IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 9, No. 4, December 2020, pp. 772~780
ISSN: 2252-8938, DOI: 10.11591/ijai.v9.i4.pp772-780

Machine learning algorithms for fall detection using kinematic and heart rate parameters-a comprehensive analysis

Anita Ramachandran[1], Adarsh Ramesh[2], Aditya Sukhlecha[3], Avtansh Pandey[4], Anupama Karuppiah[5]
[1]Department of Computer Science & Information Systems, BITS Pilani, Bangalore, India
[2]Current affiliation: Quantiphi Analytics, Bangalore, India
[3,4,5]Department of Electrical & Electronics Engineering, BITS Pilani-KK Birla Goa Campus, Goa, India

Article Info

Article history:

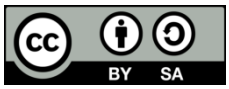
Received Jul 20, 2020
Revised Oct 30, 2020
Accepted Nov 26, 2020

Keywords:

Ensemble techniques
Fall detection
Machine learning
Wearable devices

ABSTRACT

The application of machine learning techniques to detect and classify falls is a prominent area of research in the domain of intelligent assisted living systems. Machine learning (ML) based solutions for fall detection systems built on wearable devices use various sources of information such inertial motion units (IMU), vital signs, acoustic or channel state information parameters. Most existing research rely on only one of these sources; however, a need to do more experimenation to observe the efficiency of the ML classifiers while coupling features from diverse sources, was felt. In addition, fall detection systems based on wearable devices, require intelligent feature engineering and selection for dimensionality reduction, so as to reduce the computational complexity of the devices. In this paper we do a comprehensive performance analysis of ML classifiers for fall detection, on a dataset we collected. The analysis includes the impact of the following aspects on the performance of ML classifiers for fall detection: (i) using a combination of features from 2 sensors-an IMU sensor and a heart rate sensor, (ii) feature engineering and feature selection based on statistical methods, and (iii) using ensemble techniques for fall detection. We find that the inclusion of heart rate along with IMU sensor parameters improves the accuracy of fall detection. The conclusions from our experimentations on feature selection and ensemble analysis can serve as inputs for researchers designing wearable device-based fall detection systems.



Corresponding Author:

Anita Ramachandran
Department of Computer Science & Information Systems
BITS Pilani, 3rd floor, Vayudoot Chambers
Trinity Circle, Bangalore, India-560001
Email: anita.ramachandran@pilani.bits-pilani.ac.in

1. INTRODUCTION

The application of computational intelligence in the field of fall detection has gained increased focus in the recent years because of its impact on geriatric healthcare. Internet of Things and machine learning enable design of fall detection systems (FDS) that are capable of monitoring human movements remotely. Such FDSs analyse body movements and postures, using cameras, ambient sensors or wearable sensors. Sensor based systems monitor and report kinematic parameters such as those from an inertial motion unit (IMU) (accelerometer, gyroscope and magnetometer) or biomedical observations heart rate variability (HRV), galvanic skin response (GSR) and oxygen saturation (SPO2)). Existing research shows application of various machine learning algorithms to detect falls using a subset of the above mentioned features.

Machine learning (ML) techniques for fall detection based on wearable devices differ from each other because the data sources and the characteristics of data generated in each case is different. For e.g., in [1], the features used to detect falls include acceleration, rate of turn, and the strength of the Earth's magnetic field along the X, Y and Z axes, and the classifiers applied are k-nearest neighbor (kNN) classifier, least squares method (LSM), support vector machines (SVM), bayesian decision making (BDM), dynamic time warping (DTW), and artificial neural networks (ANNs). The accuracy of fall detection in wearable sensors based techniques, is also influenced by the placement of the sensors. Yu *et al.* [2] attempts to reduce errors caused by incorrect sensor positions using hidden markov model (HMM). Guvensan *et al.* [3] explores how energy efficiency of wearable devices can be optimized. In addition to wearable devices, fall detection is also enabled by machine learning algorithms applied on data generated by sensors integrated with mobile phones [4]. Other related research in this area include [5-7].

