






# A Split-Then-Join Lightweight Hybrid Majority Vote Classifier

Moses L. Gadebe<sup>1</sup> , Sunday O. Ojo<sup>2</sup> , and Okuthe P. Kogeda<sup>3</sup> 

<sup>1</sup> Tshwane University of Technology, Pretoria, Private Bag X680, Pretoria 0001, South Africa  
gadebeml@tut.ac.za

<sup>2</sup> Durban University of Technology, 41 Sultan Road, Greyville, Durban 4001, South Africa

<sup>3</sup> University of the Free State, P. O. Box 339, Bloemfontein 9300, South Africa

**Abstract.** Classification of human activities using smallest dataset is achievable with tree-oriented (C4.5, Random Forest, Bagging) algorithms. However, the KNN and Gaussian Naïve Bayes (GNB) achieve higher accuracy only with largest dataset. Of interest KNN is challenged with minor feature problem, where two similar features are predictable far from each other because of limited number of classification features. In this paper the split-then-join combiner strategy is employed to split classification features into first and secondary (KNN and GNB) classifier based on integral conditionality function. Therefore, top  $K$  prediction voting list of both classifier are joined for final voting. We simulated our combined algorithm and compared it with other classification algorithms (Support Vector Machine, C4.5, K NN, and Naïve Bayes, Random Forest) using R programming language with Caret, Rweka and e1071 libraries using 3 selected datasets with 27 combined human activities. The result of the study indicates that our combined classifier is effective and reliable than its predecessor Naïve Bayes and KNN. The results of study shows that our proposed algorithm is compatible with C4.5, Boosted Trees and Random Forest and other ensemble algorithms with accuracy and precision reaching 100% in most of 27 human activities.

**Keywords:** Split-then-join · Ensemble · KNN · Gaussian Naïve Bayes · Lightweight algorithm

## 1 Introduction

Human Activity Recognition is focused in classifying human activities based on sensor signals from accelerometer, gyroscope and GPS. A smartphone comes with such number of sensors and permits continuous monitoring of numerous physiological signals. However, due to limited storage, processing power and limited memory most researchers in Human Activity Recognition (HAR) use smartphone accelerometer to monitor patients to provide smart healthcare [1, 2]. Moreover, classification algorithms in HAR plays crucial role to classify human activities using sensors data. The classification algorithms also helps researchers to understand human behavior and their environment [1, 2]. However, classification algorithms differs in terms dataset requirements which have impact on storage, memory and processing power of devices. Naïve Bayes, KNN and Support

Vector Machines to accurately make predictions requires training dataset with more classification features [2, 3, 7]. On the other hand, algorithms such as C4.5, Random Forest and Bagging performs better with small dataset [8–13]. The evolution of ensemble algorithms, plays a crucial role in joining algorithms to improve their prediction accuracy. Researchers in [14] proposed number of combing strategies to join a bunch of algorithms, with the aim to increase their prediction. Most of ensemble techniques use combiner strategies (combiner, arbiter and hybrid strategies) proposed in [14]. Most researchers in [9, 12, 13] and [15] employed ensemble algorithms based on voting schemes using tree-based algorithms, because tree-based algorithms produce higher accuracy than simpler algorithms (Naïve Bayes and KNN). Hence, it is prudent to close the gap between tree-based algorithms and simpler algorithms. Ensemble algorithms based on combiner strategies gives the possibilities to improve prediction accuracy of simpler algorithms. In this paper split-then-join approach is proposed based on [14] strategies and integral conditionality function to join voting lists of first and secondary classifiers (Gaussian Naïve Bayes and KNN). The aim of this paper is to augment classification features to increase classification accuracy of data hungry algorithms. The split-then-join approach is simulated in R programming language for comparison using three benchmarking datasets. The remainder of our paper is fashioned as follows: In Sect. 2, related work in HAR is presented. The methodology and experimentation are presented in Sects. 3 and 4 respectively. Experimental results are presented in Sect. 5. Finally, Sect. 6 presents conclusion and future work.

## 2 Related Work

Bootstrap Aggregating known as Bagging was invented by Breiman [8]. The technique combines different numbers of tree decision (ID3, C4.5) to amalgamate various outputs thereby calculating averages from different decision trees. The employed trees have equal weights and each subsets of training dataset are chosen randomly. Therefore, each incorporated decision tree is taken as subset, then majority vote is computed on each averaged decision tree similar to [14] combiner strategy. Boosting, differently to Bagging uses different subsets of training dataset by reweighting in each iteration such that a single learning model is weighted to construct a final strong classifier. If input dataset with  $N$  classification features such that  $(n_i, k_i) i = 1, \dots, N$  where  $n_i$  is a feature vector and  $k_i$  is a labelled class  $k_i \in 1, \dots, T$ , then iterations are performed with weaker classifier  $f(n)$ . However, in Bagging,  $N$  features are indiscriminately selected with replacement in  $T$  iterations from training features [15, 16]. Therefore, weaker classifier is applied on the indiscriminately selected features and the resulting model is stored. But, Boosting focuses on creating stronger classifier compared to Bagging, by giving more influence on successful stronger models. After  $T$  iterations, the prediction is made using weighted voting of the predictions for each successful classifier. Random Forest was also introduced by Breiman in 2001 aimed at reducing the danger of over-fitting in constructing ensemble/combined models [8, 15]. One variant of Boosting is AdaBoost, it creates a linear combination of decision trees [15]. Random Forest classifiers consist of a collection of decision trees, because more classifiers are more likely to be well-classified and likely to give appropriate weight to most relevant features [8]. A Random

