Check for updates

METHOD ARTICLE

Recognizing human activities using light-weight and effective machine learning methodologies [version 1; peer review: awaiting peer review]

Keerthi Varadhi [ID][1], Chinta Someswara Rao[2], GNVG Sirisha[2], Butchi Raju katari[1]

[1]CSE Department, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, TELANGANA, 500090, India
[2]CSE Department, SRKR Engineering College, Bhimavaram, ANDHRA PRADESH, 534204, India

Open Peer Review

Approval Status *AWAITING PEER REVIEW*

Any reports and responses or comments on the article can be found at the end of the article.

Abstract
Background

Human activity recognition is a dynamic and challenging task. It is a large field of research and development. It involves predicting the movement of a person from the raw sensor data using a machine learning model. To accurately detect human activities for e-health systems, several research attempts have been carried out using data mining and machine learning techniques, but there still is room to improve the performance. To this aim, human activities such as walking, standing, laying, sitting, walking upstairs, walking downstairs are predicted using prominent machine learning models.
The aim of human activity recognition is examining actions from photos or video clips. This serves as the driving force behind human activity identification systems' aim to accurately classify input data into the relevant activity category.
Methods
Six machine learning techniques, including decision tree, random forest, linear regression, Naïve bayes, k-nearest neighbour, and neural networks algorithms, were used for human activity recognition.
Results
The performance of decision tree, random forest, linear regression, Naïve bayes, k-nearest neighbor, and neural network algorithms was assessed with a human activity recognition dataset. From the results, the random forest classifier and neural network gave good results, whereas the Naïve bayes result was not satisfying.
Conclusions
We classified the SITTING, STANDING, LAYING, WALKING, WALKING_DOWNSTAIRS, WALKING_UPSTAIRS activities with machine learning techniques with 98% of accuracy

Keywords
Classification; Land Cover ; Deep Learning; CNN

This article is included in the Computational Modelling and Numerical Aspects in Engineering collection.

Corresponding authors: Keerthi Varadhi (varadhikeerthi@gmail.com), Chinta Someswara Rao (chinta.someswararao@gmail.com), Butchi Raju katari (butchiraju.katari@gmail.com)

Author roles: Varadhi K: Conceptualization, Methodology; **Someswara Rao C**: Methodology, Validation; **Sirisha G**: Writing – Original Draft Preparation; **katari BR**: Formal Analysis, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

How to cite this article: Varadhi K, Someswara Rao C, Sirisha G and katari BR. **Recognizing human activities using light-weight and effective machine learning methodologies [version 1; peer review: awaiting peer review]** F1000Research 2023, **12**:247 https://doi.org/10.12688/f1000research.124164.1

First published: 06 Mar 2023, **12**:247 https://doi.org/10.12688/f1000research.124164.1

Introduction

The task of recognizing the present physical action done by one or more users from a series of observations is known as human activity recognition. These observations are made while the user is doing any action in the defined environment. The rise of omnipresent, wearable, and persuasive computing has spawned a variety of new uses. Human activity recognition has evolved into a significant technology that is altering people's everyday routines. The notion of human activity recognition has recently gained popularity, resulting in a variety of applications such as assistive technology, health and fitness tracking, elder care, and automated surveillance. Furthermore, activity recognition research has progressed at such a quick pace that it is already being used for purposes other than activity recognition like monitoring elderly people or prison inmates.

Mobile phone sensors play an important part in making smart phones more useful and aware of their surroundings such as space, the presence of lamp, rubbish, among others. As a result, most smart phones have a variety of integrated sensors. It is feasible to get a large quantity of information on a person's everyday life and activities using this method. Among these sensor devices are accelerometer and gyroscope sensors. We can capture the motion of an item with the right usage of these sensors. As a result, machine learning algorithms may be used to anticipate human action based on sensor data. The information is constantly monitored. Human activity recognition is capable of identifying a variety of actions, both basic and complicated.

Literature review

Some of the literature papers on human activity recognition are included in this section. The details are depicted in Table 1.

Methods

The methodology for human activity recognition comprises four phases: input, data cleaning, data splitting, and classification and validation. Human activity recognition structure is shown in Figure 1.

Table 1. Evaluation of various methods.

First name	Last name	Grade
Authors	**Description**	**Future perception**
Schuldt et al.,[1]	Support vector machine (SVM) is used to recognize activities.	Neural network techniques must be utilized to increase the recognition rate.
Laptev et al.,[2]	For activity recognition, an SVM with a multichannel Gaussian kernel is utilised.	We need to enhance the recognition rate here as well.
Yamato et al.,[3]	With the feature vector, HMM is employed for activity recognition.	An increase in the rate of identification is required.
Oliver et al.,[4]	CHMM is a tool for recognising activities.	We need to enhance the recognition rate here as well.
Natarajan et al.,[5]	Human action is detected in this article utilising Conditional Random Fields.	The activity recognition needs to be improved.
Ning et al.,[6]	Human activity is detected using a typical way in this article.	We need to enhance activity identification here as well.
Vali M et al.,[7]	The authors combined HMM and CRF in their study.	It is necessary to boost the rate of recognition.
R. Madharshahian et al.,[8]	The authors utilized multinomial LR in this study.	We need to enhance activity identification here as well.
R.Kiros et al.,[9]	The authors employed consecutive RNNs in their study.	To improve the rate of identification, more layers must be added.
A. Grushin et al.,[10]	The authors utilized consecutive RNNs in this study as well.	More layers are required to improve the identification rate in this case as well.
Veeriah et al.,[11]	The authors employed differential RNN in this study.	More layers are required to boost the rate of activity detection in this case as well.
Du W. Wang, and L. Wang, et al.,[12]	The authors employed hierarchical RNNs in their study.	To improve the rate of activity recognition, DNN, i.e. LSTM, is necessary.
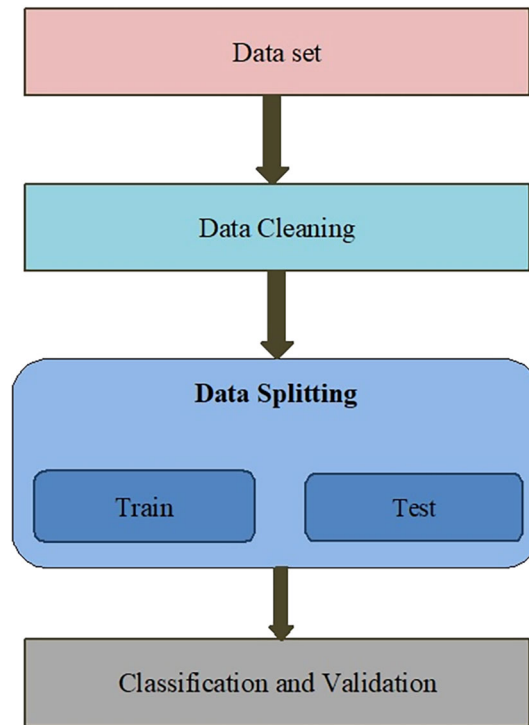
Figure 1. Human activity recognition structure.