Improving performance metrics of ML classifiers for fall detection involves techniques such as data pre-processing, feature engineering and selection, and creating ensembles which combine multiple weak classifiers to generate a strong classifier. Vallabh *et al.* [8] applies feature extraction and rank based feature selection to data generated by accelerometer and gyroscope, and evaluates naïve bayes, LSM, ANN, SVM and kNN classifiers for fall detection. In [9], feature engineering is performed after grouping the real time sensor data into windows. Subsequently, classification algorithms were applied to each window for fall detection. Another example is [10], in which data generated by a tri-axial gyroscope, is divided into consecutive and partially overlapping windows, post which three time-domain features were extracted. Decision tree was applied on the extracted features for fall classification. Other research that perform windowing and feature extraction includes [11], with data from a wearable motion sensor and a smartphone, and [12] using acceleration and angular velocity. Optimal feature selection plays an important role in the accuracy of fall detection. Researches in [13-15] explore this aspect-Wang *et al.* [13] applies bayesian framework to calculate the weight of each feature, based on which an optimal feature set was identified as input to training the ML classifier, while Tsinganos and Skodras [14] analyzes accelerometer data to extract a set of 14 features across timedomain, statistical measures and continuous wavelet transform, and Kao *et al.* [15] applies genetic algorithms for feature selection and SVM for classification. Jahanjoo *et al.* [16] proposes a fall detection algorithm based on data from 3-axis accelerometers, performs dimensionality reduction and applies a multilevel fuzzy (MLF) min-max neural network. The accuracy of the MLF algorithm was reported to be 97.29%. The use of ensembles in ML techniques for fall detection has also been widely explored [17-18]. In fact, [19] does a performance analysis of various ML classifiers on a public dataset, and reports that ensemble classifiers such as random forest and gradient boosting produce the best results. Deep learning has also been employed for fall detection, and results show high accuracy in fall classification [20-21]. However, a constrained wearble device may be unable to host deep learning-based solutions which typically run on processors with high computational capabilities. [22] provides a survey of recent advances in wearable fall detection systems.

We observe that most of the existing research on wearable fall detection systems, apply ML classifiers on single sources of information [23-25]. Therefore, there is a need to explore the impact of compounding feature vectors from multiple sources for fall detection, on the performance of ML classifiers. In wearable device based FDSs, it is also important to ascertain a minimal set of features that can efficiently detect falls, because reducing the number of features used for classification leads to lower computational complexities, and hence, decreased power consumption of the devices, while also not compromising the classifier performance.

In this paper, we explore the impact of using a feature set created from two sources of data - an IMU sensor and a heart rate sensor - on the performance of various ML classifiers. We also perform feature engineering and feature selection on the dataset we collected, in order to find out the most significant features that can enable efficient fall detection. Our study spans the performance of several base classifiers and ensemble classifiers. This work is part of an intelligent wearable device based FDS we are developing, for the deployment scenario of a geriatric care home. The observations from our analysis will help us make the right design trade-offs, in the distributed architecture of the fall detection system under development.

The rest of the paper is organized as follows: Section 2 describes our methodology, including data collection, pre-processing, dataset partitioning, feature engineering and selection, and application of ensemble classifiers. Section 3 describes the results obtained at various steps in our experimentation. Section 4 outlines our remarks on the practical implications of the results of our experimental analyses.

2. RESEARCH METHOD

The objective of our work was to analyse the impact of the following factors on the performance of ML classifiers:

– Using a feature vector comprising both vital signs and IMU parameters

- Feature engineering techniques and feature selection based on statistical methods to find out a minimal feature vector that can classify falls efficiently
- Use of ensemble techniques for fall detection to observe how a combination classifier performs with respect to base classifiers

We collected a dataset using a TicWatch S device, by performing 20 different activity simulations, which had 14 ADLs and 6 falls. The dataset consisted of IMU observations and heart rate values, generated by the corresponding sensors in the device. First, we analysed the dataset by considering only the IMU features (from the 3-axes accelerometer, 3-axis magnetometer, 3-axis gyroscope and linear acceleration sensors), using kNN, naïve byes, ANN, extreme gradient boosting (XGBoost) and random forest classifiers. Subsequently, we included both IMU and optical heart rate sensor (HR) observations from the dataset for analysis, and applied the same set of ML classifiers to observe their performance. We then applied feature engineering and selection techniques based on statistical methods on the datasets. The last step in the analysis included application of ensemble techniques on the datasets, to study their performance. The rest of the paper describes the details of work done in each of these steps and results thereof. We refer to two of our previous work, where necessary.

2.1. Data collection, pre-processing and baseline performance profiling

Currently available public datasets for fall detection have either IMU features or vital signs features, but not any that includes both. Therefore, we collected our dataset, using the TicWatch S smartwatch. This device includes a tri-axial accelerometer, magnetometer, gyroscope, linear acceleration and optical heart rate sensor. Experiments were performed across 20 different ADL/fall activity simulations, such as walking, running, climbing stairs, abrupt movements and various types of falls, in controlled environments. One of our previous papers describes the details on the fall simulation and dataset collection process [26]. Basic pre-processing was then performed on the data set. This included removal of data values that fell outside the valid experimentation windows and data imputation using last observation carried forward (LOCF).

The dataset was then partitioned into 2 sets, one containing IMU features only and the second containing both IMU and HR features, to compare the performances of ML algorithms when using only IMU features vis-à-vis using IMU and HR features. Each of these was then split into training and testing datasets (80% train and 20% test) randomly 5 times, such that every iteration produced a different combination of testing-training records. We applied kNN, naïve bayes, ANN, XGBoost and random forest to the datasets in such a manner that each ML classifier was applied on different testing and training datasets. We report observations from the test sets.