Forest creates a tree-like structure given as  $\{h(n, \Theta k), k = 1, \dots\}$  and  $\{\Theta k\}$  of individually distributed vectors, where each tree casts a unit vote for the most popular class at input  $n$ . That is, the prediction of class in training dataset is obtained by majority vote over the predictions of individual trees. Most research work in HAR, uses Random Forest and C4.5 as benchmarking algorithms because they were found to be reliable with smallest dataset [9, 11, 12].

Hence, researchers in [12] employed a group of classifiers to classify human activities using dataset collected from 20 participants using researcher's ASUS ZenFone 5 smartphone. During data collection participants used a smartphone inside their front pockets whilst performing human activities. The smartphone captured tri-axial values for each recorded human activities at frequency rate of 0.5 and 120 instances were collected per minute. Researchers in [12] extracted and annotated time domain features similar to [9] in order to train and test AdaBoost, C4.5, Support Vector Machines (SVM) and Random Forest using WEKA. Simulated algorithms (AdaBoost, C4.5, SVM and Random Forest) reported 98%, 96%, 95% and 93% in accuracy respectively [11]. Authors in [15, 16, 19] proposed ensemble techniques employing tree-oriented algorithms, because they were found to be reliable with smallest dataset and reported accuracy above 98%. However, researchers in [20] proposed a different ensemble model to join a group of expert Naïve Bayes algorithms and average the experts using weighted majority vote strategy. In their [20] work, the technique averages probabilities of each employed algorithm and observe each experts and true classes in a given sample in the dataset. Once, the conditional probabilities were learned their technique employed weighted voting to classifies each unknown sample. They [20] analyzed all expert responses which were jointly averaged to collective classify each sample. The final prediction of their technique is based on individuals experts. They [20] simulated their technique using 3 datasets (MFeat, Optodigit and Pendigit) in comparison to bagging and boosting algorithms. The results of their study, showed slight increment in accuracy close to 97% in all 3 datasets.

A closely related study is reported in [13], they proposed an ensemble technique to join Naïve Bayes Tree, KNN and C4.5 algorithms. The model follows a similar combiner strategy proposed in [14]. In the same way as in [20] the technique of [13] averages voting probabilities of three algorithms, then in each learner a posteriori probability is generated and a class with maximum posteriori is taken as voting hypothesis. In their simulations 10-fold cross validation was conducted on each of 28 datasets using WEKA. The results of their study revealed that their voting ensemble techniques outperformed individual simpler classifiers (Naïve Bayes and KNN) excluding C4.5 classifier. The result presented in [13] are comparable with our reported results in [22]. In this article, we expand our ensemble algorithm using split-then-join approach to join GNB and KNN in marriage of convenience using integral conditionality to address minor features challenges presented in [21–23].

### 3 Methodology

In this article, split-then-join approach presented in Fig. 1 is used to join KNN and Gaussian Naïve Bayes based on integral conditionality function to improve their prediction accuracy with reduced training dataset [22].

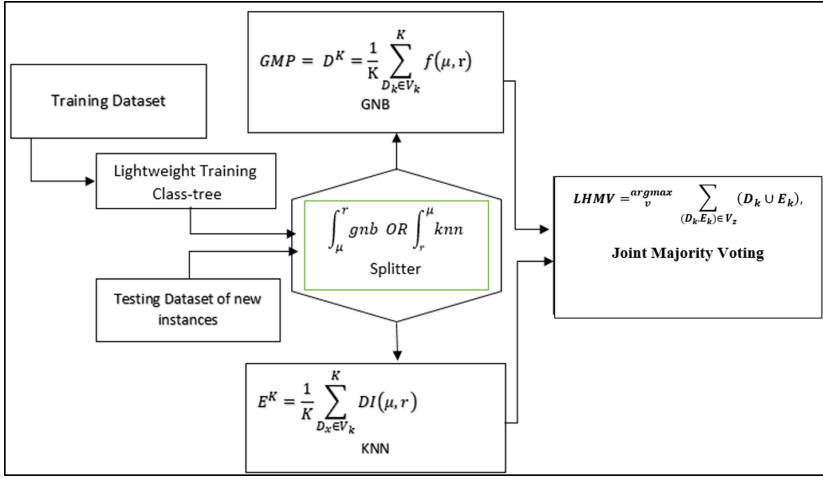


Fig. 1. Split-then-join combiner strategy

Our technique is aimed to address K neighborhood minor features problem, where two records that are closer to each other are recognized far from each other due to limited data point [7, 22]. Different to studies in [13, 14] and [20], we used bottom-up segmentation to select best classification features and thus compress dataset into reduced class-tree. Then, the reduced class-tree is therefore split to upper and lower classifier using our unique integral conditionality rule to accommodate GNB and KNN probability different structures. The split-then-join combiner strategies is presented in two stages as follows:

