated 15 different fall movements and 19 different ADLs, five times. Within the elderly group, certain ADLs (activities numbered 6, 13, 18, and 19) were not provided with medical advice, and some activities could not be carried out due to personal barriers. Additionally, one participant simulated fall movements and ADLs due to a judo experience (Table 1).

Data were collected using a specialized wearable device. The equipment included a Kinetis MKL25Z128VLK4 microcontroller produced by NPX in Austin, Texas, USA, an analog device ADXL345 accelerometer (with a range of ± 16 g and 13-bit analog-to-digital conversion freedom), a Freescale MMA8451Q accelerometer (with a range of ± 8 g and 14-bit ADC), an ITG3200 gyroscope (with a range of ±2000°/s and 16-bit ADC), and an SD card. The device was mounted on the participant's waist and powered by a 1000 mA/hr general-purpose battery. The device was positioned using a single accelerometer system owing to its ability to differentiate between different activities. The device was set up to record the original sampling frequency (200 Hz).

This research focused on training algorithms using PySpark and Scikit-Learn libraries with activity data from 23 young participants to detect and prevent falls. The participants were asked to perform 15 different fall movements and 19 ADLs. In total, 3532 observations were generated during the experiment. Upon inspection of the dataset, it was observed that 19 ADLs had 5 repetitions, with 4 of them having only 1 repetition. Similarly, 15 fall movements were collected with 5 repetitions. As a result, the total number of observations should have been 3542; however, only 3532 observations were obtained because of missing repetitions in some cases. Data erasures occurred because of missing repetitions, such as the absence of a repetition in the 17th ADL for SA15; the absence of a repetition in the 6th ADL for SA17; and the absence of repetitions in the 1st fall movement, 6th ADL, 10th fall movement, and 17th ADL for SA20, resulting in a total of 10 missing observations. Therefore, individuals with missing repetitions were considered test data to address imbalances in the training set.

The method section applied three main stages for fall detection using a dataset consisting of 3532 records from 23 young volunteers: pre-processing, feature extraction,

Table 1. SisFall Dataset: List of Falls and ADLs

Falls				ADLs		
No	**Description**	**Cause of Fall**	**Trials**	**No**	**Description**	**Trials**
1	Falling forward while walking	Slip	5	1	Slow walking	1
2	Falling backward while walking	Slip	5	2	Fast walking	1
3	Lateral falling while walking	Slip	5	3	Slow jogging	1
4	Falling forward due to a trip	Trip without stumbling	5	4	Fast jogging	1
5	Falling forward while jogging	Trip	5	5	Ascending and descending stairs slowly	5
6	Vertical Falling while walking	Fainting	5	6	Ascending and descending stairs quickly	5
7	Falling while walking, using hands on a table to dampen the fall	Fainting	5	7	Slowly sit in a chair of medium height, wait for a moment, and stand up slowly.	5
8	Falling forward when attempting to get up	Attempting to get up	5	8	Quickly sit in a chair of medium height, wait for a moment, and stand up quickly.	5
9	Lateral falling when attempting to get up	Attempting to get up	5	9	Slowly sit in a low-height chair, wait for a moment, and stand up slowly.	5
10	Falling forward when attempting to sit down	Attempting to sit down	5	10	Quickly sit in a low-height chair, wait for a moment, and stand up quickly.	5
11	Falling backward when attempting to sit down	Attempting to sit down	5	11	Sitting for a moment, attempting to get up, and collapsing back into the chair.	5
12	Lateral fall when attempting to sit down	Attempting to sit down	5	12	Sitting for a moment, lying down slowly, waiting for a moment, and sitting up again.	5
13	Falling forward while sitting	Fainting or falling asleep	5	13	Sitting for a moment, lying down quickly, waiting for a moment, and sitting up again.	5
14	Falling backward while sitting	Fainting or falling asleep	5	14	Changing from lying on one's back to a lateral position, waiting for a moment, and changing back to lying on one's back.	5
15	Lateral falling while sitting	Fainting or falling asleep	5	15	Standing, bending at knees slowly, and getting up.	5
				16	Standing, bending without bending knees slowly, and getting up.	5
				17	Boarding a car, staying seated, and disembarking from the car.	5
				18	Stumbling while walking.	5
				19	Gently jumping without falling (attempting to reach a high object).	5

and classification. During the pre-processing stage, various adjustments and cleaning procedures were performed to obtain meaningful data from the dataset. In the feature extraction stage, significant features are extracted from the preprocessed dataset to detect fall events. Finally, in the classification stage, the extracted features were utilized to determine whether a fall event occurred. The performance metrics required for fall detection were analysed using five different machine learning techniques available in the PySpark and Scikit-Learn libraries.

