# HUMAN ACTIVITY RECOGNITION USING SMART PHONE

**A Project Report**

Submitted to the Faculty of Engineering of

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA**

In partial fulfillment of the requirements for the award of the Degree of

**BACHELOR OF TECHNOLOGY**

IN

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**


By

**K.Sai Kumar** **(20481A5426)**

**K.Raghavendra** **(20481A5428)**

**T.Anirudh Kumar** **(20481A5454)**

**D.Hemanth** **(20481A5412)**

Under the Supervision of

**Mr. K. Ashok Reddy** **M.Tech,( PhD)**

**Assistant Professor**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE**

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

**SESHADRIRAO KNOWLEDGE VILLAGE**

**GUDLAVALLERU – 521 356**

**ANDHRA PRADESH**

**2020-2024**

**SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE**

**(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)**

SHESHADRI RAO KNOWLEDGE VILLAGE

GUDLAVALLERU-521356

**DEPARTMENT**

**OF**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

# CERTIFICATE

This is to certify that the project report entitled "**HUMAN ACTIVITY RECOGNITION USING SMART PHONE**" is a bonafide record of work carried out by **K.Sai Kumar (20481A5426) , K.Raghavendra (20481A5428), T.Anirudh Kumar (20481A5454), D.Hemanth (20481A5412)** under the guidance and supervision of **Mr. K. Ashok Reddy** in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in **Artificial Intelligence and Data Science** of **Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru.**

**Supervisor** **Head of the Department**

**Mr. K. Ashok Reddy** **Dr. K. Srinivas**

# Acknowledgements

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragements crown all the efforts with success.

We would like to express our deep sense of gratitude and sincere thanks to **Mr. K. Ashok Reddy** M. Tech, (PhD) Assistant professor, Department of Artificial Intelligence and Data Science for his constant guidance, supervision and motivation in completing the project work.

We feel elated to express our floral gratitude and sincere thanks to **Dr. K. Srinivas,** Head of the Department, Artificial Intelligence and Data Science, for his encouragement all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the project work.

We would like to thank our beloved principal **Dr. G. V. S. N. R. V. Prasad** for providing great support for us in completing our project and givingus the opportunity to do the project.

Our Special thanks to the faculty of our department and programmers of our computer lab. Finally, we thank our family members, non-teaching staff and our friends, who have directly or indirectly helped and supported us in completing our project in time.

**Team members**

K.Sai Kumar (20481A5426)
K.Raghavendra (20481A5428)
T.Anirudh Kumar (20481A5454)
D.Hemanth (20481A5412)

# ABSTRACT

Human Activity Recognition (HAR) is an important research area that aims to develop algorithms for recognizing human activities based on sensor data from wearable devices such as smartphones. In this project, we developed a machine learning model to recognize six activitiesof daily living (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING,LAYING) using sensor signals from a waist-mounted smartphone. The dataset was randomly partitioned into training and test sets, and features were extracted from the time and frequency domains of the signals. We used a Support Vector Machine (SVM) algorithm for classification and achieved an accuracy of 99.3% on the test data. Our model can have various applications in fields such as healthcare, fitness tracking, and smart homes, and can help to improve the quality of life for individuals by enabling activity monitoring and feedback. Future work can explore the use of additional sensors and more complex algorithms for activity recognition, as well as the integration of HAR into mobile apps and other wearable devices.

# Table of Contents


| SI. No | Topic | Page No |
|---|---|---|
| 1. | INTRODUCTION | 1 |
| | 1.1 Introduction | 1 |
| | 1.2 Objectives | 1 |
| | 1.3 Problem Statement | 2 |
| 2 | LITERATURE REVIEW | 3 |
| 3 | METHODOLOGY | 4 |
| 4 | IMPLEMENTATION | 17 |
| 5 | RESULTS AND DISCUSSION | 25 |
| 6 | CONCLUSIONS | 29 |
| 7 | REFERENCES | 30 |