Input: The dataset downloaded from the UCI dataset repository was the input of our study. The dataset contained information about triaxial acceleration from an accelerometer and triaxial angular velocity from a gyroscope. The dataset contains totally 10,299 data entries and 561 features.

Data cleaning: Data cleaning means removing the unnecessary columns and removing or replacing the inaccurate records from the dataset. In data cleaning, an encoding technique is applied to convert categorical data into numerical data. In our dataset, all feature values were numerical, hence there was no need to apply an encoding technique. Data cleaning prevents an error from happening again.

The first step in data cleaning is removing unwanted observations from the dataset. These unwanted data include irrelevant and duplicate data. Duplicate data most frequently arises when we combine data from multiple resources; irrelevant data refers to data that do not fit to the problem.

The second step is to handle the missing data. This can be done in two ways:

- Removing the record

- Filling the missing values based on other observations

Removing the record and filling the missing value manually were not fully optimal because we were losing some information; for example, range might be replaced with another value. Hence, for missing numerical data we had to take the following two steps:

- Flag the observation with an indicator value of missingness

- Fill the observation with 0 just to ensure that there are no missing value

By using the technique of flagging and filling, the algorithm can estimate the missing value optimally.

Data splitting: In this phase the data is segmented into two components

- Training data

- Testing data

Training data: The data which was used to train the model. We took 60% of the data as the training data.

Testing data: The data which was used to test our model is called testing data. We have taken 20% of the data from the dataset as testing data.

Classification and validation: The classification and validation was further comprised of two phases

- Classification algorithms

- Evaluation metrics

Classification: Classification is the process predicting the target class of new observation based on the training data. The dataset we selected contains the target class hence we implemented the model using classification algorithms. We used the following algorithms

- Naïve Bayes[13]

- K-nearest neigbours[14]

- Decision tree[15]

- Random forest[16]

- Simple logistic regression[17]

- Neural network[18]

Decision tree: This classification uses tree representation to solve a problem. Each internal node represents the features, branches represent the outcome of the feature and leaf node represent target class labels.

Algorithm
1. Identify the best attribute and make it as root.
2. Subdivide the training set into sections.
3. Repeat the previous two stages until all of the tree's branches have leaf nodes.

Attribute selection measures: We had to select the best attribute for node at each level. This can be done by two measures

- Information gain

- Gini Index

Information gain is mathematically defined as Equation 1

$$Entropy = -\sum P_i \times \log_2 P_i \qquad (1)$$

where P_i is the probability of class i

Hence from entropy, information gain is mathematically represented as

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_i|}{|S|} . \text{Entropy}(S_v) \qquad (2)$$

Gini index: It measures how often a randomly chosen element would be incorrectly classified.

$$\text{Gini Index} = 1 - \sum_j P^2{}_j \qquad (3)$$

As our dataset were unbalanced, meaning there were multiple classes, a decision tree made use of information gain to select the best attribute.

Logistic regression: Logistic regression is a statistical method used for classification which falls under supervised learning. It is used for predictions based on probability concepts. It analyses data set and take discrete independent input values and give a single output.

Logistic function: Logistic function is also known as the sigmoid function, which takes a real-value number as input and maps it into a value between 0 and 1, but not exactly at those limits. It maps predictions to probabilities.

By using the sigmoid function, if we give an input value on x, then it predicts the target value on the y axis.

$$\text{Logistic function} = \frac{L\left(e^{k(x-x_0)}\right)}{1 + e^{k(x-x_0)}} \qquad (4)$$

$$\text{Sigmoid function} = \frac{e^x}{1 + e^x} \qquad (5)$$

Here $k = 1$, $x_0 = 0, L = 1$

Algorithm
1. Plot the labelled data
2. Draw the regression curve
3. Find out the best fitted curve using maximum likelihood estimator (MLE)

  - Convert y-axis probability scale to log (odds)
  - Assume regression line and scale paper data to regression line
  - Apply sigmoid function

Naïve Bayes classifier: Naïve is a probabilistic classification algorithm which is based on Bayes theorem. It assumes that the occurrence of an instance is independent of other instances.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad (6)$$

The conditional probability of an object with feature vector $x_1, x_2, \dots \dots, x_n$ belongs to a particular class C_i, and it is calculated with Equation 7.

$$P(C_i|, x_2, \dots \dots, x_n) = \frac{P(x_1, x_2, \dots \dots, x_n | C_i) . P(C_i)}{P(x_1, x_2, \dots \dots, x_n)} \text{ for } 1 \le i \le k \qquad (7)$$

Algorithm
1. The data set is used to construct a frequency table.
2. By computing the probabilities of all the elements in the dataset, a likelihood table is constructed.
3. Posterior probability is calculated for each class by using (1).
4. The class with the highest posterior probability is the output.

Gaussian naïve Bayes: As values of each feature of our data set are continuous, we use the Gaussian distribution, which is also called the normal distribution.

$$P(A/B) = \frac{1}{\sqrt{2\pi\sigma_B^2}} \exp\left(-\frac{(A-\mu B)^2}{2\sigma_B^2}\right) \qquad (8)$$

Where μ is the mean of all the values a feature and σ is the standard deviation.

K-nearest neighbours is one of the essential classification algorithms. It falls under the category of supervised learning. It assumes that similar data are close to each other. Here the data is classified into groups based on an attribute. It is used in applications like pattern recognition, intrusion detection and data mining.

Algorithm

Let N be the number of training samples of data and U is the unknown point.

      The data set is stored in an array each element represented as a tuple (x, y).

      For i = 0 to n-1

      Begin

            Euclidian distance from each point to U is calculated.

            S is the set of m smallest distances (all the points related to distances must be classified)

      end

      Return   (Mode (m labels))

Random forest: Random forests are ensembles, which means combination of two or more models to get better results. Random forests create a forest of decision trees on data samples. Each decision tree gives a target class as output and the final target class is identified by performing some measures on outputs from each decision tree of corresponding input record.

Algorithm

1. Picks random samples from the provided dataset.
2. For each random sample, creates a decision tree.
3. Predicts the result from every decision tree.
4. Predicts target class of sample by simple voting as the final result.