We did an initial analysis of how various ML classifiers performs on these two datasets, and the accuracies are summarized in Table 1. In both cases, random forest gave the highest accuracy. It was also observed that with heart rate values in combination with IMU values, the accuracy of the classifiers improved. Python scikit-learn libraries were used for software implementation. These results were also used as baseline figures against which the efficiencies of feature engineering and scaling were compared. In the next step, we applied feature engineering using statistical methods to extract features from the attributes of the raw dataset.

Table 1. Summary of accuracy (Baseline)-IMU-Only vs IMU+HR

Dataset	kNN	XGBoost	Naïve-Bayes	ANN	Random Forest
IMU Only	77	69	39	66	77
IMU+HR	89	72	40	69	93

2.2. Feature engineering

The feature engineering steps we performed included feature scaling, feature extraction and selection. Feature scaling was applied on the dataset using the standard scaler library from Python scikit-learn to transform the data such that its distribution has a mean of 0 and standard deviation of 1. Since the collected dataset was time series data, we applied a rolling window technique to perform statistical analysis on the dataset. This technique splits the data into windows of the specified length, with the number of increments between successive rolling windows set to 1 period. The dataset was partitioned into N subsamples, using (1):

$$N=T-m+1 \quad (1)$$

where T is the sample size and m is the rolling window size

All points within a window were evenly weighted. A window size of 20, which maps to a window interval of 5 seconds, gave us marginally better results, because such a window has enough samples to accurately classify an activity. Subsequently, feature extraction was performed for every window, for each of the set of values reported by the magnetometer, gyroscope, accelerometer and heart rate sensors, within that window. Thus, the feature vector was extended to include not only the instantaneous time series parameters, but also, the mean, median, standard deviation, variance, skew and kurtosis, of the attribute values within a window.

2.3. Feature selection

During further analysis, it was found, as expected, that some features presented more importance to the classifier accuracy than certain other features. Hence the feature importance for all the features was compared to enable feature selection so as to observe (i) the classifier performance based on the more important features and (ii) the impact of inclusion of heart rate observations along with IMU sensor observations, on the classifier accuracy. We applied ranking based feature selection techniques supported by Python scikit-learn libraries, for this purpose. For kNN, features were selected according to k highest scores, where k=10 and number of nearest neighbors as 7. In the naïve bayes classifier, the top 10 features with the highest scores, were selected. Random forest classifier was applied with no: of trees set to 10, and entropy as the function to measure quality of the split. Feature selection was done based on importance weights, as set by a threshold, such that those features whose importance is greater than or equal to the threshold are retained and the others are discarded. For analysis using ANN, multi-layer perceptron classifier was used, with weight optimization using quasi-Newton method, and hidden layer sizes as (5, 2). The activation function to the hidden layer was the rectified linear unit function. XGBoost classifier was tuned for tree based boosting with 100 trees to fit and a maximum depth of 3. We experimented with 2 threshold values-0.01 and 0.02, in combination with window sizes set to 10 and 20. Threshold values of 0.01 and 0.02 gave us comparable results, hence we chose threshold = 0.01 for further analysis.

2.4. Application of ensemble classifiers

Ensemble techniques for machine learning are typically used when there is data from disparate sources. They are also used if certain classifiers are found to be performing poorly, but is nevertheless preferred to solve the problem at hand, for various reasons. Ensemble classifiers combine different base classifiers using a meta classifier to produce a single output, which performs better than the base classifiers. Ensemble classifiers take advantage of the concept of wisdom of crowds - they overcome the drawbacks of the constituent base classifiers and creates a combination classifier that performs better than the base classifiers. The effectiveness of the ensemble classifier depends on: (i) the base classifiers being used, (ii) the input to each base classifier, and (iii) the method to combine the outputs of the base classifier. Some examples of ensemble classifiers include random forest, extra tree classifier, Adaboost and its variants. In order to evaluate the results of applying ensemble classifiers further to our problem, we experimented with the following scenarios, on the IMU+HR dataset, post feature engineering and feature selection.

2.4.1. Stacking classifiers

A stacking classifier implements a stack of estimators with a final classifier. We used a stacking classifier with the following combinations of base classifiers: decision tree, ANN, naïve bayes, kNN and logistic regression, taking combinations of 2 and 3 of these classifiers to form different configurations of stacked ensemble classifiers. The input into the base classifiers was the complete IMU+HR dataset.

2.4.2. Voting classifiers

A voting classifier fits multiple base classifiers and predicts an output class based on the highest majority of voting by the base classifiers. The voting can be hard or soft-hard, if the output is the one which had the highest probability of being predicted by the base classifiers, and soft, if the output class is predicted based on votes obtained according to sum of weighted probabilities of each individual classifiers. We used a combination of decision tree, ANN, naïve bayes, kNN and logistic regression as base classifiers, with voting type as hard and soft, to form different configurations of voting ensemble classifiers.