**a. Stage One: Lightweight Training Class-Tree**

In [22], we introduced lightweight class-tree to normalize and improve training complexity of lazy classifiers inspired by C4.5 [25]. The technique was borrowed from the C4.5 concept of best attribute selection. Input training dataset with multiple classes  $C = \{c_1, c_2, \dots, c_N\}$  represented as  $TrainerSet^{N \times M}$  class-tree is reduced using  $R$  reduction transformers based on our data segmentation technique defined by Eq. (1) and (2) [22]:

$$Data\ Segmentation^{R \times S} = \int_S^R \frac{1}{S} \sum_{x=1}^N p_x \tag{1}$$

where  $N$  is rows and  $M$  is features from input  $TrainerSet^{N \times M}$  and  $R$  is compressed subgroups each with  $S$  rows given as  $R = \{r_1, r_2, \dots, r_s\}$  of all mean averages using Eq. (2) [22]:

$$mean = \frac{1}{S} \sum_{x=1}^S i_x \tag{2}$$

Our data segmentation technique is expressed in Algorithm 1 to prepare reduced training class-tree before actual classification.

---



---

**Algorithm 1: Compressed Lightweight Class-Tree Algorithm**

---

1. **Required** : Multi-featured training dataset  $TrainerSet^{N \times M}$  as  $T$
2. **procedure**  $compressDataset()$
3. Set  $R$  to a value to reduce training dataset predictors
4. **Output**: reduced training class-tree  $reducedTree^{R \times M}$  to empty
5. Set  $N$  to number of rows in class  $K$
6. **for each** class  $K \in T$  **do**
7.  $S \leftarrow \lfloor \frac{N}{R} \rfloor$  //split the group into  $R$  groups
8. **cols**  $\leftarrow$  set to number of attributes in predictor  $I$
9. **Set row** = 1 //the rows of reduced number of groups
10. **for**  $j=1$  to **cols**//for each predictor compute sum and average
11. **for**  $i = 1$  to  $S$  **do** //number of reduced splitting group
12.  $sum \leftarrow sum + (I_{i,j})$  // Sum of each predictor in element  $j, i$
13. **end-for**
14.  $mean_{row,j} \leftarrow \frac{sum}{S}$  //compute mean based on equation 2
15.  $row \leftarrow row + 1$  //accumulates count of reduced tree rows
16.  $ReducedTree_{e_{row}} \leftarrow mean$  // Store each mean reduced group
17. **end-for**
18. **end-for**
19. **end**

---

The Algorithm expects a training dataset of any dimension in line 1 with reduction transformation value of  $R$  size in line 2. Then, the algorithm computes number of  $S$  rows for each  $R$  group in line 6. Thereafter, the algorithm sums all input predictor  $i_s$  in each subgroup  $R$  and computes the mean average from line 5 to 14.

### **b. Lightweight Hybrid Majority Vote Classifier**

In this stage, reduced class-tree of best mean features  $\phi = \{\mu_1, \mu_2, \dots, \mu_r\}$  for each class  $c_r$  is used as input training dataset. Our proposed LHMV presented in [22] uses reduced training dataset to train our classifier based on union of convenience. In this paper integral conditionality is employed to increase classification feature to minimize minor-features problems reported in [7, 22, 23, 25]. Such that voting results in the first and second classifiers prediction results are stored in  $D^k$  and  $E^k$  vectors and then joint into union of convenience based on majority voting principle defined by Eq. (3) [22]:

$$LHMV = \underset{v}{argmax} \sum_{(D_k, E_k) \in V_z} (D_k \cup E_k), \quad (3)$$

where  $K$  is top potential neighbors in  $D_k$  and  $E_k$  voting lists of predicted classes labels in a joint vote list  $V_z$ . The combination of GNB and KNN is joint probability function  $P(C \leq RT) \cup P(C > RT)$  of classification tree  $C$  with best mean  $\phi = \{\mu_1, \mu_2, \dots, \mu_r\}$

to classify real-time instances  $RT = \{r_1, r_2, r_3, r_j\}$  that factorizes as splitting integral function defined by Eq. (4):

$$P(C|RT) = \left( \sum_{\mu_r \leq r_j} \sum_{\mu_r > r_j} VCF(\mu_r, r_j) DUVF(\mu_r, r_j) \right) \tag{4}$$

where,  $VCF(\mu_r, r_j)$  is Vote Cast Function (VCF) known as probability approximation condition  $\mu_r \leq r_j$ , whereas  $DUVF(\mu_r, r_j)$  is the KNN Distance Unit Vote Function within supplementary integral condition  $\mu_r > r_j$  to accommodate all missed near-neighbors not within the first VCF conditionality. Our combiner strategy splits the input training dataset into first and secondary classifier then later joins their top  $K$  neighbors.