2.1 Data formation

The Sisfall dataset is widely used for developing fall detection algorithms. This dataset is based on data collected from the ADXL345 accelerometer sensor (A_x, A_y, A_z), the ITG3200 gyroscope sensor (G_x, G_y, G_z), and the MMA8451Q accelerometer sensor (A_x, A_y, A_z). Here, the

abbreviations "*A*" and "*G*" refer to the accelerometer and gyroscope sensor data, respectively, and "x, y, and z" represent each vertical axis (Figure 1). Each file consisted of data records with 9 columns and 200 rows at a frequency of 200 samples per second. The data collection duration varied from 12 seconds (s) to 100 s for each movement. In this study, one accelerometer and one gyroscope were included in the research out of three sensors. The accelerometer to be used with the ITG3200 gyroscope was chosen considering the measurement range. The ADXL345 accelerometer is capable of measuring between -16 g and +16g, while the MMA8451Q accelerometer can measure between -8 g and +8g. Therefore, the ADXL345 accelerometer was preferred because a wider measurement range was required. This indicates that data records consisting of 6 columns and 200 rows were considered for this study.

To manage the large dataset, the data frames were divided into 0.5-second frames. Each 0.5-second frame consists of two 0.25-second frames (T_{HA}) surrounding the highest acceleration recorded by the waist sensor. This approach simplifies data management in the development of fall-detection algorithms.

$$T_{HA} = \sqrt{{A_x}^2 + {A_y}^2 + {A_z}^2} \quad (1)$$

The total acceleration (T_{HA}) vector is calculated from the average acceleration values along the x, y, and z axes as shown in Equation (1). Because each test lasts between 12 s to 100 s, the T_{HA} vector typically varies between 2400 to 20000 observations (12 s x 200 Hz - 100 s x 200 Hz). To remove meaningless data, the first and last 50 observations (0.25 seconds x 200 Hz) at the beginning and end of each test are disregarded.

In addition, any unusually high acceleration values are discarded. The remaining values include the highest acceleration value, which is used to create a 0.5-second vector consisting of two 0.25-second frames surrounding this value. Each frame contains 50 observations (0.25 seconds x 200 Hz), resulting in a total of 101 observations (50 observations + T_{HA} value + 50 observations). A single test repetition yields a data array consisting of 101 rows and 6 columns. Each row includes the measured acceleration or angular velocity values along the x, y, and z axes, and each column represents a vector. Therefore, with a single repetition of the test, a dataset of 101 * 6 is formed for one sensor unit.

2.2 Feature extraction

In this study, 26 different types of features were defined for motion recognition. Each sensor unit records signals along three axes for acceleration (A_x, A_y, A_z) and angular velocity (G_x, G_y, G_z). From these signals, various features are extracted at 0.5-second intervals. These features are divided into five main categories: basic statistical features (42 * 1), frequency domain features (114 * 1), time series features (94 * 1), motion features (27 * 1), and relational features (27 * 1). Using these features, a feature vector is generated for each movement (Table 2).

In this study, 3532 movements were recorded, including 15 falls and 19 activities of daily living. For each movement, 352 features were extracted (352 * 1). Using these features, a feature set of dimensions of 352 (352 * 3532) was created.

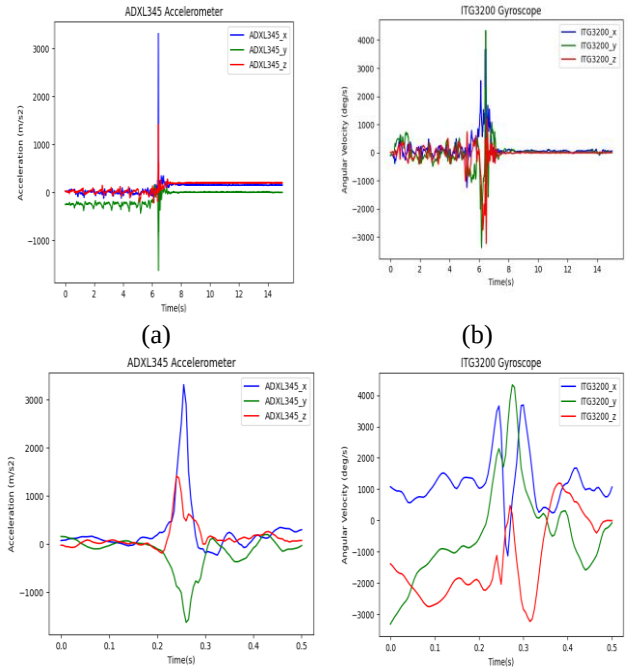

Figure 1. These figures showcase participant SA05 performing the "Fall Number 2: Falling backward while walking due to slipping" movement. Figures (a) and (b) display raw data recordings of 15 seconds (3000 samples) sampled at 200 Hz. Figures (c) and (d) depict the same data but compressed to 101 samples, representing 0.5 seconds of shortened data.