2.4.3. Bagging classifiers

Bagging classifiers fit base classifiers on random subsets of the original dataset, and combine the individual predictions either by voting or averaging, to form a final prediction. This is used to reduce the variance in the base classifiers. In our experiments, we created bagging classifiers with kNN, decision tree, logistic regression, naïve bayes and ANN as the base classifiers, with samples from the dataset drawn with replacement.

3. RESULTS AND DISCUSSION

We summarize our results and observations on the performance of ML classifiers on the IMU-Only dataset and the IMU+HR dataset, with respect to the impact of: (i) feature engineering and selection, (ii) inclusion of HR attribute along with IMU attributes, and (iii) impact of using ensembles for fall classification.

3.1. Impact of feature engineering and selection

Results from feature engineering and feature selection, on the IMU-Only and IMU+HR datasets, are described below. Subsequently, at the end of this section, we summarize the accuracy of the ML classifiers on the two datasets, across the various scenarios we experimented with.

3.1.1. IMU-only dataset

In the IMU-Only dataset, the most significant features for the various classifiers selected, based on feature ranking, is summarized in Table 2. It was observed that the mean and median of attribute values within a window exhibited higher significance than other derived features, in most cases. Hence, in order to optimize the number of features for classification, we explored limiting the feature vector to contain only the mean of all IMU attributes. Table 3 lists the features extracted in this case.

Subsequently, we applied selected both the mean and median of IMU attributes for inclusion in the feature vector. Random forest gave us the best accuracy (99.7%) in the case where the feature vector comprised of the mean and median of the IMU attributes. The detailed performance metrics of the 5 ML classifiers applied, in these cases, is described in [27].

Table 2. Impact of feature engineering & selection-IMU-only dataset [Highest scores]

Classifier	Selected features from IMU-Only dataset [Highest scores]
kNN	Mean of X, Y and Z-axis acceleration, mean of X-axis gyroscope value, standard deviation of X-axis acceleration, standard deviation of Z-axis magnetometer value, standard deviation of X, Y and Z-axis liner acceleration, skew of X-axis magnetometer value
Naïve Bayes	Mean of X, Y and Z-axis acceleration, mean of X-axis gyroscope value, standard deviation of X-axis acceleration, standard deviation of Z-axis magnetometer value, standard deviation of X, Y and Z-axis liner acceleration, skew of X-axis magnetometer value
Random Forest	Mean of X, Y and Z-axis acceleration, mean of Y and Z-axis magnetometer value, mean of X, Y and Z-axis linear acceleration, median of X, Y and Z-axis acceleration, median of X, Y and Z-axis magnetometer values, median of X, Y and Z-axis gyroscope values, median of X and Z-axis linear acceleration, standard deviation of X, Y and Z-axis acceleration, standard deviation of X and Y-axis magnetometer value, standard deviation of Z-axis gyroscope values, variance of X, Y and Z-axis acceleration, variance of X and Z-axis linear acceleration
ANN	All features
XGBoost	Mean of X, Y and Z-axis acceleration, mean of X, Y and Z-axis magnetometer values, Mean of X and Y-axis linear acceleration, median of X, Y and Z-axis acceleration, median of X, Y and Z-axis magnetometer values, median of Z-axis linear acceleration, standard deviation of X, Y and Z-axis acceleration, standard deviation of X and Y-axis magnetometer value, standard deviation of X and Z-axis gyroscope values, standard deviation of X, Y and Z-axis linear acceleration, skew of Y-axis gyroscope values, skew of Y-axis linear acceleration, kurtosis of X and Y-axis magnetometer values

Table 3. Impact of feature engineering & selection-IMU-only dataset [Mean of attribute values]

Classifier	Selected features from IMU-Only dataset [Mean of attribute values]
kNN	Mean of X, Y and Z-axis acceleration, mean of X and Z-axis magnetometer values, mean of X and Y-axis gyroscope values, mean of X, Y and Z-axis linear acceleration
Naïve Bayes	Mean of X, Y and Z-axis acceleration, mean of X and Z-axis magnetometer values, mean of X and Y-axis gyroscope values, mean of X, Y and Z-axis linear acceleration
Random Forest	Mean of X, Y and Z-axis acceleration, mean of X, Y and Z-axis magnetometer values, mean of X, Y and Z-axis gyroscope values, mean of X, Y and Z-axis linear acceleration
ANN	All
XGBoost	Mean of X, Y and Z-axis acceleration, mean of X, Y and Z-axis magnetometer values, mean of Z-axis gyroscope values, mean of X, Y and Z-axis linear acceleration

3.1.2. IMU+HR dataset

In the IMU+HR dataset, the most important features for the various classifiers applied, based on feature ranking, is summarized in Table 4. Table 5 lists the features selected in the case where the mean of all attributes was considered. As in the case of IMU-Only dataset, a similar set of features were selected with mean and median of IMU and HR attributes for analysis row (8) of Table 6. Random forest gave us the best accuracy (99.8%) in the case where the feature vector comprised of the mean and median of the IMU attributes. The detailed performance metrics of the 5 ML classifiers applied, in these cases, is summarized in [23].