Artificial neural network (ANN): ANN are made up of nodes that are connected to one other. The values of the property are stored in these nodes. The input values for ANN are collected from the training example's characteristics. The weighted values are subsequently delivered to the next set of nodes. This weighted total is subjected to a non-linear activation function, and the resulting value is transmitted to the next layer, where the process is repeated until the final output is reached. Figure 2 depicts this process.

By adjusting the weights in ANN, the anticipated output value gets closer to the observed value. The back-propagation algorithm is the most often used method for altering weights. The complete training set is applied repeatedly due to the nature of ANN learning, where each application is unique.

Validation: Evaluation metrics were used to verify how well the model fits to our data. There are many metrics to evaluate the model; some of the measures are accuracy, precision, recall, F1 score.

Precision: Precision is also known as positive predicted value which is a measure of accuracy. It is mathematically defined as true positives divided by the sum of true positives and false positives.

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \qquad (9)$$

Recall: Recall is also referred toas sensitivity, which is a measure of accuracy. It is mathematically defined as true positives divided by sum of true positives and false negatives.
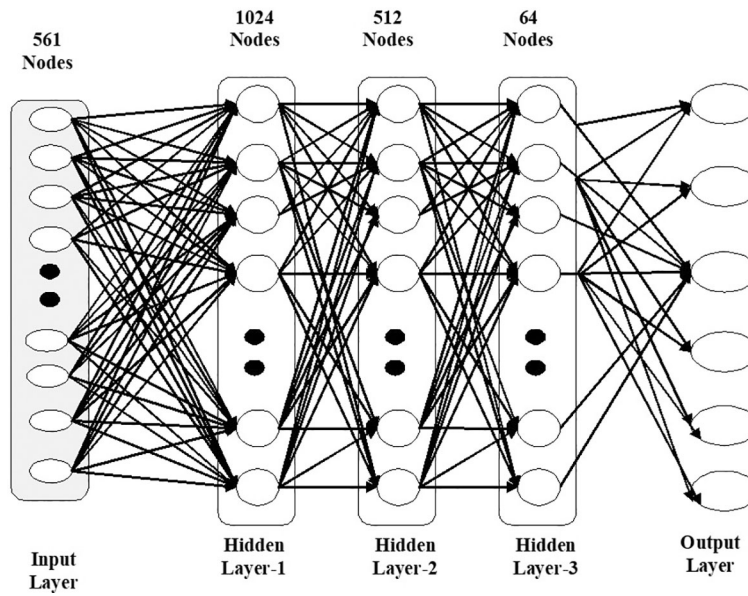
Figure 2. Neural network model structure.

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \tag{10}$$

F1 score: The F1 score is calculated with equation 11

$$\text{F1 score} = 2 \times \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \tag{11}$$

Accuracy: Accuracy defines how well the model predicts the class. It is mathematically defined as the number of truly predicted classes divided by total number of classes.

$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{\text{True positives} + \text{True negatives} + \text{False positives} + \text{False negatives}} \tag{12}$$

Results and discussion

The Results section comprises two parts:

- Exploratory data analysis results

Exploratory data analysis means gaining insights from the data before applying any model to it. It can be done using mathematical functions and visualizations. The total number of rows and columns in a dataset are identified by using the 'shape' function. It is shown in Figure 3.

From this we know that the dataset contains 10299 instances and 562 attributes.
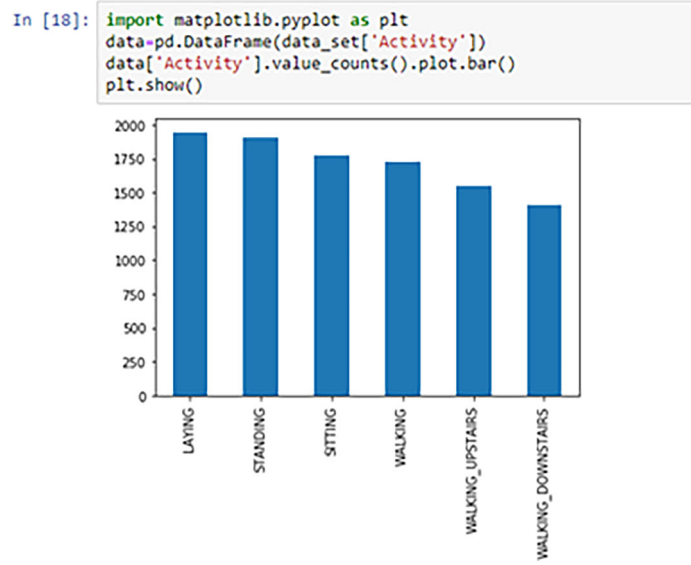
Out of 562 attributes 561 are independent attributes and one is dependent variable.

We used graphs to explore the data. We used a bar graph, box plot and heatmap.

```
In [17]: data_set.shape
Out[17]: (10299, 562)
```

Figure 3. Shape of the dataset.

```
In [18]: import matplotlib.pyplot as plt
         data=pd.DataFrame(data_set['Activity'])
         data['Activity'].value_counts().plot.bar()
         plt.show()
```
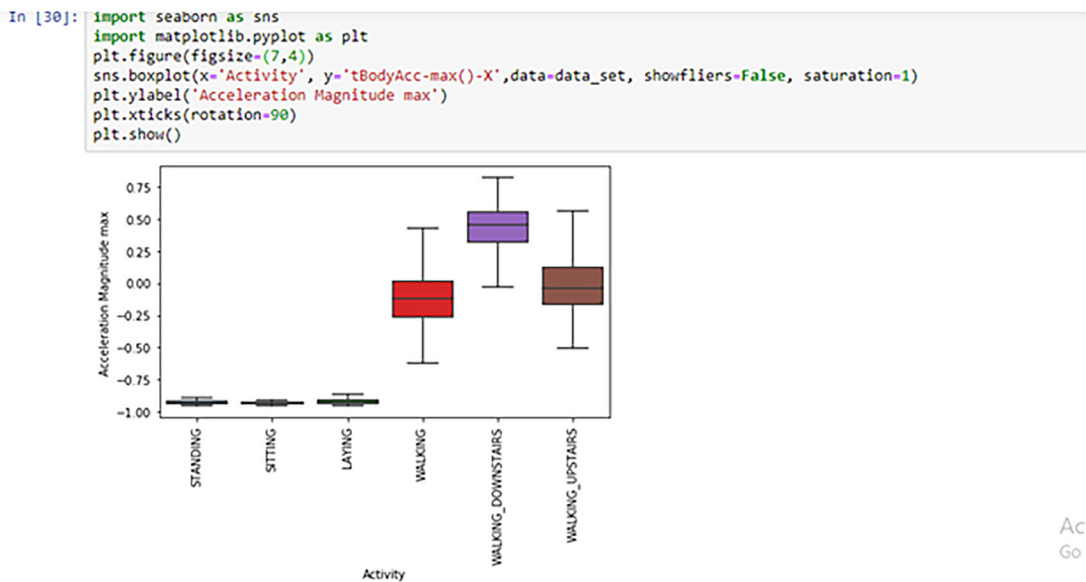

Figure 4. Bar graph for the attribute 'Activity'.