2.3 Machine learning algorithms

In this study, machine-learning algorithms were employed to detect falls and daily life activities. Features were extracted from the raw data and used as inputs for the classifiers. When an algorithm detects a fall or daily life activity, the developed model associates the input data with the corresponding labeled fall or daily life activity.

Five different machine learning algorithms were applied to detect falls, and their classification performance was compared. The following is a brief description of these algorithms.

2.3.1 Decision tree (DT)

DT is an algorithm used to make predictions by analyzing the data. The working principle of this algorithm involves creating a tree-like structure, in which each node represents a feature and each branch represents a decision. Starting from the root node, a decision tree asks questions regarding a feature and branches downwards, based on the responses. This process continues until a leaf node representing a prediction is reached. Decision trees can handle numerical and categorical data and have been used in fall detection systems to classify fall events [32].

Table 2. Features and Formation of Vector. A, G, Var., Std., Freq. are the accelerometer, gyroscope, variance, standard deviation, and frequency, respectively.

Type	Code	Features	Number of Features
Basic Statistical Features	F1	Minimum values [20]	xyz xyz A G **6 Features**
	F2	Maximum values [21]	xyz xyz A G **6 Features**
	F3	Mean values [20]	xyz xyz A G **6 Features**
	F4	Variance [22]	xyz xyz A G **6 Features**
	F5	Skewness [23]	xyz xyz A G **6 Features**
	F6	Kurtosis [23]	xyz xyz A G **6 Features**
	F7	Root Mean Square [24]	xyz xyz A G **6 Features**
Frequency Domain Features	F8	Discrete Fourier Transformation [25]	$xxxxx$ $yyyyy$ $zzzzz$ $A_x, A_y, \ldots\ldots, G_z$ **5, 5, ….. 5** **DFT(Peak) - DFT (Freq.)** **30 - 30** **60 Features**
	F9	Power Spectral Density [21]	$xxxxx$ $yyyyy$ $zzzzz$ $A_x, A_y, \ldots\ldots, G_z$ **51 Features**
	F10	Angular Velocity [26]	**Var. Std. Energy** xyz xyz xyz G **3 Features**
Time Series Features	F11	Autocorrelation [27]	$xxxxxxxxxx$ $y..$ $z..$ $A_x, A_y, \ldots\ldots, G_z$ **11, 11, …...11** **DFT(Peak) - DFT (Freq.)** **66 Features**
	F12	Spectral Entropy [28]	xyz xyz A G **6 Features**
	F13	Energy [24]	xyz xyz A G **6 Features**
	F14	Singular Value Decomposition [29]	xyz xyz A G **6 Features**
	F15	Signal Magnitude Area [30]	xyz A **10 Features**

Table 2.(Continue) Features and Formation of Vector. A, G, Var., Std., Freq. are the accelerometer, gyroscope, variance, standard deviation, and frequency, respectively.

Motion Features	**F16**	**Range of Variations [21]**	***xyz xyz*** ***A G*** **6 Features**
	F17	Coefficient of Variation [20]	***xyz xyz*** ***A G*** **6 Features**
	F18	Standard Error Mean [20]	***xyz xyz*** ***A G*** **6 Features**
	F19	Jerk [26]	***xyz*** ***A*** **3 Features**
	F20	Peak values [21]	***xyz*** ***A*** **3 Features**
	F21	Mean Crossing Rate [24]	***xyz*** ***A*** **3 Features**
Relational Features	F22	Correlation [20]	***xyz*** ***A*** **3 Features**
	F23	Covariance [20]	***xyz*** ***A*** **3 Features**
	F24	Autoregression [27]	***xyz*** ***A*** **33 Features**
	F25	Cross-Correlation [24]	***xyz*** ***A*** **33 Features**
	F26	Mutual Information [31]	***xyz*** ***A*** **3 Features**

2.3.2 Support vector machine (SVM)

SVM specializes in separating different categories by creating a hyperplane in a multidimensional feature space. The main objective of the SVM is to have a hyperplane that optimizes the margin between the data samples of each category. It is widely used in fall detection systems because of its ability to handle complex data structures and effectively generalize [33], [34].

2.3.3 Random forest classifier (RFC)

RFC is a technique that integrates multiple decision trees to produce predictions. In this approach, each tree is trained on a randomly selected subset of training data and pertinent features. Finally, an aggregated prediction is obtained through majority voting or averaging. Random forest is robust to overfitting and can effectively handle high-dimensional data. It has been used to improve the classification accuracy in fall detection systems [35].

2.3.4 Gradient boosting classifier (GBC)

GBC is an algorithm that combines several weak classifiers to create a strong classifier. The methodology of this algorithm involves building a progressive model with each subsequent classifier and correcting the mistakes of the previous classifiers. Gradient Boosting Classifier has achieved high accuracy in fall detection tasks [36].