Table 4. Impact of feature engineering & selection-IMU+HR dataset [Highest scores]

Classifier	Selected features from IMU+HR dataset [Highest scores]
kNN	Mean of X and Y-axis acceleration, mean of Z-axis magnetometer values, mean of Z-axis gyroscope values, standard deviation of X-axis acceleration, standard deviation of Z-axis magnetometer values, standard deviation of X, Y and Z-axis linear acceleration, skew of X-axis magnetometer values
Naïve Bayes	Mean of X and Y-axis acceleration, mean of Z-axis magnetometer values, mean of X-axis gyroscope values, standard deviation of X-axis acceleration, standard deviation of Z-axis magnetometer values, standard deviation of X, Y and Z-axis linear acceleration, skew of X-axis magnetometer values
Random Forest	Mean of X, Y and Z-axis acceleration, mean of heart rate, mean of Z-axis magnetometer value, mean of X, Y and Z-axis gyroscope value, mean of Z-axis linear acceleration, median of X, Y and Z-axis acceleration, median of Y and Z-axis magnetometer values, median of Y and Z-axis linear acceleration, standard deviation of X and Y-axis magnetometer value, standard deviation of X and Y-axis linear acceleration, skew of heart rate, skew of X and Y-axis magnetometer values, skew of Z-axis gyroscope values, skew of X-axis linear acceleration, kurtosis of Z-axis acceleration and X-axis gyroscope values
ANN	All features
XGBoost	Mean of X, Y and Z-axis acceleration, mean of Z-axis magnetometer value, mean of X and Y-axis gyroscope value, mean of Z-axis linear acceleration, median of X and Y-axis acceleration, median of Y and Z-axis magnetometer values, median of Y and Z-axis linear acceleration, standard deviation of X-axis acceleration, standard deviation of heart rate, standard deviation of X, Y and Z-axis magnetometer values, standard deviation of Z-axis gyroscope values, standard deviation of X and Y-axis linear acceleration, variance of Y-axis magnetometer values, variance of Z-axis gyroscope values, variance of Y-axis linear acceleration, skew of X-axis magnetometer values, skew of X-axis linear acceleration, kurtosis of Y-axis acceleration, kurtosis of heart rate, kurtosis of Y-axis magnetometer values, kurtosis of Y and Z-axis gyroscope values

Table 5. Impact of feature engineering & selection-IMU+HR dataset [Mean of attribute values]

Classifier	Selected features from IMU+HR Dataset [Mean of attribute values]
kNN	Mean of X, Y and Z-axis acceleration, Mean of heart rate, Mean of X and Z-axis magnetometer values, Mean of X and Y-axis gyroscope values, Mean of Y and Z-axis linear acceleration
Naïve Bayes	Mean of X, Y and Z-axis acceleration, Mean of heart rate, Mean of X and Z-axis magnetometer values, Mean of X and Y-axis gyroscope values, Mean of Y and Z-axis linear acceleration
Random Forest	Mean of X, Y and Z-axis acceleration, Mean of heart rate, Mean of X and Z-axis magnetometer values, Mean of X and Y-axis gyroscope values, Mean of X, Y and Z-axis linear acceleration
ANN	All
XGBoost	Mean of X, Y and Z-axis acceleration, Mean of heart rate, Mean of X, Y and Z-axis magnetometer values, Mean of Z-axis gyroscope values, Mean of X and Z-axis linear acceleration

3.1.3. Summary of feature engineering and selection

Table 6 summarizes the accuracies we obtained in the various cases we experimented with. Row (6) of Table 6 pertains to the performance figures obtained from using this feature set. It was observed that, in most cases, the parameters that exhibited higher significance were the mean and median of the attribute values.

Table 6. Summary of ML classifier accuracy

Row No.	Scenario	ANN	kNN	XGB	Naïve Bayes	Random Forest
1	IMU-Only (Baseline)	66	77	69	39	77
2	IMU-Only (Highest scores)	88.5	88.7	87.8	53.2	99.6
3	IMU-Only (Mean)	73	96.3	84.8	67.8	99.5
4	IMU-Only (Mean+Median)	85.7	94.3	85.3	66	99.7
5	IMU+ HR (Baseline)	69	89	72	40	93
6	IMU + HR (Highest scores)	87	91	89	56	99.6
7	IMU + HR (Mean)	84.5	97.1	85.5	67.8	99.8
8	IMU + HR (Mean+Median)	66	79	86.2	67	99.9

We note that the significance of the mean and median of the heart rate values were higher than others, and therefore, in order to reduce the number of features further via feature selection, in the next step, we selected the following features only for performance analysis:

- Mean of all IMU and HR features
- Mean and median of all IMU and HR features

We observe that in both IMU-Only dataset and IMU+HR dataset, feature extraction and feature selection reduce the dimensionality of the dataset, while improving the accuracy. In most cases, using only the mean of all attribute values results in maximum reduction in dimensionality, with no considerable drop to the accuracy. This is indicated by the rows (1) and (3), for the IMU-Only dataset, and rows (5) and (7) for the IMU+HR dataset.