# LIST OF IMAGES


| Fig No | Figure Name | Page no |
|---|---|---|
| 1 | General Methodology of Machine Learning Models | 5 |
| 2 | KNN Algorithm | 8 |
| 3 | Decision Tree Algorithm | 9 |
| 4 | Random Forest Algorithm | 10 |
| 5 | SVM Algorithm | 11 |
| 6 | Artificial Neural Networks Algorithm | 13 |
| 7 | Data Description | 14 |
| 8 | Correlation Matrix | 15 |
| 9 | Pair plots | 16 |
| 10 | Decision Tree Example | 19 |
| 11 | Accuracy Graph | 25 |
| 12 | Different Methods of Neural Networks Error Graphs | 26 |
| 13 | SVM confusion Matrix | 27 |
| 14 | ANN confusion Matrix | 27 |

# 1.INTRODUCTION

## 1.1 Introduction

Human Activity Recognition (HAR) is an important research area that aims to develop algorithms for recognizing human activities based on sensor data from wearable devices such as smartphones. HAR has various applications in fields such as healthcare, fitness tracking, and smart homes, and can help to improve the quality of life for individuals by enabling activity monitoring and feedback.

Smartphones have become a popular platform for HAR due to their ubiquitous nature and embedded sensors such as accelerometers, gyroscopes, and magnetometers. These sensors can capture various aspects of human motion, such as linear acceleration, angular velocity, and orientation, and enable the recognition of activities of daily living (ADL) such as walking, running, sitting, standing, and sleeping. HAR using smartphones can be a non-intrusive and cost-effective way to monitor and track activity levels in individuals, and can provide valuable insights into their health and wellbeing.

The process of HAR using smartphones typically involves collecting sensor data fromthe smartphone, preprocessing the data to remove noise and artifacts, extracting features from the data using signal processing techniques, and developing machine learning models to classify the activities based on the extracted features. Various machine learning algorithms such as decision trees, support vector machines (SVMs), and deep neural networks (DNNs) have been used for HAR, and their performance depends on factors such as the type of sensor data, the feature extraction techniques, and the choice of algorithm.

Overall, HAR using smartphones has the potential to revolutionize healthcare and wellness by enabling continuous monitoring of activity levels and providing personalized feedback and interventions. As smartphone technology continues to advance and sensors become more accurate and diverse, the potential applications of HAR using smartphones are only expected to grow.

## 1.2 Objective

The objective of the Human Activity Recognition (HAR) project using smartphone data is to develop a machine learning model that can accurately classify activities of daily living (ADL) performed by individuals based on sensor data collected from a smartphone. Specifically, the project aims to classify six different activities - walking, walking upstairs, walking downstairs, sitting, standing, and laying - using data collected from the smartphone's embedded accelerometer and gyroscope.

## 1.3 Problem Statement

The Human Activity Recognition (HAR) project using smartphone data is to develop a machine learning model that can accurately classify activities of daily living (ADL) performed by individuals based on sensor data collected from a smartphone. Accurately recognizing and classifying ADLs using smartphone data is a challenging task due to factors such as sensor noise, variations in human motion, and the complexity of the feature space.

## 2. LITERATURE REVIEW

The demands for understanding human activities have grown in health-care domain, especially in elder care support, rehabilitation assistance, diabetes, and cognitive disorders. [1,2,3]. A huge amount of resources can be saved if sensors can help caretakers record and monitor the patients all the time and report automatically when any abnormal behavior is detected. Other applications such as human survey system and location indicator are all benefited from the study. Many studies have successfully identified activities using wearable sensors with very low error rate, but the majority of the previous works are done in the laboratories with very constrained settings. Readings from multiple body-attached sensors achieve low error-rate, but the complicated setting is not feasible in practice. This project uses low-cost and commercially available smartphones as sensors to identify human activities. The growing popularity and computational power of smartphone make it an ideal candidate for non-intrusive body-attached sensors. According to the statistic of US mobile subscribers, around 44%of mobile subscribers in 2011 own smartphones and 96% of these smartphones have built-in inertial sensors such as accelerometer or gyroscope [4,5]. Research has shown that gyroscope can help activity recognition even though its contribution alone is not as Human activity recognition has been studied for years and researchers have proposed different solutions to attack the problem. Existing approaches typically use vision sensor, inertial sensor and the mixture of both. Machine learning and threshold-base algorithms are often applied. Machine learning usually produces more accurate and reliable results, while threshold-based algorithms are faster and simpler. One or multiple cameras have been used to capture and identify body posture [8, 9]. Multiple accelerometers and gyroscopes attached to different body positions are the most common solutions [10-13]. Approaches that combine both vision and inertial sensors have also been purposed [14]. This technique, however, has yet been applied on the human activity problem before