2.3.5 Logistic regression (LR)

LR is a algorithm used to model the relationship between one or more independent variables and a dependent variable. LR belongs to the family of supervised learning algorithms. The implementation of logistic regression can vary depending on whether the dependent variable is binary or multiclass. Logistic regression has been used to differentiate between falls and daily life activities in fall detection systems [37], [38].

2.4 Classification

Fall detection systems are crucial, especially for the elderly and disabled. However, the performance of fall detection systems depends on the machine learning techniques employed and quality of the training data used.

In this study, the accuracy, sensitivity, and specificity criteria were used to evaluate the performance of the fall detection system. The accuracy represents the overall correctness rate of the system's decisions. The sensitivity represents the rate at which the system correctly detects all falls. Specificity represents the rate at which the system correctly identifies non-fall situations. To determine these criteria, four different scenarios need to be considered. In the first scenario, a real fall occurs, and the algorithm correctly detects it (True Positive - T_p). In the second scenario, no fall occurs, and the algorithm does not produce a fall alert (True Negative - T_n). T_n and T_p scenarios are considered as correct decisions by the algorithm. Incorrect decisions are labeled wrongly by the algorithm. In the third scenario, no fall actually occurs, but the algorithm incorrectly generates a fall alert (False Positive - F_p). Additionally, the algorithm may fail to detect a fall, which is known as a False Negative (F_n). This scenario is the most dangerous, as it can lead to severe injuries or even fatalities.

Sensitivity (Se), also known as Recall, is one of the most important criteria for fall detection systems, as a False Negative, which is the failure to detect an actual fall, can lead to severe injuries or death as in Equation (2).

$$Se = \frac{T_p}{T_p + F_n}\ x\ 100 \tag{2}$$

Specificity (Sp) is the measure of an algorithm's ability to correctly identify negative cases. It represents the ratio of correctly identified true negatives (T_n). A high Sp value indicates that the algorithm is successful in accurately identifying non-fall cases as in Equation (3). This is important in minimizing false alarms.

$$Sp = \frac{T_n}{T_n + F_p}\ x\ 100 \tag{3}$$

Accuracy (Acc) measures how well an algorithm predicts both sensitivity (Se) and specificity (Sp). Accuracy is calculated from T_p, T_n, F_p and F_n as in Equation (4).

$$Acc = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}\ x\ 100 \tag{4}$$

Therefore, a good binary classifier is expected to have high scores for all three criteria: sensitivity, specificity, and accuracy. However, in the case of fall detection, sensitivity is often the most important criteria.

This study aims to detect unexpected falls during daily activities. An ideal fall detection system should be able to distinguish falls resulting from rapid movements of body parts from routine activities. The algorithms need to be robust, intelligent, and sensitive in order to minimize false positive and false negative outcomes. While false positive alerts can be canceled by the user, it is critical that falls are not falsely classified as another activity. Missed falls can negatively impact a user's physical or mental well-being and hinder opportunities for intervention. Therefore, it is important for fall detection systems to have a high sensitivity and not miss any falls. Additionally, measures should be taken to prevent false alarm notifications from unnecessarily consuming the system resources. This study aimed to enhance the reliability and sensitivity of the proposed algorithm.

One of the most effective strategies for evaluating the effectiveness of machine learning models is to test them with unknown data in addition to performance criteria. In this study, a dataset collected from two groups consisting of 23 young adult participants was used. A test set of seven individuals was separated to evaluate the performance of the model. This approach ensures that no samples from the test set are used during the training of the model, thereby providing a more realistic and unbiased performance evaluation.

3 Results and discussion

In this section, the accuracy performance of machine learning algorithms trained on the fall detection dataset using PySpark and Scikit-Learn libraries are compared independently. Subsequently, the models with the highest training accuracy were evaluated for their generalization performance on the test set, including the accuracy, sensitivity, specificity, training time, and test time. After obtaining the results, the artificial intelligence library with the highest fall-detection performance was identified.

Table 3 shows the performance of five different machine learning algorithms for predicting fall and ADL classes using the PySpark library. When assessing the results of each algorithm, specific strengths and weaknesses were identified.

The LR algorithm stands out with the highest accuracy (98.40%). These results demonstrate the successful ability of the LR algorithm to distinguish between the fall and ADL classes. Additionally, both the sensitivity (97.89%) and specificity (98.89%) values were considerably high. However, the training time was longer than those of the other algorithms (12.3538 s), and the test time was slightly longer (0.1675 s).

The RFC algorithm also performs well, achieving a high accuracy rate of 98.31%. Both the sensitivity (98.08%) and specificity (98.53%) values are at a high level. Furthermore, the training time is similar to that of the SVM algorithm (6.6498 s), and the test time could be completed in a short period (0.1904 s).

The SVM algorithm also provided a high accuracy rate (98.31%) with balanced sensitivity (97.70%) and specificity (98.89%). The training time was slightly longer than that of the other algorithms (6.7290 s), but the test time was short (0.0942 s).