A bar graph shows the comparison between different target classes. The bar graph showing how many persons had the same classification is shown in Figure 4. The following observations were made from Figure 4: LAYING--1944, STANDING--1906, SITTING--1777, WALKING--1722, WALKING_UPSTAIRS--1544, WALKING_DOWNSTAIRS--1406.

Box plot: Box plots show the distribution of data by using five measures, namely minimum, first quartile(Q1), median, third quartile(Q3), and maximum. The box plot is drawn for the target class 'Activity' and the feature 'tBodyAcc-max()-X' in Figure 5.

From Figure 5, we made the following observations:

- If tBodyAcc-max()-X is less than -0.75 then activities are either Standing, Sitting or Laying.

```
In [30]: import seaborn as sns
         import matplotlib.pyplot as plt
         plt.figure(figsize=(7,4))
         sns.boxplot(x='Activity', y='tBodyAcc-max()-X',data=data_set, showfliers=False, saturation=1)
         plt.ylabel('Acceleration Magnitude max')
         plt.xticks(rotation=90)
         plt.show()
```


Figure 5. Box plot.

- If tBodyAcc-max()-X is greater than -0.50 then activities are classified as Walking or Walking_Downstairs or Walking_Upstairs.

- If tBodyAcc-max()-X is greater than 0.00 then activity is Walking_Downstairs.

Confusion matrix for naïve Bayes classifier is shown in Figure 6. From Figure 6, the following observations were made:

- The actual number of Laying was 389 but the model correctly predicted 361 as Laying and incorrectly predicted 22 as Sitting and nine as Walking_Upstairs.

- The actual number of Sitting was 372 but the model correctly predicted 336 as Sitting and incorrectly predicted two as Laying, 31 as Standing and three as Walking_Upstairs.

- The actual number of Standing was 375 but the model correctly predicted 157 as Standing, and incorrectly predicted 214 as Sitting and four as Walking_Upstairs.

- Th actual number of Walking was 345 but model correctly predicted 256 as Walking and incorrectly predicted 31 as Walking_Downstairs and 58 as Walking_Upstairs.

- The actual number of Walking_Downstairs was 282 but the model correctly predicted 199 of them as Walking_Downstairs and incorrectly predicted 25 as Walking and 58 as Walking_Upstairs.

- The actual number of Walking_Upstairs was 297 but the model correctly predicted 275 of them as Walking_ Upstairs and incorrectly predicted 17 as Walking_Downstairs and five as Walking.

Precision, recall and F1 score values of naïve Bayes classifier are depicted in the bar plot shown in Figure 7. From Figure 7, we can conclude that the class Laying was most correctly predicted among all other classes. Among all the classes, Standing had a low number of correctly predicted instances.

Confusion matrix for the decision tree classifier is shown in Figure 8. From Figure 8, the following observations were made:

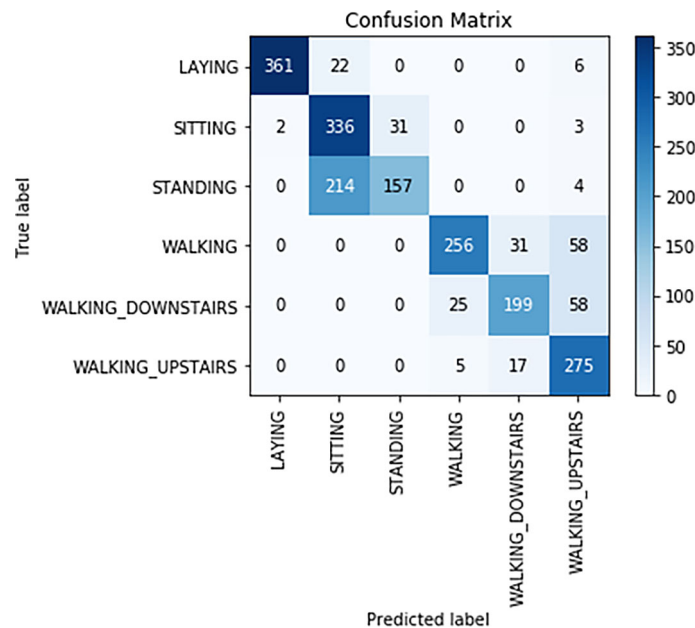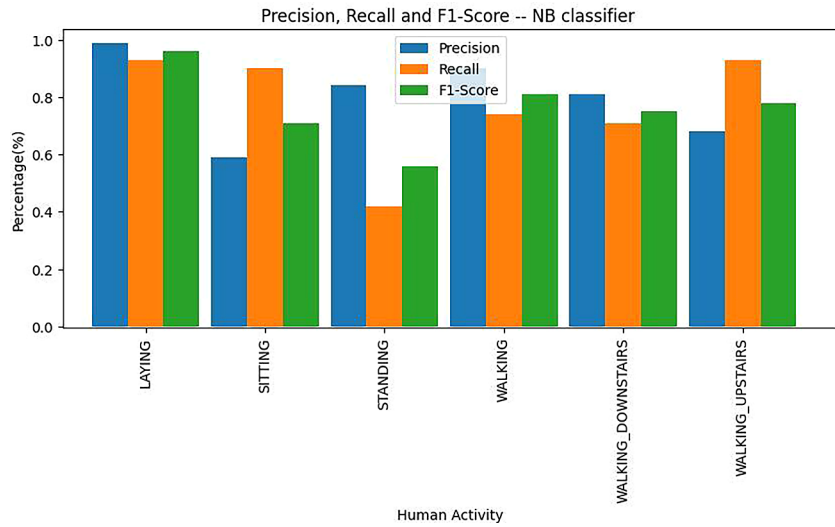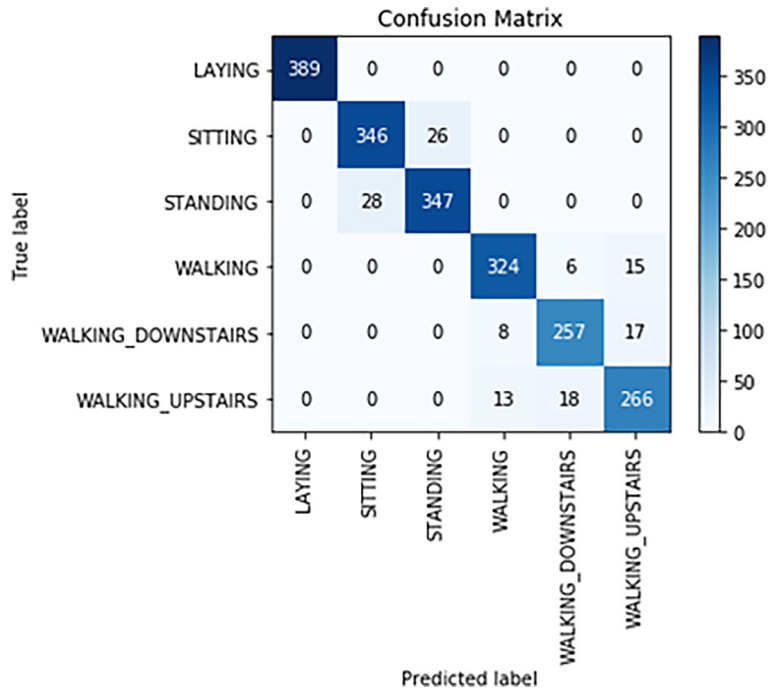- The actual number of Laying was 389 and the model correctly predicted 389 as Laying.