3.2. Impact of inclusion of HR attribute

Prior to feature engineering, the inclusion of HR attribute improved the accuracy of ML classifiers. A similar observation is evident post feature extraction also-it was observed that all the classifiers performed better with the IMU+HR dataset, in the case when all features with highest scores were included in the feature vector Figure 1. Feature selection decreased the accuracy of some classifiers in the IMU+HR dataset, as shown in Table 6. However, in most cases, the ML classifiers performed better with the inclusion of HR attribute and feature selection, than on the IMU-Only dataset with no feature engineering. This is evident from a comparison of rows (1) and (7) in Table 6. Across all the cases we experimented with, we observe that random forest exhibited the best accuracy while considering mean and median of the features from IMU+HR dataset, with a window size of 20 and threshold value of 0.01.

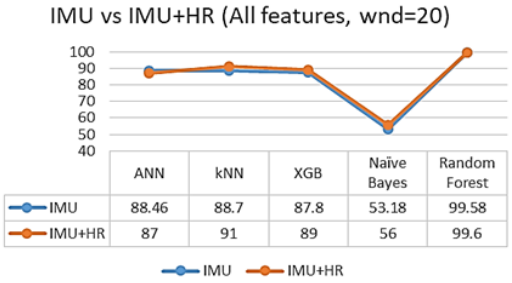


Figure 1. Comparison of ML classifiers-IMU vs IMU+HR

3.3. Application of ensemble methods for fall classification

From the results captured in the previous sections, we observe that ensemble classifiers such as random forest and XGBoost consistently performed better than other classifiers. In order to study the performance of ensemble classifiers further, we applied well-known ensembles such as AdaBoost, GradientBoosting and ExtraTree classifiers on the IMU+HR dataset post feature selection, where only the top 10 most significant features were included. This gave us accuracies of 73.4%, 76.9% and 98.8% respectively. Subsequently, we applied bagging, stacking and voting classifiers with different combinations of the base classifiers, on the IMU+HR dataset, after feature engineering and feature selection, to study their performance. In this section, we refer to a set of weak base classifiers (ANN, logistic regression, naïve bayes) and strong base classifiers (kNN, decision tree). Our observations are summarized below. Ensemble techniques typically require high computational costs and memory [28], and hence the applicability of the below results to a given system, should also be based on the classifiers' algorithmic efficiency vis-à-vis the system constraints.

3.3.1. Observations from bagging

Bagged ensemble classifiers with base classifiers from kNN, ANN, decision tree, logistic regression and naïve bayes were applied on the IMU+HR dataset. Bagging using decision trees gave us the highest accuracy of 97.9%. The accuracies of other bagging classifiers were not as high as that of decision tree: kNN (97.1%), logistic regression (70.3%), naïve bayes (41.8%) and ANN (88.9%). We observe that bagging classifiers do perform as good as or better than the individual base classifiers. However, there is no marked improvement in the performance of the bagging classifier when compared to the base classifiers. This could be because of the feature engineering and feature selection already applied on the dataset.

3.3.2. Observations from stacking

A stacked ensemble of kNN and decision tree, with the latter as the final estimator, gave us the best accuracy of 97.4%. All meta estimators gave us similar results across different combinations of stacked base classifiers, with only minor variations in the accuracies. Weak estimators performed better when paired with good estimators; however, weak estimators when grouped with other weak estimators, did not show significant improvement in their performance. The stacking ensemble with decision tree and kNN gave good accuracy for all the meta estimators we used.

3.3.3. Observations from voting

A hard-voting ensemble with decision tree, kNN and ANN as base classifiers gave us the best accuracy of 96.2%, while the accuracy of a soft voting ensemble with kNN and decision tree with weightages

in the ratio of 1.8:1, was 97.4%. We observe that an ensemble of weak classifiers did not show any improvement when voted together in pairs of two and three. At the same time, an ensemble of strong classifiers together with the weak ones, worked better than the weak base classifiers themselves, without any significant decrease in accuracy of the strong base classifiers. For example, the accuracy of the voting ensemble of naïve bayes and decision tree was 94.4%, when the individual accuracies were 53.2% and 94.6%. The accuracy of the voting ensemble of logistic regression and decision tree was 94.5%, when the individual accuracies were 69.6%% and 94.6%. Table 7 summarizes the accuracies we obtained in the best cases.

We also observe that, in general, soft voting methods with default equal weightages for the base classifier outputs, performed better than hard voting methods. For example, the accuracy of a hard-voting classifier with naïve bayes and kNN was 57.1%, while that of a soft voting ensemble for the same base classifiers was 92.7%. This difference in accuracy, in varying measures, was observed for all combinations of base classifiers that constituted the soft voting ensembles.