The DT algorithm achieved high accuracy (97.56%) with high sensitivity (98.27%) and specificity (96.88%). The

training time was 2.6269 s and the testing process was completed in 0.2263 s.

The GBC algorithm demonstrated similar performance to the other algorithms, achieving an accuracy rate of 97.75%. The sensitivity (97.70%) and specificity (97.79%) were balanced. However, the training time was slightly longer than those of the other algorithms (10.8316 s), and the test time was slightly higher (0.1849 s).

In conclusion, these findings indicate that different algorithms can be preferred depending on the specific datasets and usage scenarios. In cases that require faster results, algorithms such as SVM or LR, which perform faster, can be chosen. However, for the highest accuracy rate, algorithms such as LR, RFC, and SVM should be preferred. Therefore, the choice of the algorithm depends on many factors that need to be considered within a specific application context.

Furthermore, Table 4 evaluates the abilities of different machine learning algorithms to predict the "Falls" and "ADL" classes using the Scikit-Learn library.

According to the Accuracy results, LR (98.60%) stands out as the algorithm with the highest accuracy rate. This indicates that the algorithm generally classified the dataset correctly. On the other hand, DT (96.07%) appears to have the lowest accuracy rate, meaning it has a tendency to make more errors compared to the other algorithms.

The Sensitivity values measure the ability to correctly identify the "Falls" class. In this regard, SVM (98.28%) had the highest sensitivity values, indicating a better ability to identify the "Falls" class. DT (94.65%), on the other hand, has the lowest sensitivity value, indicating a tendency to misclassify the "Falls" class more frequently.

The Specificity metric measures the ability to correctly identify the "ADL" class. In this regard, LR (99.08%) has the highest specificity value, implying a better ability to identify the "ADL" class. On the other hand, SVM (97.25%) has the lowest specificity value, indicating a tendency to misclassify the "ADL" class more frequently.

Table 3. Analysis of fall detection performance using different machine learning techniques with the PySpark library on the test set. C.Fall: Classified Fall, C.ADL: Classified ADL, A.Fall: Actually Fall, A.ADL: Actually ADL

PySpark Library										
	DT		SVM		GBC		RFC		LR	
	C.Fall	C.ADL	C.Fall	C.ADL	C.Fall	C.ADL	C.Fall	C.ADL	C.Fall	C.ADL
A.Fall	514	9	511	12	511	12	513	10	512	11
A.ADL	17	528	6	539	12	533	8	537	6	539
Sp (%)	96.88		**98.89**		97.79		98.53		**98.89**	
Se (%)	**98.27**		97.70		97.70		98.08		97.89	
Acc (%)	97.56		98.31		97.75		98.31		**98.40**	
Training Time	**2.6269**		6.7290		10.8316		6.6498		12.3538	
Test Time	0.2263		**0.0942**		0.1849		0.1904		0.1675	

Table 4. Analysis of fall detection performance using different machine learning techniques with the Scikit-Learn library on the test set. C.Fall: Classified Fall, C.ADL: Classified ADL, A.Fall: Actually Fall, A.ADL: Actually ADL

Scikit-Learn Library										
	DT		SVM		GBC		RFC		LR	
	C.Fall	C.ADL	C.Fall	C.ADL	C.Fall	C.ADL	C.Fall	C.ADL	C.Fall	C.ADL
A.Fall	495	28	514	9	513	10	513	10	513	10
A.ADL	14	531	15	530	10	535	8	537	5	540
Sp (%)	97.43		97.25		98.17		98.53		**99.08**	
Se (%)	94.65		**98.28**		98.09		98.09		98.09	
Acc (%)	96.07		97.75		98.13		98.31		**98.60**	
Training Time	0.7036		0.5904		25.8212		**0.5156**		1.3104	
Test Time	**0.0239**		0.2283		0.0399		0.0609		0.0259	

Regarding training and test times, RFC (0.5904 seconds) stands out as the fastest-trained algorithm, indicating that the dataset can be trained quickly. Additionally, DT (0.0239 seconds) appears to have the fastest test time, producing results quickly. However, the training time of GBC (25.8212 seconds) is longer compared to other algorithms, so training times should be taken into consideration as well.

In conclusion, while LR exhibits the best overall performance, factors such as training time and test time requirements should be considered. The preference will depend on the specific requirements of the dataset and usage scenario.

The research aimed to compare PySpark and Scikit-Learn libraries in terms of fall detection. The main goal was to assess the performance difference between the two libraries and determine which library achieved better results.

To achieve this, five different machine learning algorithms were examined using both libraries. In the experiments with PySpark, the highest accuracy rate of 98.40% was obtained with the LR algorithm. The sensitivity (97.89%) and specificity (98.89%) values were also quite high. Similarly, in the experiments using the Scikit-Learn library, the highest accuracy rate of 98.60% was achieved with the LR algorithm, which also showed successful results in terms of sensitivity and specificity.