*First Classifier: Improved Gaussian Naïve Bayes*

As the first classier the GNB is improved and implemented for all best mean averages  $\phi = \{\mu_1, \mu_2, \dots, \mu_r\}$  in each class category  $c_r$  that are within first integral condition  $VCF(\mu_r, r_j)$  as proper distribution function defined by Eq. (5):

$$VCF(\mu_r, r_j) = \int_{\mu}^r p(\mu|r) \tag{5}$$

Expanded to Eq. (6):

$$VCF(\mu_r, r_j) = \left\{ \begin{array}{l} 1 : \mu_r \leq r_j \\ 0 : otherwise \end{array} \right\} \tag{6}$$

Provided that the integral condition  $VCF(\mu_r, r_j)$  is a proper Cumulative Distribution Function (CDF) of all best mean averages  $\mu_r$  within probability limit function  $f(\mu_r, r_j) : \rightarrow [0, 1]$  by  $f(\mu_r \leq r_j)$ , which is defined by Eq. (7):

$$P(I \leq RT) = f(\mu_r \leq r_j) = \left\{ \begin{array}{l} 1: \mu_r \leq i < r_j \\ 0 : \mu_r > r_j \end{array} \right. \tag{7}$$

As a result, the probability approximation function holds when the area under  $VCF(\mu_r, r_j)$  is in CDF neighborhood. Therefore, each best mean average  $\mu_r$  in class category  $c_r$  is approximated relative to real-time instances  $r_j$  as  $p(\mu_r|r_j)$  products indicated in Table 1.

The vector  $D^K$  accumulates all probability products  $p(\mu_r|r_j)$  per class category  $c_r$  defined by Eq. (8):

$$D^k = \prod_{\mu_r \leq r_j} p(\mu_r|r_j) \tag{8}$$

where  $p(\mu_r|r_j)$  is probability approximation of all  $\mu_r$  ordered as  $\{\mu_1 \leq \mu_2 \leq \mu_3 \leq \dots \leq \mu_r\}$  in relation to  $r_j$  within first integral condition  $f(\mu_r \leq r_j)$  per class category  $c_r$ .

**Table 1.** Joint vote cast function of Gaussian Nearest Neighbours

$VCF(c_r, r_j)$	$r_1$	$r_2$	$r_3$	$r_j$	$D^K$
$c_1$	$p(\mu_1 r_1)$	$p(\mu_2 r_2)$	$p(\mu_3 r_3)$	$p(\mu_r r_j)$	$(D_1) = \prod_{\mu_r \leq r_j} p(\mu_r r_j)$
$c_2$	$p(\mu_1 r_1)$	$p(\mu_2 r_2)$	$p(\mu_3 r_3)$	$p(\mu_r r_j)$	$(D_2) = \prod_{\mu_r \leq r_j} p(\mu_r r_j)$
$c_3$	$p(\mu_1 r_1)$	$p(\mu_2 r_2)$	$p(\mu_3 r_3)$	$p(\mu_r r_j)$	$(D_3) = \prod_{\mu_r \leq r_j} p(\mu_r r_j)$
$c_k$	$p(\mu_1 r_1)$	$p(\mu_2 r_2)$	$p(\mu_3 r_3)$	$p(\mu_r r_j)$	$(D_k) = \prod_{\mu_r \leq r_j} p(\mu_r r_j)$

Therefore, the standard deviation  $\sigma$  under the area of CDF is integral standard normal distribution, defined by equation (9) [22]:

$$GNB = f(\mu_r, r_j) = \int_{\mu_r}^r \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r-\mu_r)^2}{2\sigma^2}} \quad (9)$$

where  $r$  is a real-time instance and  $\mu_r$  is mean average from reduced dataset and  $\sigma$  is a standard deviation computed using Eq. (10):

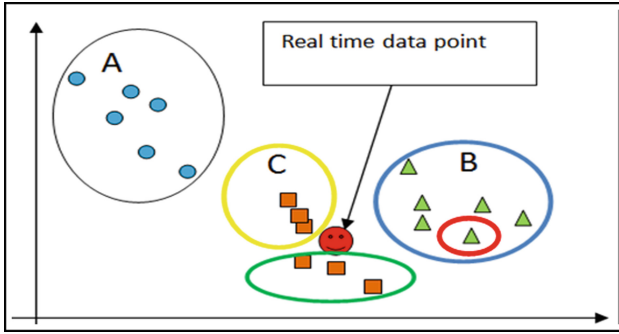
$$Standard\ deviation - \sigma = \sqrt{\frac{1}{k-1} \sum_{r=1}^k (r_j - \mu_r)^2} \quad (10)$$