Table 7. Summary of ensemble techniques

Ensemble Technique	Base Classifiers	Accuracy
Bagging	Decision Tree	97.9%
Stacking	kNN, Decision Tree	97.4%
Voting (hard)	kNN, Decision Tree, ANN	96.2%
Voting (soft)	kNN, Decision Tree, (weightage ratio = 1.8:1)	97.4%

4. CONCLUSION

In this paper, we presented observations from our analysis of the impact of the following factors on the accuracy of ML classifiers: (i) Inclusion of vital signs parameters with IMU parameters in the feature vector; (ii) Feature engineering and feature selection based on statistical methods; and (iii) Use of ensemble techniques for fall detection. The performance of most base classifiers improved when combining heart rate attributes with IMU sensor parameters, than when using IMU sensor parameters independently. The implication of this is that since multiple sensors can be incorporated in a wearable device without increasing its form factor, the combination of IMU and heart rate sensor parameters is better suited to the design of a fall detection solution based on wearable devices. With feature engineering and feature extraction, when the mean and/or median of all attributes within a window were extracted and selected as input feature sets, the performance of the classifiers further improved in certain cases. The best accuracy was reported by random forest, when the feature set consisted of the only mean and median of IMU+HR observations, instead of all extracted features. We also note that further reduction in dimensionality was achieved by inclusion of only the mean of the attribute values, and since this does not result in a considerable drop in accuracy, this trade-off is worth including in the design of a wearable device based FDS. Dimensionality reduction is important in wearable device based FDS because it serves to reduce the power consumed by the devices during sensing, processing and communication. Since random forest and XGBoost, which are ensemble classifiers, consistenly showed better results than other classifiers, we focused on the impact of applying ensemble techniques using bagging, stacking and voting classifiers, on the dataset. We observe that the accuracy of the custom created ensemble classifiers, in general, was better than the constituent weak base classifiers in many cases. Therefore, for the dataset we used, an ensemble classifier could be more suited to detecting falls than individual classifiers. However, a detailed study on ensemble classifiers' algorithmic efficiency, to make the right trade-offs between computational complexity, memory consumption and accuracy of the classifiers, for the system under design, needs to be carried out. We are currently working on a prototype wearable device with appropriate sensors to enable real time collection of data for fall detection. As future work, we plan to integrate this prototype device with a system architecture for an end to end fall detection system. The observations from the experimental analysis described above will be used to make appropriate design choices while building the fall detection system.

REFERENCES

[1] A. T. Özdemir, B. Barshan, "Detecting falls with wearable sensors using machine learning techniques," *Sensors (Basel)*, 14(6):10691-10708, 2014. Doi: 10.3390/s140610691.

[2] S. Yu, H. Chen, R. A. Brown, "Hidden Markov Model-based fall detection with motion sensor orientation calibration: A case for real-life home monitoring," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 6, pp. 1847-1853, 2018. Doi: 10.1109/JBHI.2017.2782079.

[3] M. A. Guvensan, A. O. Kansiz, N. C. Camgoz, H. Turkmen, A. G. Yavuz, M. E. Karsligil, "An energy-efficient multi-tier architecture for fall detection on smartphones," *Sensors (Basel)*, 17(7):1487, 2017. Doi: 10.3390/s17071487.