**Figure 6. Confusion matrix for naïve Bayes classification.**

Figure 7. Precision, recall and F1-Score for naïve Bayesclassifier.

Figure 8. Confusion matrix for decision tree.

- The actual number of Sitting was 372 but the model correctly predicted346 of them as Sitting and incorrectly predicted 26 as Standing.

- The actual number of Standing was 375 but the model correctly predicted 347 of them as Standing and incorrectly predicted 28 as Sitting.

- The actual number of Walkingwas 345 but the model correctly predicted324 as Walking and incorrectly predicted six of them as Walking_Downstairs and 15 as Walking_Upstairs.

- The actual number of Walking_Downstairs was 282, but the model correctly predicted 257 of them as Walking_Downstairs and incorrectly predicted eight as Walking and 15 as Walking_Upstairs.

- The actual number of Walking_Upstairs was 297, but the model correctly predicted 266 as Walking_Upstairs and incorrectly predicted13 as Walking and 18 as Walking_Downstairs.
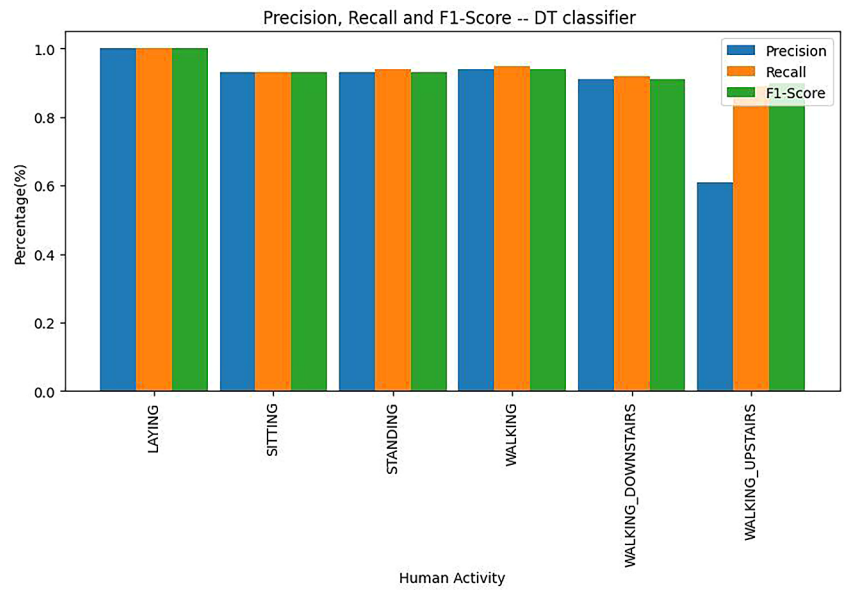
Figure 9. Precision, recall and F1 score for decision tree classifier.

Precision, recall and F1 score values of decision tree classifier are depicted in the bar plot shown in Figure 9. From Figure 9 we can see that the class Laying was predicted correctly without any incorrectly classified classes.

Confusion matrix for K-nearest neighbours classifier is shown in Figure 10. From Figure 10, the following observations were made:

- The actual number of Laying was 389 and the model correctly predicted 389 as Laying.

- The actual number of Sitting was 372, but the model correctly predicted 332 of them as Sitting and incorrectly predicted 239 as Standing and one as Sitting.
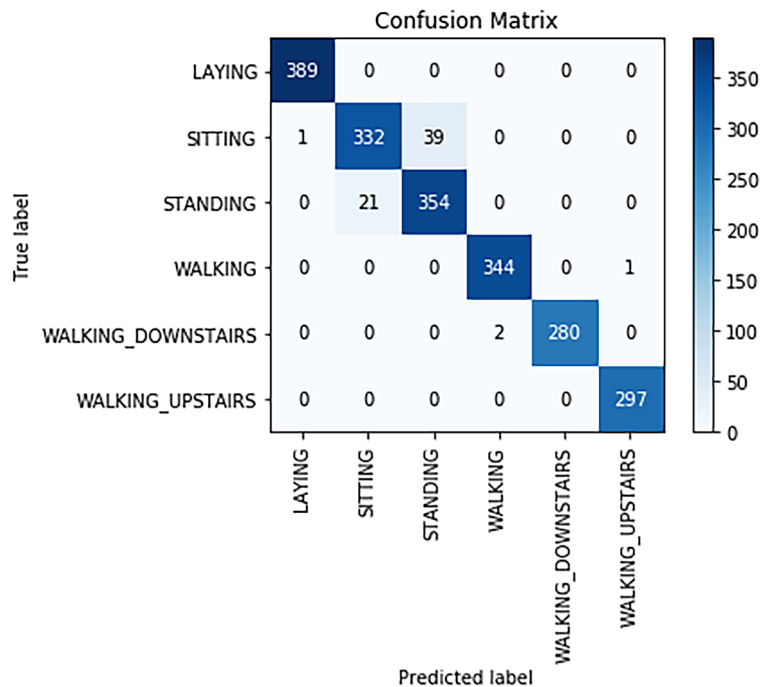
Figure 10. Confusion matrix for K-nearest neighbour.