# 3.METHODOLOGY

## METHODOLOGY and FLOWCHARTS

We are following the steps listed below to implement the machine learning algorithms on the data:

**1.Collecting Data:**

As you know, machines initially learn from the data that you give them. It is of the utmost importance to collect reliable data so that your machine learning model can find the correct patterns. The quality of the data that you feed to the machine will determine how accurate your model is. If you have incorrect or outdated data, you will have wrong outcomes or predictions which are not relevant.

**2.Preparing the Data:**

After you have your data, you have to prepare it. You can do this by :

- Putting together all the data you have and randomizing it. This helps make sure that data is evenly distributed, and the ordering does not affect the learning process.
- Cleaning the data to remove unwanted data, missing values, rows, and columns, duplicate values, data type conversion, etc. You might even have to restructure the dataset and change the rows and columns or index of rows and columns.
- Visualize the data to understand how it is structured and understand the relationship between various variables and classes present.
- Splitting the cleaned data into two sets - a training set and a testing set. The training set is the set your model learns from. A testing set is used to check the accuracy of your model after training.

**3.Choosing a Model:**

A machine learning model determines the output you get after running a machine learning algorithm on the collected data. It is important to choose a model which is relevant to the task at hand. Over the years, scientists and engineers developed various models suited for different tasks like speech recognition, image recognition, prediction, etc. Apart from this, you also have to see if your model is suited for numerical or categorical data and choose accordingly. But here we have a problem of classification so we will be working on classification algorithms like KNN, SVM, Random Forest etc...

**4.Training the Model:**

Training is the most important step in machine learning. In training, you pass the prepareddata to your machine learning model to find patterns and make predictions. It results in themodel learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.

**5.Evaluating the Model:**

After training your model, you have to check to see how it's performing. This is done bytesting the performance of the model on previously unseen data. The unseen data used is the testing set that you split our data into earlier. If testing was done on the same data which is used for training, you will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy.

When used on testing data, you get an accurate measure of how your model will performand its speed.

**6.Parameter Tuning:**

Once you have created and evaluated your model, see if its accuracy can be improved in any way. This is done by tuning the parameters present in your model. Parameters are the variables in the model that the programmer generally decides. At a particular value of your parameter, the accuracy will be the maximum. Parameter tuning refers to finding these values.

**7.Making Predictions**

In the end, you can use your model on unseen data to make predictions accurately.

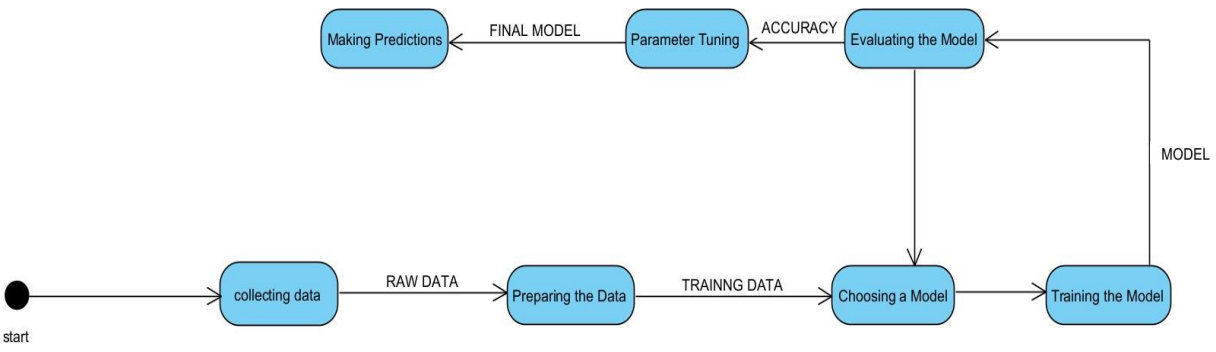


Fig 1:General Methodology of Machine Learning Models

## Collecting data:

We have chosen to work with human activity recognition so we have taken the dataset from kaggle