In terms of training time, it appeared that the PySpark library required slightly longer durations. On the other hand, the Scikit-Learn library provided shorter test times. This can vary depending on factors such as the size of the dataset and the processing power. For instance, when working with larger datasets, PySpark might provide faster results due to its parallel processing capabilities.

As a result, both libraries can effectively address the fall detection problem. However, the choice of library and algorithm should depend on factors such as the size of the dataset, processing power, and training/test times. This research can assist users in selecting the most suitable library and algorithm based on their specific requirements and datasets, thus achieving better accuracy performance. In addition, it is worth noting that the comparison between PySpark and Scikit-Learn in the context of fall detection models is a relatively unexplored area in the existing literature. While there have been studies that have examined the performance of these libraries in various machine learning tasks [3-6], their specific application in fall detection has not been extensively investigated. This highlights the novelty and significance of this study in contributing to the understanding of the performance of PySpark and Scikit-Learn in the development of fall detection models.

In addition, when the performance of the study in terms of fall detection was examined, the proposed approach using logistic regression with new features extracted from the SisFall dataset achieved a promising accuracy rate of 98.6% in fall detection. This performance surpasses previous studies like Shi et al. [39] (improved pre-impact fall detection with 95.33% accuracy) and Zulj et al. [40] (various algorithms with accuracies ranging from 80.1% to 98.6%), and competes favorably with more complex approaches like Lee et al. [41] (fall detection using both plantar pressure and acceleration data with 95% accuracy). These results suggest that the newly extracted features effectively enhance the discriminatory power of fall detection systems, aligning with Gjoreski et al.'s [42] findings on the importance of data fusion and machine learning for accurate activity recognition and fall detection. Importantly, the achieved accuracy paves the way for potential real-world implementation, echoing Ojetola et al.'s [43] emphasis on the significance of wearable sensors in fall prevention. Overall, this study not only contributes to the development of more accurate and effective fall detection systems but also holds promising implications for improving safety measures in various settings.

4 Conclusion

This study comprehensively evaluated the performance of two prominent machine learning libraries, PySpark and Scikit-Learn, in fall detection modeling. The findings demonstrate the robust and efficient capabilities of both libraries in accurately classifying fall events from a diverse set of activities. Logistic regression and random forest algorithms consistently outperformed other models, achieving the highest accuracy rates across both libraries.

PySpark's distributed computing capabilities proved advantageous for handling large datasets, enabling efficient training and processing. However, Scikit-Learn exhibited superior performance in test times, making it more suitable for smaller datasets or real-time applications.

This study distinguishes itself from previous work by employing a comprehensive feature extraction approach that encompasses 26 features across five categories: basic statistical features, frequency domain features, time series features, motion features, and relational features. This novel approach captured a wider range of information from the Sisfall dataset, contributing to the enhanced accuracy observed in this study.

The choice between PySpark and Scikit-Learn for fall detection modeling depends on factors such as dataset size, processing power, and time constraints. PySpark is well-suited for large datasets due to its distributed computing capabilities, while Scikit-Learn offers a user-friendly interface and faster test times for smaller datasets.

Overall, this study provides valuable insights into the strengths and limitations of PySpark and Scikit-Learn for fall detection modeling. It also highlights the effectiveness of the proposed feature extraction approach in improving fall detection accuracy. These findings can guide researchers and practitioners in selecting the most suitable library and algorithm for their specific fall detection needs.

In addition, to always make progress in the research field, it is necessary to ask new questions and generate ideas for future studies.

In this direction, the following possibilities can lead to significant advances in the field of fall detection:

- Integration of different sensors: Combining the data from different sensors (accelerometers, GPS,

cameras, audio, etc.) for fall detection can create a more comprehensive model.

- Investigation of deep learning methods: Deep learning techniques have been shown to be very successful in processing time series data in recent years. Adapting deep learning models for fall detection and comparing their performances will contribute to the advancement of the field.
- Focus on personalized modeling: Developing personalized fall detection models can better adapt to real-world scenarios.

These suggestions reveal exciting research opportunities for the future in the field of fall detection. Each approach should be studied in depth and the effectiveness of fall detection systems in real life should be increased through applied research.

Conflict of interest

There is no conflict of interest with any person/institution in the prepared article.

Data and materials availability

Data and Materials Availability: The dataset utilized in this study has been made available as open source and can be accessed via the following link. (https://github.com/JiayangLai/SisFallDatasetAnnotation)

Similarity rate (iThenticate): 16%

References

[1] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, SisFall: A fall and movement dataset, Sensors, vol. 17, no. 1, Art. no. 1, Jan. 2017, doi: 10.3390/s17010198.

[2] M. Islam et al., Deep learning based systems developed for fall detection: A Review, IEEE Access, vol. 8, pp. 166117–166137, 2020, doi: 10.1109/ACCESS.2020.3021943.