[4] M. V. Albert, K. Kording, M. Herrmann, A. Jayaraman, "Fall classification by machine learning using mobile phones," *PLoS One*, 7: e36556, 2012. Doi: 10.1371/journal.pone.0036556.
[5] Y. Choi, A. S. Ralhan, S. Ko, "A study on machine learning algorithms for fall detection and movement classification," *International Conference on Information Science and Applications*, 2011, pp. 1-8. Doi: 10.1109/ICISA.2011.5772404.
[6] A. Jefiza, E. Pramunanto, H. Boedinoegroho, M. H. Purnomo, "Fall detection based on accelerometer and gyroscope using back propagation," *4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2017, pp. 1-6. Doi: 10.1109/EECSI.2017.8239149.
[7] F. Hossain, M. L. Ali, M. Z. Islam, H. Mustafa, "A direction-sensitive fall detection system using single 3D accelerometer and learning classifier," *International Conference on Medical Engineering, Health Informatics and Technology (MediTec)*, 2016, pp. 1-6. Doi: 10.1109/MEDITEC.2016.7835372.
[8] P. Vallabh, R. Malekian, N. Ye, D. C. Bogatinoska, "Fall detection using machine learning algorithms," *24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2016, pp. 1-9, doi: 10.1109/SOFTCOM.2016.7772142.
[9] X. Yang, A. Dinh, A, L. Che, "A wearable real-time fall detector based on Naive Bayes Classifier," *Proceedings of the 23rd Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, 2010.
[10] S. Zhao, W. Li, W. Niu, R. Gravina, G. Fortino, "Recognition of human fall events based on single tri-axial gyroscope," *IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, 2018, pp. 1-6. Doi: 10.1109/ICNSC.2018.8361365.
[11] J. He, S. Bai, X. Wang, "An unobtrusive fall detection and alerting system based on Kalman filter and Bayes network classifier," *Sensors*, 2017;17 (6): 1393, 2017. Doi:10.3390/s17061393.
[12] A. Chelli and M. Pätzold, "A machine learning approach for fall detection and daily living activity recognition," *IEEE Access*, vol. 7, pp. 38670-38687, 2019. Doi: 10.1109/ACCESS.2019.2906693.
[13] H. Wang, M. Li, J. Li, J. Cao, Z. Wang, "An improved fall detection approach for elderly people based on feature weight and Bayesian classification," *IEEE International Conference on Mechatronics and Automation*, pp. 471-476, 2016. Doi: 10.1109/ICMA.2016.7558609.
[14] P. Tsinganos and A. Skodras, "A smartphone-based fall detection system for the elderly," *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis, Ljubljana*, pp. 53-58, 2017. Doi: 10.1109/ISPA.2017.8073568.
[15] H. Kao, J. Hung, C. Huang, GA-SVM applied to the fall detection system, *2017 International Conference on Applied System Innovation (ICASI)*, 2017, pp. 436-439. Doi: 10.1109/ICASI.2017.7988446.
[16] A. Jahanjoo, M. N. Tahan, and M. J. Rashti, "Accurate fall detection using 3-axis accelerometer sensor and MLF algorithm," *in Proceedings of the 2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA)*, pp. 90–95, Shahrekord, Iran, April 2017.
[17] D. Genoud, V. Cuendet, J. Torrent, Soft fall detection using machine learning in wearable devices, *IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 501-505, doi: 10.1109/AINA.2016.124.
[18] G. Chetty, M. White, F. Akther, "Smart phone-based data mining for human activity recognition," *Procedia Computer Science*, 46:1181-1187, 2015, Doi: 10.1016/j.procs.2015.01.031.
[19] N. Zurbuchen, P. Bruegger and A. Wilde, "A Comparison of Machine Learning Algorithms for Fall Detection using Wearable Sensors," *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Fukuoka, Japan, pp. 427-431, 2020. Doi: 10.1109/ICAIIC48513.2020.9065205.
[20] G. Leoni, P. T. Endo, K. Monteiro, E. Rocha, I. Silva and T. G. Lynn, "Accelerometer-Based Human Fall Detection Using Convolutional Neural Networks," *Sensors (Basel)*, 19(7), 2019. pii: E1644, doi:10.3390/s19071644.
[21] M. Musci, D. Martini, N. Blago, T. Facchinetti and M. Piastra, "Online Fall Detection using Recurrent Neural Networks," 2018, ArXiv, abs/1804.04976.
[22] A. Ramachandran, K. R. Anupama, "A Survey on Recent Advances in Wearable Fall Detection Systems," *BioMed Research International*, vol 2020, doi: 10.1155/2020/2167160.
[23] B. Rodrigues, D. Salgado, M. Cordeiro, K. Osterwald, T. Freire, V. Lucena, E. Naves & N. Murray, "Fall Detection System by Machine Learning Framework for Public Health," *Procedia Computer Science*, 141. 358-365. Doi: 10.1016/j.procs.2018.10.189.
[24] F. Luna-Perejón, MJ Domínguez-Morales, A. Civit-Balcells, "Wearable Fall Detector Using Recurrent Neural Networks", *Sensors (Basel)*, 19(22):4885, 2019. Doi:10.3390/s19224885.
[25] D. Giuffrida, G. Benetti, D. De Martini and T. Facchinetti, "Fall Detection with Supervised Machine Learning using Wearable Sensors," *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Helsinki, Finland, pp. 253-259, 2019. Doi: 10.1109/INDIN41052.2019.8972246.
[26] A. Ramachandran, A. Ramesh, P. Pahwa, A. P. Atreyaa, S. Murari and K. R. Anupama, "Performance Analysis of Machine Learning Algorithms for Fall Detection," *2019 IEEE International Conference on E-health Networking, Application & Services (HealthCom)*, Bogota, Colombia, 2019, pp. 1-6. Doi: 10.1109/HealthCom46333.2019.9009442.
[27] A. Ramachandran, A. Ramesh and A. Karuppiah, "Evaluation of Feature Engineering on Wearable Sensor-based Fall Detection," *2020 International Conference on Information Networking (ICOIN)*, Barcelona, Spain, 2020, pp. 110-114. Doi: 10.1109/ICOIN48656.2020.9016479.
[28] L. Rokach, "Ensemble-based classifiers", *Artificial Intelligence Review*, 33, 1–39 (2010). Doi: 10.1007/s10462-009-9124-7.