- The actual number of Standing was 375, but the model correctly predicted 354 as Standing and incorrectly predicted 21 as Sitting.

- The actual number of Walking was 345, but the model correctly predicted 344 of them as Walking and incorrectly predicted one as Walking_Upstairs.

- The actual number of Walking_Downstairs was 282, but the model correctly predicted 280 of them as Walking_Downstairs and incorrectly predicted two as Walking.

- The actual number of Walking_Upstairs was 297, and the model correctly predicted all 297 as Walking_Upstairs.

Precision, recall and F1 score values of the K-nearest neighbour classifier are depicted in the bar plot shown in Figure 11. From Figure 11, we can see that the activity Laying was predicted correctly without any incorrectly classified classes; hence, all metrics had a 100% score.

Confusion matrix for the random forest classifier is shown in Figure 12. From Figure 12, the following observations were made:

- The actual number of Laying was 389 and the model correctly predicted 389 as Laying.

- The actual number of Sitting was 372, but the model correctly predicted 355 as Sitting and incorrectly predicted 17 as Standing.

- The actual number of Standing was 375, but the model correctly predicted 355 of them as Standing and incorrectly predicted 20 as Sitting.

- The actual number of Walking was 345, but the model correctly predicted 336 as Walking and incorrectly predicted two as Walking_Downstairs and seven as Walking_Upstairs.

- The actual number of Walking_Downstairs was 282, but the model correctly predicted 265 of them as Walking_Downstairs and incorrectly predicted 12 as Walking and five as Walking_Upstairs.

- The actual number of Walking_Upstairs was 297, but the model correctly predicted 286 as Walking_Upstairs and incorrectly predicted seven as Walking and four as Walking_Downstairs.

Precision, recall and F1 score values of random forest are depicted as a bar plot shown in Figure 13. From Figure 13 we can see that all instances having Laying as target were correctly predicted.
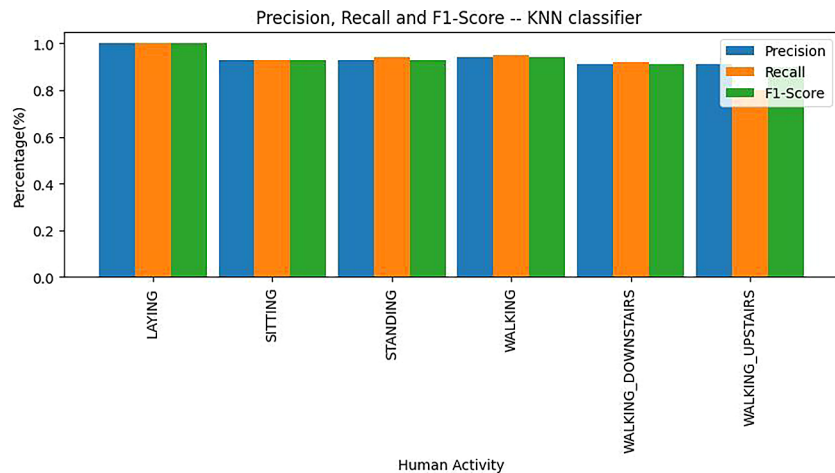

Figure 11. Precision, recall and F1 score for K-nearest neighbour classifier.
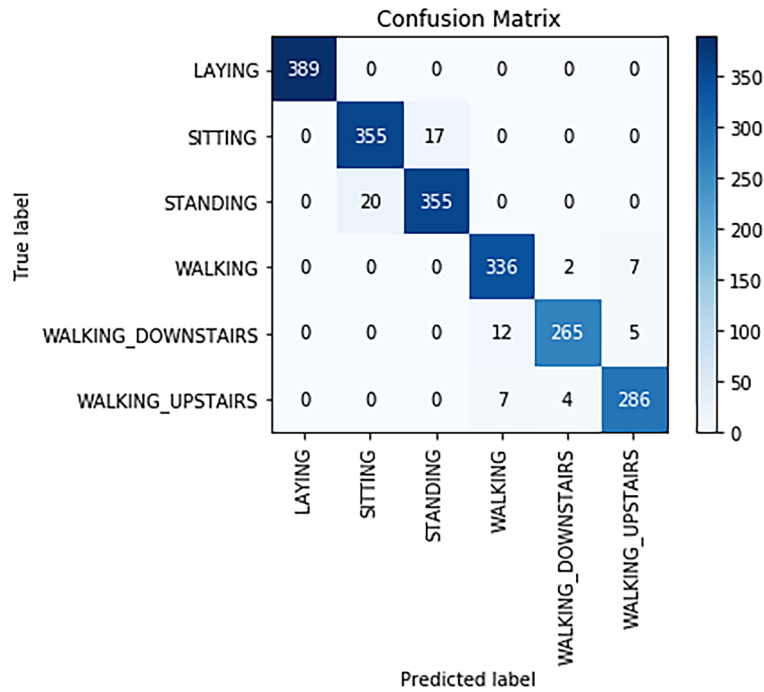
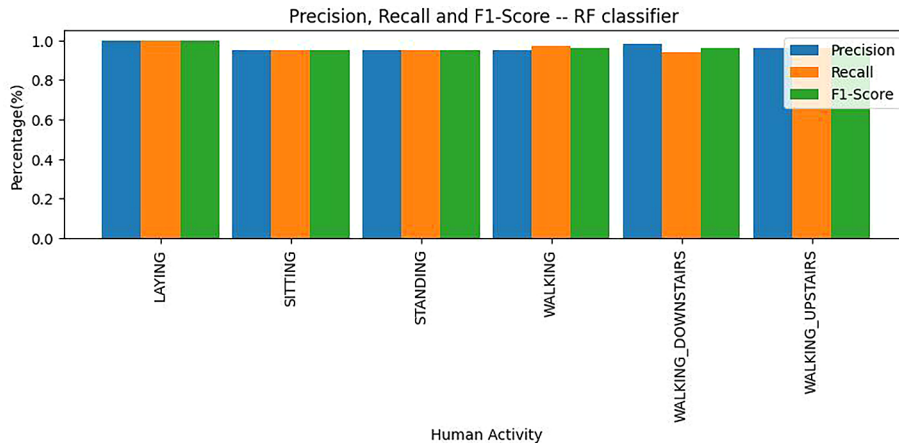**Figure 12. Confusion matrix for random forest.**

Figure 13. Precision, recall and F1 score for random forest classifier.