We modified and transformed the GNB into Gaussian Majority Probability (GMP) defined by Eq. (11):

$$GMP = D^K = \frac{1}{K} \sum_{D_k \in V_k}^K f(\mu_r, r_j) \quad (11)$$

where  $f(\mu_r, r_j)$  is integral conditional function of all probability approximations  $p(\mu_r|r_j)$  as majority vote list  $D^k$  within  $f(\mu_r \leq r_j)$ , whereas  $K$  is number of nearest neighbors in voting list  $V_k$ . Figure 2 portrays possible neighbors given A, B and C classes where their mean predictors  $\phi = \{\mu_1, \mu_2, \dots, \mu_r\}$  are within probability limit  $f(\mu_r \leq r_i)$ .

The data points circled in green in class C are within first limit function  $f(\mu_r \leq r_j)$  and are accumulated in  $D^k$  voting list. However, those circled in yellow are missed neighbors outside probability limit  $f(\mu_r \leq r_j)$ . Most data points circled in blue are near-missed neighbors in class B, except one feature circled in red. When looking closely in Fig. 2, class B has 5 potential neighbors which could have resulted to a majority class in



**Fig. 2.** Closest neighbor data point

relation to real time  $r_j$ ; but it is not the case due to  $f(\mu_r \leq r_j)$  condition, thus resulting into to minor feature problem because eligible features above  $\mu_r > r_j$  are discarded.

*Second Classifier: Improved Supplementary KNN Classier*

In the second classifier all eligible near-missed predictors with smallest distance closer to real-time data point  $r_j$  are preserved by employing secondary KNN Distance Unit Vote Function (DUVF) defined by Eq. (12):

$$DUVF(\mu_r > r_i) = DI(\mu_r, r_j) \tag{12}$$

where  $DI$  is smallest similarity distance of all best mean  $\phi = \{\mu_1, \mu_2, \dots, \mu_r\}$  in relation to real time instances  $RT = \{r_1, r_2, r_3, r_j\}$  satisfying  $f(\mu_r > r_i)$  condition. We modified the distance  $DI$  to include integral limit function, such that only best mean  $\mu_r$  greater than real-time  $r_j$  are accommodated using Eq. (13) (see Table 2)

$$DI(I > r_i) = \sqrt{\sum_{x=1}^k \int_{r_x}^{\mu_x} (\mu_x - r_x)^2} \tag{13}$$

**Table 2.** KNN distance vote cast unit

$DI(c_r > r_i)$	$r_1$	$r_2$	$r_3$	$r_j$	Smallest Distance $E^K$
$c_1$	$DI(\mu_1 - r_1)^2$	$DI(\mu_2 - r_2)^2$	$DI(\mu_3 - r_3)^2$	$DI(\mu_r - r_j)^2$	$(E_1) = \sqrt{\sum_{x=1}^k \int_{r_x}^{\mu_x} (\mu_x - r_x)^2}$
$c_2$	$DI(\mu_1 - r_1)^2$	$DI(\mu_2 - r_2)^2$	$DI(\mu_3 - r_3)^2$	$DI(\mu_r - r_j)^2$	$(E_2) = \sqrt{\sum_{x=1}^k \int_{r_x}^{\mu_x} (\mu_x - r_x)^2}$
$c_i$	$DI(\mu_1 - r_1)^2$	$D2(\mu_2 - r_2)^2$	$DI(\mu_3 - r_3)^2$	$DI(\mu_r - r_j)^2$	$(E_k) = \sqrt{\sum_{x=1}^k \int_{r_x}^{\mu_x} (\mu_x - r_x)^2}$



Consequently, producing a vector  $E^K$  of all top  $K$  smallest neighbors within the secondary integral  $f(\mu_r > r_i)$  condition to accumulate all smallest nearest neighbors based on  $DI$  distance of Eq. (13) for a specific class  $c_r$ . Finally, both majority vote lists  $D^K$  and  $E^K$  in Eq. (11) and (13) are joint as majority vote of convenience  $JV^K = D^K \cup E^K$  known as LHMV defined by Eq. (14):

$$LHMV = JV^K = \underset{v}{argmax} \sum_{(D_k, E_k) \in V_z} (D_k \cup E_k), \quad (14)$$

where,  $V_z$  is a majority vote vector with predicted class categories parallel to  $D_k$  in conjunction with voting list  $E_k$ . Thus, the Eq. (14) computes a majority class from  $D_k$  and  $E_k$  voting list, such that a class category with more votes is assigned to new instance as class label. The LHMV Eq. (14) is expanded into Algorithm 2.