[3] T. C. Nokeri, Principal component analysis with Scikit-Learn, PySpark, and h2o, in data science solutions with python: fast and scalable models using keras, PySpark, mllib, h2o, xgboost, and Scikit-Learn, T. C. Nokeri, Ed., Berkeley, CA: Apress, 2022, pp. 101–110. doi: 10.1007/978-1-4842-7762-1_9.

[4] T. C. Nokeri, Cluster analysis with Scikit-Learn, PySpark, and h2o, in data science solutions with python: fast and scalable models using keras, PySpark mllib, h2o, xgboost, and Scikit-Learn, T. C. Nokeri, Ed., Berkeley, CA: Apress, 2022, pp. 89–99. doi: 10.1007/978-1-4842-7762-1_8.

[5] M. Junaid et al., Performance evaluation of data-driven intelligent algorithms for big data ecosystem, Wirel. Pers. Commun., vol. 126, no. 3, pp. 2403–2423, Oct. 2022, doi: 10.1007/s11277-021-09362-7.

[6] T. C. Nokeri, Tree modeling and gradient boosting with Scikit-Learn, xgboost, PySpark, and h2o, in data science solutions with python: fast and scalable models using keras, PySpark mllib, h2o, xgboost, and Scikit-Learn, T. C. Nokeri, Ed., Berkeley, CA: Apress, 2022, pp. 59–74. doi: 10.1007/978-1-4842-7762-1_6.

[7] T. S. Patel, D. P. Patel, and C. N. Patel, Real time scalable data acquisition of covid-19 in six continents through PySpark - a big data tool. medRxiv, p. 2021.07.04.21259983, Jul. 06, 2021. doi: 10.1101/2021.07.04.21259983.

[8] A. Gupta, H. K. Thakur, R. Shrivastava, P. Kumar, and S. Nag, A big data analysis framework using apache spark and deep learning, in 2017 IEEE International Conference on Data Mining Workshops (ICDMW), Nov. 2017, pp. 9–16. doi: 10.1109/ICDMW.2017.9.

[9] K. Rothauge, H. Ayyalasomayajula, K. J. Maschhoff, M. Ringenburg, and M. W. Mahoney, Running alchemist on cray xc and cs series supercomputers: dask and PySpark interfaces, deployment options, and data transfer times. arXiv, Nov. 28, 2019. doi: 10.48550/arXiv.1910.01354.

[10] S. Gafner et al., Evaluation of hip abductor and adductor strength in the elderly: a reliability study, Eur. Rev. Aging Phys. Act., vol. 14, no. 1, p. 5, Apr. 2017, doi: 10.1186/s11556-017-0174-6.

[11] A. Abraham et al., Machine learning for neuroimaging with Scikit-Learn, Front. Neuroinformatics, vol. 8, 2014, doi: 10.3389/fninf.2014.00014.

[12] L. Buitinck et al., API design for machine learning software: experiences from the Scikit-Learn project. arXiv, Sep. 01, 2013. doi: 10.48550/arXiv.1309.0238.

[13] A. Auti, D. Patil, O. Zagade, P. Bhosale, and P. Ahire, Bitcoin price prediction using svm, vol. 6, no. 11, 2022.

[14] J. Hao and T. K. Ho, Machine learning made easy: a review of Scikit-Learn package in python programming language, J. Educ. Behav. Stat., vol. 44, no. 3, pp. 348–361, Jun. 2019, doi: 10.3102/1076998619832248.

[15] M. W. Liemohn et al., Model evaluation guidelines for geomagnetic index predictions, Space Weather, vol. 16, no. 12, pp. 2079–2102, 2018, doi: 10.1029/2018SW002067.

[16] E. Uzunhisarcıklı, E. Kavuncuoğlu, and A. T. Özdemir, Investigating classification performance of hybrid deep learning and machine learning architectures on activity recognition, Comput. Intell., vol. 38, no. 4, Art. no. 4, 2022, doi: 10.1111/coin.12517.

[17] E. Kavuncuoğlu, E. Uzunhisarcıklı, B. Barshan, and A. T. Özdemir, Investigating the performance of wearable motion sensors on recognizing falls and daily activities via machine learning, Digit. Signal Process., p. 103365, Dec. 2021, doi: 10.1016/J.DSP.2021.103365.

[18] M. Ş. Turan and B. Barshan, Classification of fall directions via wearable motion sensors, Digit. Signal Process., p. 103129, Jun. 2021, doi: 10.1016/j.dsp.2021.103129.

[19] A. T. Özdemir, An analysis on sensor locations of the human body for wearable fall detection devices: Principles and practice, Sens. Switz., vol. 16, no. 8, Art. no. 8, Jul. 2016, doi: 10.3390/s16081161.

[20] D. C. Montgomery, E. A. Peck, and G. G. Vining, Introduction to linear regression analysis. John Wiley & Sons, 2013.