Confusion matrix for logistic regression is shown in Figure 14. From Figure 14, the following observations were made:

- The actual number of Laying was 389 and the model correctly predicted 389 as Laying.

- The actual number of Sitting was 372, but the model correctly predicts 351 of them as Sitting and incorrectly predicts 21 as Standing.

- The actual number of Standing was 375 but the model correctly predicted 360 of them as Standing and incorrectly predicted 15 as Sitting.

- The actual number of Walkingwas 345, but the model correctly predicted 344 of them as Walking and incorrectly predicted one as Walking_Upstairs.
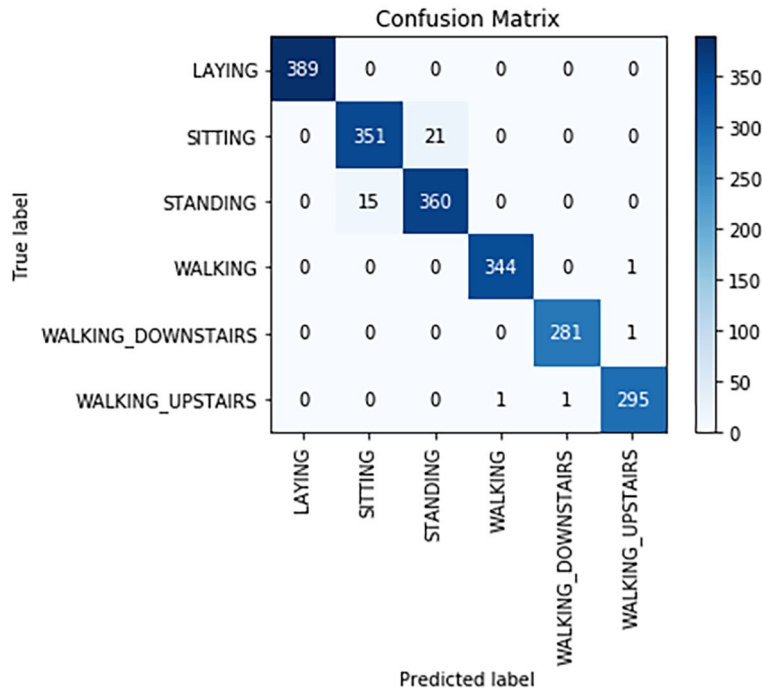
Figure 14. Confusion matrix for logistic regression.

- The actual number of Walking_Downstairs was 282, but the model correctly predicted 281 as Walking_Downstairs and incorrectly predicts one as Walking_Upstairs.

- The actual number of Walking_Upstairs was 297 but the model correctly predicted295 of them as Walking_Upstairs and incorrectly predicted one as Walking and one as Walking_Downstairs.

Precision, recall and F1 score values of logistic regression are depicted as a bar plot shown in Figure 15. From Figure 15, we can see that Laying, Walking, Walking_Downstairs instances were correctly classified.

Neural network: Epoch versus training loss for the human activity dataset is represented in Figure 16. From Figure 16, the testing loss decreased and hence the accuracy increased. Of all the algorithms we applied, neural networks predicted more accurately, with an accuracy of 98.93%
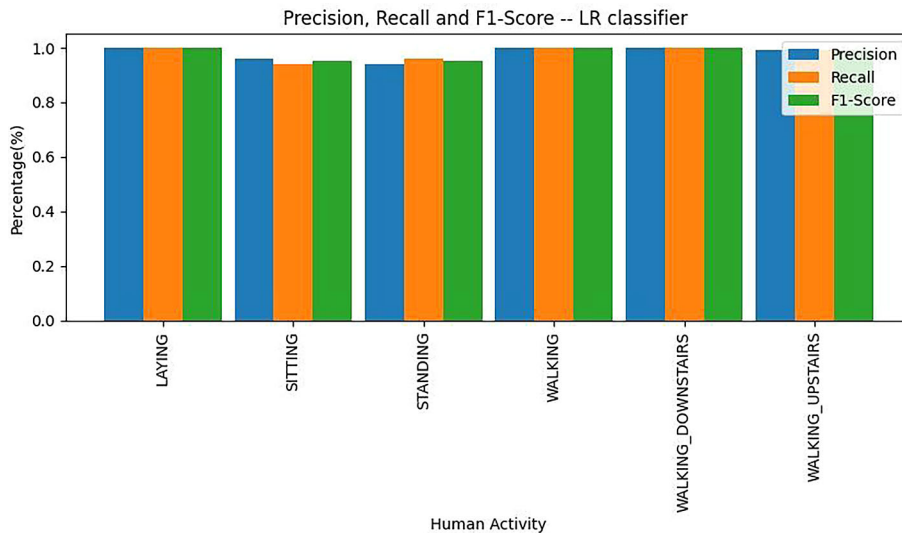
Figure 15. Precision, recall and F1 score for logistic regression classifier.
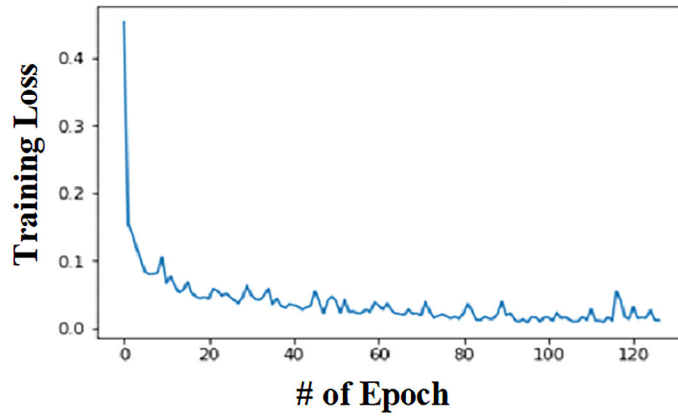
Figure 16. **Loss graph for neural network.**
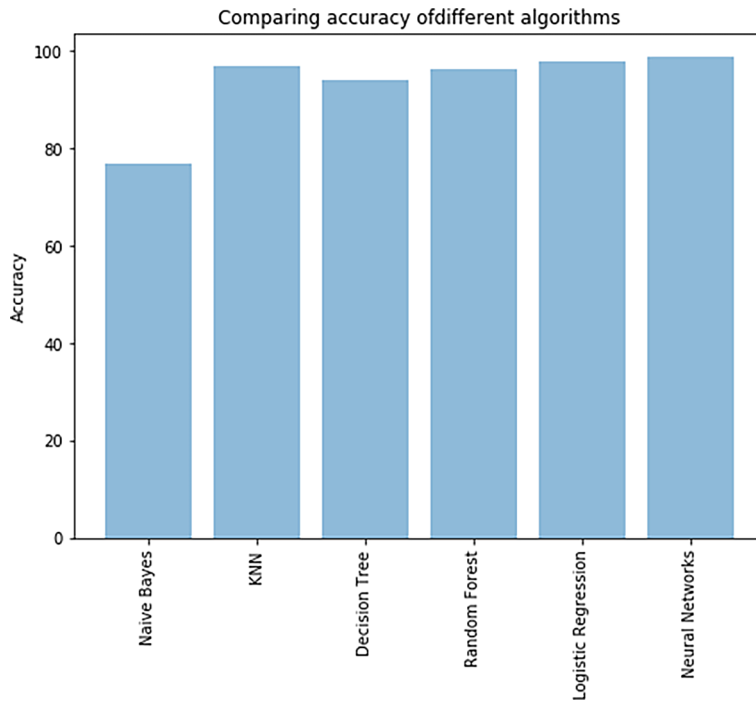
Figure 17. **Comparison of algorithms.**