---

**Algorithm 2: Lightweight Hybrid Majority Vote Algorithm**


---

1. **Require** :  $TrainerSet^{N \times M}$  training dataset each with combined
  2. **Input** :
  3. Set  $RT^{Row \times J} \leftarrow extractRealTimeFeature(acce.x, acce.y, acce.z) // accelerometer X,Y,Z$
  4. Set  $R$  to value to resize training dataset rows
  5.  $reducedTree^{R \times M} \leftarrow compressDataset(TrainerSet^{K \times N}, R) // based on Algorithm 1$
  6. **Output** : initialise majority vote  $JV^K$  to empty
  7. Set  $K$  to top nearest neighbours to control majority vote
  8. **for** each testRow  $\in RT^{Row \times J}$  **do** // select each row real time  $\{r_1, r_2, r_3, r_j\}$
  9. select each  $r_j$  from testRow // select each attribute in testRow
  10. **for** each  $\phi$  in  $C_k \in \mathcal{D}$  **do** // for each human activity in reduced training set
  11. **for**  $\mu_r \in \phi$  **do** // for each multi-featured attribute in class category
  12. **if**  $\mu_r \leq r_j$  **then** // first integration limit for Gaussian Normal Distribution
  13.  $\sigma \leftarrow standardDeviation(\mu_r, r_j) // compute using equation .10$
  14.  $D_k \leftarrow D_k \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_j - \mu_r)^2}{2\sigma^2}} // Gaussian probability approximation Naïve Bayes equation.8$
  15. **end-if**
  16. **if**  $\mu_r > r_j$  **then** // Second integration supplementary condition
  17.  $E_k \leftarrow E_k + \left( \sqrt{(\mu_r - r_j)^2} \right) // Smallest distance probability using KNN equation$
  18. **end-if**
  19. **end-for**
  20. **end-for**
  21.  $LHMV = JV^K = \underset{v}{argmax} \sum_{(D_k, E_k) \in V_z} (D_k \cup E_k), // joint votes list of  $D_k \cup E_k$$
  22. **end-for**
- 

Our LHMV algorithms require training dataset and test dataset to classify human activities. Dataset can be divided into training and test dataset using specific ratio 0.7:0.30 or 0.6:0.4 or 50:50 or any splitting ratio. In line 5, training  $TrainerSet^{N \times M}$  is reduced to training class-tree  $reducedTree^{R \times M}$  using Algorithm 1 as preloaded training dataset. Therefore, each row of real time test vector is extracted from dataset  $RT^J$  in line 8. Then, probability approximation is estimated between test attribute  $r_j$  and training attribute

mean  $\mu_r$  to compute standard deviation  $\sigma$  for each human activity class category  $C_r$  in training dataset, see line 13 and line 14 of Algorithm 2. Thereafter, each probability approximation is accumulated into  $D_k$  only if the first limit function  $f(\mu_r \leq r_j)$  is met otherwise, smallest distance is computed and accumulated into  $E_k$  vector. Thereafter,  $D_k$  and  $E_k$  majority neighbors are sorted in descending and ascending order respectively and joined into marriage of convenience as  $JV^K$  majority vote vector in line 21. Lastly, the first  $K$ th human activity class in each vector  $D_k$  and  $E_k$  are matched and any human activity category with more votes is assigned to new instance  $r_j$  as a predicted class.

## 4 Experimentation

We selected four HAR datasets consisting of raw continuous random variables with 27 combined human activities classes. All the datasets PAMAP2, WISDM and Dataset-HAR-PUC-Rio-Ugolino (PUCRU) were downloaded from UCI of machine learning repository. Thereafter, we removed unimportant attributes such as subject id, gender, age, BMI, weight and height. We then normalized all datasets by removing missing values and capped features to 302 rows per human activity as listed in Table 4. We converted all datasets into Comma Separated Values (CSV) files to meet requirements of R programming language [26, 28, 29].

**Table 3.** Publicly available human activity recognition dataset

Source	Dataset	Sensor	Features	Instances	Classes
[4]	PAMAP2	3 Colibri wireless inertial measurement units	9	3624	12 classes by 302 instances
[29]	WISDM	Smartphone and Smartwatch Accelerometer	6	5436	12 classes by 302 instances
[30]	PUCRU	Wearable devices Accelerometer	12	1208	5 classes by 302 instances

We simulated our LHMV algorithm and compared it with SVM, C4.5, KNN, Boosted Tree (BT), Naïve Bayes and Random Forest (RT) in R programming language. The comparison is conducted in R programming language using Caret, e1071 and Rweka libraries [26, 28]. During simulation we used 60 rows as transformation reductions per human activity to compress every dataset to best mean classification class-tree similar to [23]. On each existing R classifiers we used R default settings. In our simulation setup, we used cross validation approach proposed by Kuhn [26, 28]. We employed K-fold cross validation approach to randomly partition each dataset, one after another, into equal K sub-sets; such that K-1 is used as training and K subset is used as testing on each personalized dataset. The cross-validation algorithm iterates K times, such that all K-fold instances are used as both training and exactly once as testing. The results of cross validation are presented in precision, accuracy, recall and f-measure.