[21] J. S. Bendat and A. G. Piersol, Random data: analysis and measurement procedures. John Wiley & Sons, 2011.
[22] J. L. Devore, Probability and statistics for engineering and the sciences. Cengage Learning, 2015.
[23] N. L. Johnson, S. Kotz, and N. Balakrishnan, Continuous univariate distributions, *Vol. 1*, 2nd edition. New York: Wiley-Interscience, 1994.
[24] A. V. Oppenheim and R. W. Schafer, Discrete-time signal processing. Pearson, 2010.
[25] J. G. Proakis and D. G. Manolakis, Digital signal processing: principles, algorithms, and applications. Macmillan, 1992.
[26] Signal processing, in biomechanics and motor control of human movement, John Wiley & Sons, Ltd, 2009, pp. 14–44. doi: 10.1002/9780470549148.ch2.
[27] P. J. Brockwell and R. A. Davis, Introduction to time series and forecasting. in Springer Texts in Statistics. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-29854-2.
[28] T. Inouye et al., Quantification of eeg irregularity by use of the entropy of the power spectrum, Electroencephalogr. Clin. Neurophysiol., vol. 79, no. 3, pp. 204–210, Sep. 1991, doi: 10.1016/0013-4694(91)90138-T.
[29] G. H. Golub and C. F. V. Loan, Matrix computations. JHU Press, 2013.
[30] A. Mannini and A. M. Sabatini, Machine learning methods for classifying human physical activity from on-body accelerometers, Sensors, vol. 10, no. 2, Art. no. 2, Feb. 2010, doi: 10.3390/s100201154.
[31] T. M. Cover and J. A. Thomas, Elements of information theory. John Wiley & Sons, 2012.
[32] T. Hastie, R. Tibshirani, and J. Friedman, Overview of supervised learning, in the elements of statistical learning: data mining, inference, and prediction, T. Hastie, R. Tibshirani, and J. Friedman, Eds., in Springer Series in Statistics. , New York, NY: Springer, 2009, pp. 9–41. doi: 10.1007/978-0-387-84858-7_2.
[33] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov., vol. 2, no. 2, pp. 121–167, Jun. 1998, doi: 10.1023/A:1009715923555.
[34] H. T. Babacan, Ö. Yüksek, and F. Saka, Yapay zeka ve sezgisel regresyon yöntemlerinin yağış-akış modellemesi için performans değerlendirmesi: Aksu Deresi için bir uygulama, Niğde Ömer Halisdemir Üniversitesi Mühendis. Bilim. Derg., vol. 11, no. 3, Art. no. 3, Jul. 2022, doi: 10.28948/ngumuh.1079616.
[35] L. Breiman, Random forests, Mach. Learn., vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.
[36] J. H. Friedman, Greedy function approximation: a gradient boosting machine, 2001.
[37] Introduction to the logistic regression model, in Applied Logistic Regression, John Wiley & Sons, Ltd, 2013, pp. 1–33. doi: 10.1002/9781118548387.ch1.
[38] M. Demirhan and S. Behdioğlu, Sağlık çalışanlarının maruz kaldığı şiddetin sıralı lojistik regresyon analizi ile incelenmesi, Toplum Ekon. Ve Önetim Derg., vol. 4, no. 1, Art. no. 1, Jun. 2023, doi: 10.58702/teyd.1228283.
[39] J. Shi, D. Chen, and M. Wang, Pre-impact fall detection with cnn-based class activation mapping method, Sensors, vol. 20, no. 17, Art. no. 17, Jan. 2020, doi: 10.3390/s20174750.
[40] S. Zulj, G. Seketa, I. Lackovic, and R. Magjarevic, Accuracy comparison of ml-based fall detection algorithms using two different acceleration derived feature vectors, in World Congress on Medical Physics and Biomedical Engineering 2018, L. Lhotska, L. Sukupova, I. Lacković, and G. S. Ibbott, Eds., in IFMBE Proceedings. Singapore: Springer, 2019, pp. 481–485. doi: 10.1007/978-981-10-9038-7_89.
[41] C. M. Lee, J. Park, S. Park, and C. H. Kim, Fall-detection algorithm using plantar pressure and acceleration data, Int. J. Precis. Eng. Manuf., vol. 21, no. 4, pp. 725–737, Apr. 2020, doi: 10.1007/s12541-019-00268-w.
[42] H. Gjoreski et al., Wearable sensors data-fusion and machine-learning method for fall detection and activity recognition, Stud. Syst. Decis. Control, vol. 273, pp. 81–96, 2020, doi: 10.1007/978-3-030-38748-8_4.
[43] O. Ojetola, E. I. Gaura, and J. Brusey, Fall detection with wearable sensors–safe (smart fall detection), in 2011 Seventh International Conference on Intelligent Environments, Jul. 2011, pp. 318–321. doi: 10.1109/IE.2011.38.