**Link:**

https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones/download?datasetVersionNumber=2

The Human Activity Recognition database was built from the recordings of 30 study participants performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors. The objective is to classify activities inti one of the six activities performed.

**Preparing the Data/Pre processing:**

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

Splitting dataset into train and test sets:

- Splitting 80 percent of data for training
- 20 percent of data for testing and evaluation of the model
- The random state hyperparameter in the **train_test_split()** function controls the shuffling process.

With `random_state=None` , we get different train and test sets across different executionsand the shuffling process is out of control.

With `random_state=0` , we get the same train and test sets across different executions.With `random_state=42`, we get the same train and test sets across different execu

## Choosing a Model:

Human activity recognition is a problem of classifying the users actions by the sensor reading we take form a smart phone. So, we will be working on various classification algorithms

**Classification:**

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc. Classes can be called as targets/labels or categories.

Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.

**y=f(x), where y = categorical outputTypes of classifiers:**

- **Binary Classifier:** If the classification problem has only two possible outcomes.

  **Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes.

  **Example:** Classifications of types of crops, Classification of types of music.

The problem we have chosen is **Multi-class Classification** problem

## Classification algorithms:

**1.K-Nearest Neighbor(KNN):**

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Algorithm:

**Step-1:** Select the number K of the neighbors

**Step-2:** Calculate the Euclidean distance of **K number of neighbors**

**Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.

**Step-4:** Among these k neighbors, count the number of the data points in eachcategory.

**Step-5:** Assign the new data points to that category for which the number of theneighbor is maximum.

**Step-6:** Our model is ready.

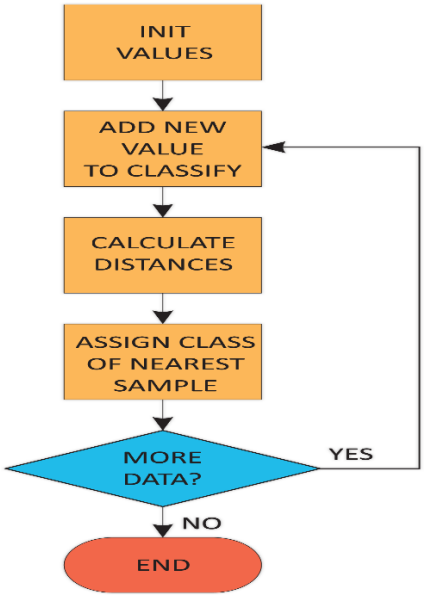


Fig 2:KNN Algorithm

**2.Decision Tree Classification:**

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf

Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

Algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3.

  Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

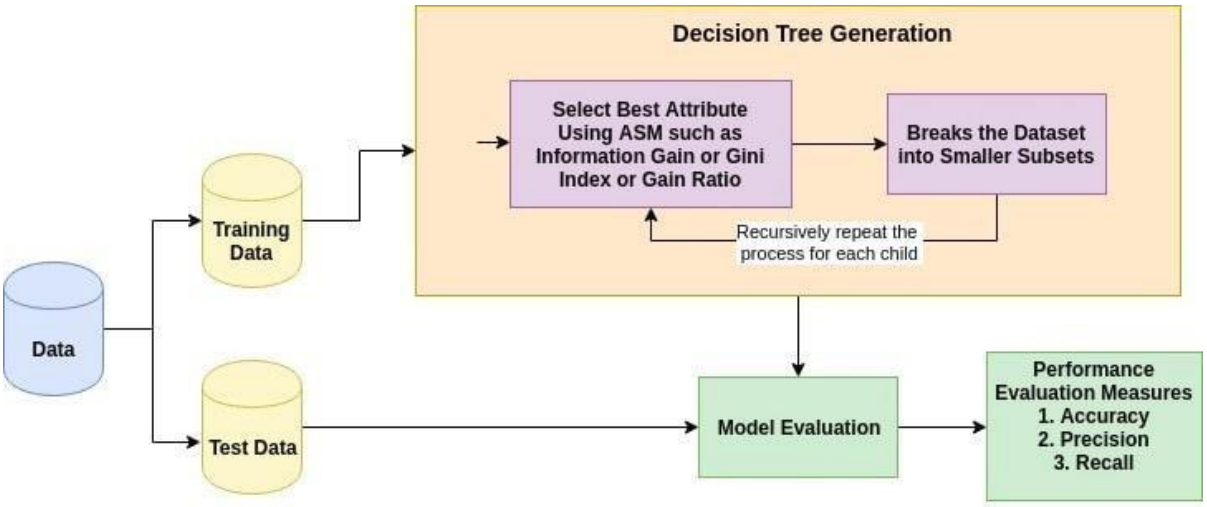


Fig 3:Decesion Tree Algorithm

### 3.Random Forest Classification:

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML.It is based on the concept of **ensemble learning,** which is a process of combining multipleclassifiers to solve a complex problem and to improve the performance of the model.
- As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votesof predictions, and it predicts the final output.
- **The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

**Algorithm:**

- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.
- **Step-5:** For new data points, find the predictions of each decision tree, and assign the newdata points to the category that wins the majority votes.

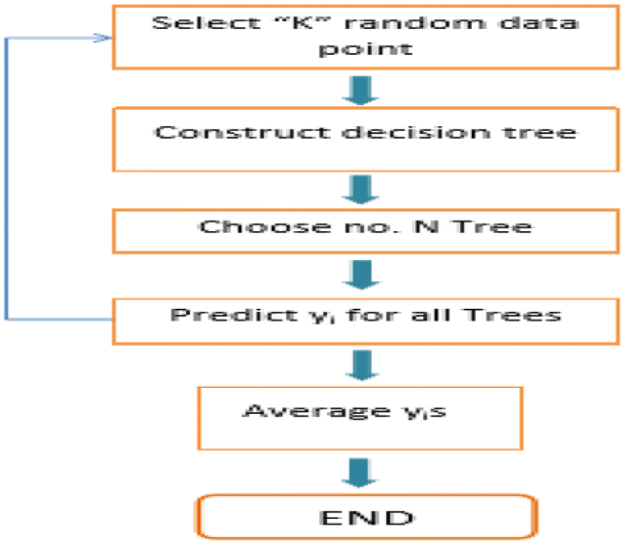


**Fig 4: Random Forest Algorithm**

**4.Support Vector Machine:**

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

**SVM can be of two types:**

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, thensuch data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non- linear SVM classifier.