## 5 Experimental Results

We present the results of comparison (LHMV against KNN, SVM, C4.5, BT, and Random Forest (RF)) using three selected datasets given in Table 3. All algorithms using WISDM dataset (consisting of 6 tri-axial values) achieved accuracy, precision, recall and f-measure of 100% across all human activities as shown in Table 4.

**Table 4.** Comparison results using WISDM dataset

Measurements	SVM	C4.5	RF	BT	KNN	NB	LHMV
Accuracy	0	100	100	100	100	100	100
Precision	0	100	100	100	100	100	100
Recall	0	100	100	100	100	100	100
F-Measure	0	100	100	100	100	100	100

However when PAMAP2 dataset (consisting 9 raw tri-axial attributes from accelerometer, gyroscope and magneto devices) is used, the Random Forest, C4.5 and Boosted Tree achieved 100% in all human activities in accuracy, precision and recall as shown in Table 5.

**Table 5.** Comparison results using PAMAP2 dataset

Measurements	SVM	C4.5	RF	BT	KNN	NB	LHMV
Accuracy	97	100	100	100	97	97	90
Precision	97	100	100	100	97	97	86
Recall	97	100	100	100	100	100	98
F-Measure	97	100	100	100	98	98	91

The results are similar to preliminary results of tree-based and ensemble algorithms presented in [10, 12], but with improved accuracy, precision and recall above 97% in KNN and Naïve Bayes. Algorithms SVM and KNN reported nil in all human activities as shown in Table 6 using PUCRU training instances whereas all other algorithms, including our LHMV, reported precision, accuracy, recall and f-measure of 100% in static human activities (sitting and standing).

The improved classification accuracy in simpler algorithms is owed to usage of largest training datasets; a confirmation that Naïve Bayes and KNN performs optimally than other sophisticated algorithms using largest dataset [22, 23]. The similarity between Naïve Bayes and LHMV in all the presented results is owed to the implementation of GNB as our first classifier and KNN as second classifier using split-then-join strategy. As observed, failure of one classifier (KNN) as shown in Table 6 does not affect the prediction of another classifier unless both classifiers fails. Overall, we can therefore conclude

**Table 6.** Comparison results using PUCRU dataset

Measurements	SVM	C4.5	RF	BT	KNN	NB	LHMV
Accuracy	0	100	100	100	0	99	83
Precision	0	100	100	100	0	100	83
Recall	0	100	100	100	0	99	83
F-Measure	0	100	100	100	0	100	83

that our LHMV is reliable, effective using reduced classification features of different sizes and is competitive with other algorithms. Our novel split-then-join strategy is effective and suitable to join simpler algorithms based on integral conditionality as compared other strategies presented in [13, 14, 20]. Moreover, the experimental results are competitive with voting ensemble strategies implemented in [10–13, 20]. The results confirm that ensemble algorithm increases predictions accuracy and outwit simpler algorithms (SVM, KNN and Naïve Bayes) [14, 16].

## 6 Conclusion and Future Work

The split-then-join approach to join GNB and KNN algorithms is presented in this paper. The split-then-join approach compresses and split the training dataset into first and second classifiers and then join their voting lists in marriage of convenience for final prediction. The results of our simulations showed that our LHMV is effective and competitive with tree-oriented and other ensemble algorithm yet using small and reduced training dataset. We can conclude that our split-then-join approach as the union of convenience improved classification accuracy, precision, recall and f-measure of KNN and Naïve Bayes classifiers. The results reveals that if one algorithm fails, it does not impact the prediction unless both algorithms fails. In all training datasets, our algorithm reached the accuracy and precision between 80% and 100%. In future, we intend to evaluate time-complexity of our LHMV to determine its viability to be implemented on resources constraint smartphone with reduced small training instances.

## References

1. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: ESANN, pp. 437–442 (2013)
2. Zhang M, Sawchuk AA. USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, 5 September 2012, pp. 1036–1043. ACM (2012)
3. Parkka, J., Ermes, M., Korpipaa, P., Mantjarvi, J., Peltola, J., Korhonen, I.: Activity classification using realistic data from wearable sensors. *IEEE Trans. Inf Technol. Biomed.* **10**(1), 119–128 (2006)
4. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: *Wearable Computers (ISWC), 2012 16th International Symposium on* 18 June 2018, pp. 108–109. IEEE (2012)