Table 2. **Accuracy of naïve Bayes, decision tree, K-nearest neighbour, random forest, logistic regression, neural network.**

Algorithm	Accuracy
Naïve Bayes	76.89
K-nearest neighbours	96.40
Decision tree	93.39
Random forest	96.89
Logistic regression	98.05
Neural networks	98.93

Comparison of different algorithms: Different algorithms used to train the model were considered as variable x and their corresponding accuracy as variable y. The bar graph drawn for these x and y variables is shown in Figure 17. The accuracy results are shown in Table 2. From Figure 17, when comparing all algorithms, neural network predicted the data with the highest accuracy, *i.e.*98.93%.

Conclusions

For this work, human activity recognition was used is to detect the activity of a person. The data set was collected by sensors from triaxial accelerometer and gyroscope. The data set consisted of 561 features and 10,299 records, and six classes, namely Sitting, Standing, Laying, Walking, Walking_Downstairs, Walking_Upstairs. In this paper we used six algorithms, namely naïve Bayes, decision tree, random forest, K-nearest neighbours and neural network. For neural networks, the input layer consisted of 561 nodes and three hidden layers (1024,512,64);output layers consisted of six nodes, each representing a target class. From the experimental results, naïve Bayes classifier achieved 76.89% accuracy, the decision tree classifier achieved 93.39%, random forest classifier achieved 96.89%; K-nearest neighbours achieved 96.40%;logistic regression classifier achieved 98.05%. Among all these models,the neural networks model predicted the target class with an accuracy of 98.93%.

Data availability
Underlying data
Kaggle: Human Activity Recognition with Smartphones, https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones

Data are available under the terms of the Creative Commons Zero "No rights reserved" data waiver (CC0 1.0 Public domain dedication).

Extended data
Analysis code

Source code available from: https://github.com/someshchinta/Human_Actiity_recognition

Archived source code at time of publication: https://doi.org/10.5281/zenodo.7108706

License: Apache-2.0

References

1. Schüldt IL, Caputo B: **Recognizing human actions: a local SVM approach.** *Pattern Proceedings of the 17th International Conference on Pattern Recognition.* 2004; vol. 23: pp. 32–36. Cambridge, UK.

2. Laptev I, Marszalek M, Schmid C, *et al.*: **Learning realistic human actions from movies.** *2008 IEEE Conference on Computer Vision and Pattern Recognition.* 2008; vol. 4: pp. 1–8. Anchorage, AK, USA.

3. Yamato J, Ohya J, Ishii K: **Recognizing human action in time-sequential images using hidden markov model.** *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 1992; vol. 1992: pp. 379–385, Champaign, IL, USA.

4. Oliver NM, Rosario B, Pentland AP: **A Bayesian computer vision system for modeling human interactions.** *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2000; vol. 22: pp. 831–843,

5. Suk HI, Sin BK, Lee SW: **Hand gesture recognition based on dynamic Bayesian network framework.** *Pattern Recogn.* 2000; **43**: 3059–3072.

6. Natarajan P, Nevatia R: **View and scale invariant action recognition using multiview shape-flow models.** *2008 IEEE Conference on Computer Vision and Pattern Recognition.* 2008; pp. 1–8. Anchorage, AK, USA.

7. Ning H, Xu W, Gong Y, Huang T: **Latent pose estimator for continuous action recognition.** *Computer Vision--European Conference on Computer Vision 2008.* Marseille, France: Springer; 2008; vol. 43: pp. 419–433.

8. Vail DL, Veloso MM, Lafferty JD: **Conditional random fields for activity recognition.** *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems.* 2007; vol. 1: pp. 235.

9. Kiros R, Zhu Y, Salakhutdinov RR, *et al.*: **Skip-thought vectors.** *Adv. Neural Inf. Proces. Syst.* 2015; **1**: 3294–3302.

10. Grushin A, Monner DD, Reggia JA, *et al.*: **Robust human action recognition via long short-term memory.** *The 2013 International Joint Conference on Neural Networks (IJCNN).* 2013; vol. 25: pp. 1–8.

11. Veeriah V, Zhuang N, Qi GJ: **Differential recurrent neural networks for action recognition.** *2015 IEEE International Conference on Computer Vision (ICCV).* 2015; vol. 4: pp. 4041–4049.

12. Du Y, Wang W, Wang L: **Hierarchical recurrent neural network for skeleton based action recognition.** *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2015; vol. 23: pp. 1110–1118.

13. Leung KM: *Naive bayesian classifier.* Polytechnic University Department of Computer Science/Finance and Risk Engineering; 2007; pp. 123–156.

14. Peterson LE: **K-nearest neighbor.** *Scholarpedia.* 2009; **4**(2): 1883. **Publisher Full Text**

15. Swain PH, Hauska H: **The decision tree classifier: Design and potential.** *IEEE Trans. Geosci. Electron.* 1977; **15**(3): 142–147. **Publisher Full Text**

16. Liaw A, Wiener M: **Classification and regression by randomForest.** *R news.* 2002; **2**(3): 18–22.

17. Domínguez-Almendros S, Benítez-Parejo N, Gonzalez-Ramirez AR: **Logistic regression models.** *Allergol. Immunopathol.* 2011; **39**(5): 295–305. **Publisher Full Text**

18. Féraud R, Clérot F: **A methodology to explain neural network classification.** *Neural Netw.* 2002; **15**(2): 237–246. **Publisher Full Text**

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias

- You can publish traditional articles, null/negative results, case reports, data notes and more

- The peer review process is transparent and collaborative

- Your article is indexed in PubMed after passing peer review

- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research