5. Su, X., Tong, H., Ji, P.: Activity recognition with smartphone sensors. *Tsinghua Sci. Technol.* **19**(3), 235–249 (2014)
6. Mannini, A., Intille, S.S., Rosenberger, M., Sabatini, A.M., Haskell, W.: Activity recognition using a single accelerometer placed at the wrist or ankle. *Med. Sci. Sports Exerc.* **45**(11), 2193 (2013)
7. Kaghyan, S., Sarukhanyan, H.: Activity recognition using K-nearest neighbor algorithm on smartphone with tri-axial accelerometer. *Int. J. Inform. Models Anal.* **1**, 146–156 (2012)
8. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
9. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. *ACM SIGKDD Explor. Newsl.* **12**(2), 74–82 (2011)
10. Catal, C., Tufekci, S., Pirmitt, E., Kocabag, G.: On the use of ensemble of classifiers for accelerometer-based activity recognition. *Appl. Soft Comput.* **31**(37), 1018–1022 (2015)
11. Daghistani, T., Alshammari, R.: Improving accelerometer-based activity recognition by using ensemble of classifiers. *Int. J. Adv. Comput. Sci. Appl.* **7**(5), 128–133 (2016)
12. Gupta, S., Kumar, A.: Human activity recognition through smartphone's tri-axial accelerometer using time domain wave analysis and machine learning. *Int. J. Comput. Appl.* **127**(18), 22–26 (2015)
13. Gandhi, I., Pandey, M.: Hybrid Ensemble of classifiers using voting. In: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), pp. 399–404. IEEE, October 2015
14. Chan, P.K., Stolfo, S.J.: Experiments on multistrategy learning by meta-learning. In: Proceedings of the Second International Conference on Information and Knowledge Management, pp. 314–323. December. 1993
15. De Stefano, C., Fontanella, F., Di Freca, A.S. A novel naive bayes voting strategy for combining classifiers. In: 2012 International Conference on Frontiers in Handwriting Recognition, pp. 467–472. IEEE, September 2012
16. Reiss, A., Hendeby, G., Stricker, D.: A competitive approach for human activity recognition on smartphones. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2013), 24–26 April, 2013, Bruges, Belgium, pp. 455–460. ESANN (2013)
17. Reiss, A.: Personalized mobile physical activity monitoring for everyday life. Ph.D. thesis in Computer Science, Technical University of Kaiserslautern (2014)
18. Bao, L., Intille, S.: Activity recognition from user-annotated acceleration data. In: International Conference on Pervasive Computing, pp. 1–7 (2004)
19. Ravi, N., Dandekar, N., Mysore, P., Littman, M.L.: Activity recognition from accelerometer data. In: AAAI 2005 July 9 (vol. 5, No. 2005), pp. 1541–1546 (2005)
20. Lockhart, J.W., Weiss, G.M.: Limitations with activity recognition methodology & data sets. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, 13 September, pp. 747–756. ACM (2014)
21. Gadebe, M.L., Kogeda, O.P., Ojo, S.O.: Personalized real time human activity recognition. In: 2018 5th International Conference on Soft Computing & Machine Intelligence (ISCMI), pp. 147–154. IEEE, November 2008
22. Gadebe, M.L., Kogeda, O.P.: A lightweight hybrid majority vote classifier using Top-k dataset. In: Patel, K.K., Garg, D., Patel, A., Lingras, P. (eds.) *Soft Computing and its Engineering Applications*. *icSoftComp 2020*, CCIS, vol. 1374, pp. 182–196. Springer, Singapore (2021). [https://doi.org/10.1007/978-981-16-0708-0\\_16](https://doi.org/10.1007/978-981-16-0708-0_16)
23. Kose, M., Incel, O.D., Ersoy, C.: Online human activity recognition on smart phones. In: Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data 16 April 2012, vol. 16, No. 2012, pp. 11–15 (2012)
24. Kuhkan, M.: A method to improve the accuracy of k-nearest neighbor algorithm. *Int. J. Comput. Eng. Inf. Technol.* **8**(6), 90 (2016)

25. Ruggieri, S.: Efficient C4. 5 [classification algorithm]. *IEEE Trans. Knowl. Data Eng.* **14**(2), 438–444 (2002)
26. Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. *Stat. Survey* **4**, 40–79 (2010)
27. Kuhn, M.: Building predictive models in R using the caret package. *J. Stat. Softw.* **28**(5), 1–26 (2008)
28. Kuhn, M.: A Short Introduction to the caret package. *R Found Stat. Comput.*, pp.1–10 (2015)
29. Weiss, G.M., Yoneda, K., Hayajneh, T.: Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access* **7**, 133190–133202 (2019)
30. Ugulino, W., Cardador, D., Vega, K., Velloso, E., Milidiú, R., Fuks, H.: Wearable computing: accelerometers’ data classification of body postures and movements. In: Barros, L.N., Finger, M., Pozo, A.T., Giménez-Lugo, G.A., Castilho, M. (eds.) *SBIA 2012. LNCS (LNAI)*, pp. 52–61. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34459-6\\_6](https://doi.org/10.1007/978-3-642-34459-6_6)