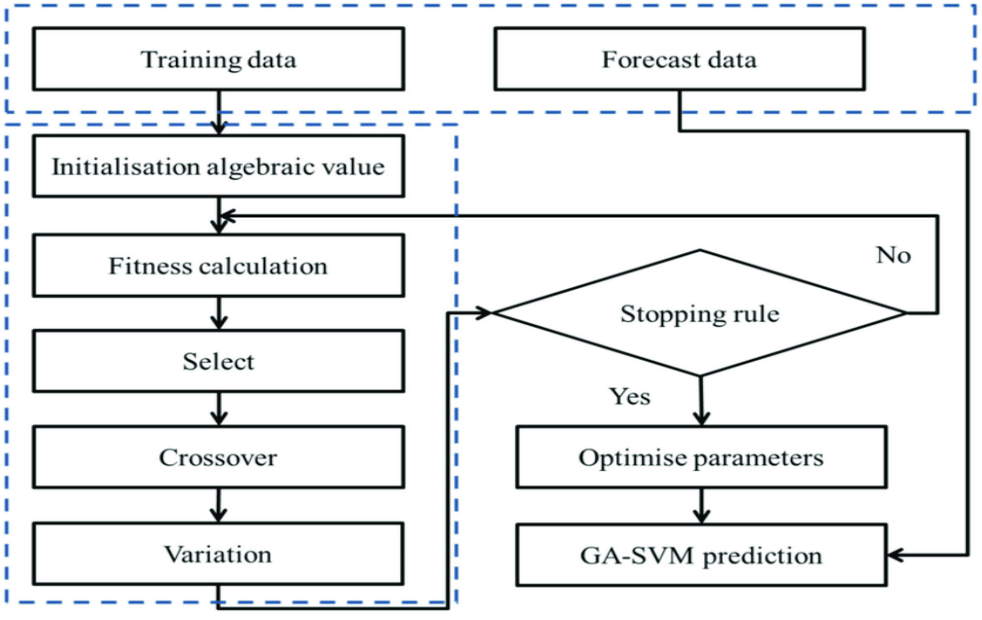


**Fig 5: SVM Algorithm**

## 5.Artificial Neural Networks(ANN):

- An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial
- neural network is designed by programming computers to behave simply like interconnectedbrain cells.
- There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.
- We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons inour brain, which are "learning."
- Parts of artificial neural networks:
  - **Input Layer:**

    As the name suggests, it accepts inputs in several different formats provided by the programmer.
  - **Hidden Layer:**

    The hidden layer presents in-between input and output layers. It performs all the calculationsto find hidden features and patterns.
  - **Output Layer:**

    The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

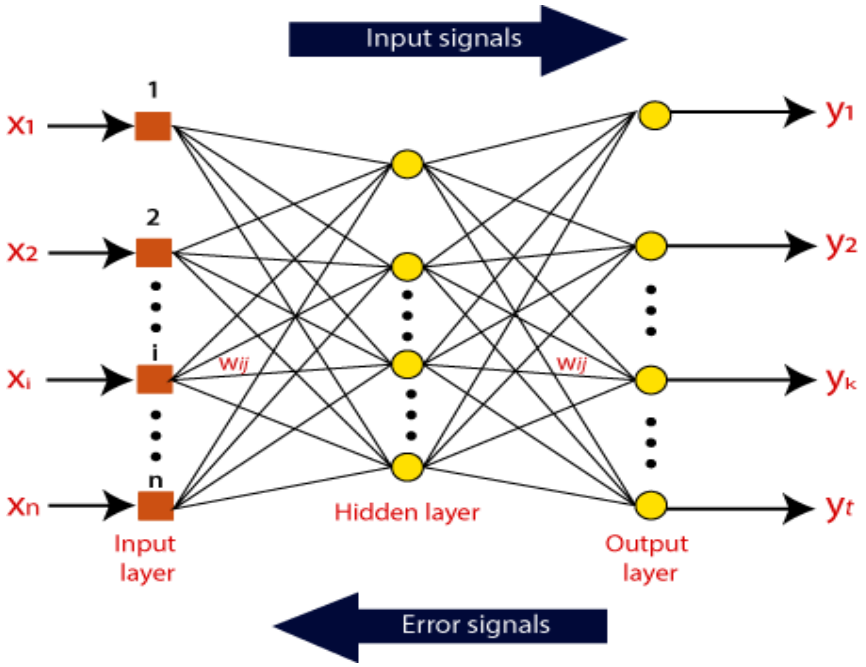


**Fig 6: Artificial Neural Networks Algorithm**

## Training:

The selected machine learning model is trained using the pre-processed and feature-extracted dataset. The model parameters are optimized to minimize the prediction error and improve the model's accuracy.

## Evaluation:

The trained machine learning model is evaluated using the test dataset, and its performance is measured in terms of accuracy, precision, recall, and F1 score. The model's robustness and generalization ability are also evaluated by testing it on new and unseen data.

**Data Representation**:

**Info and Describe:**

# Print a concise summary of a DataFrame.

# This method prints information about a DataFrame includingthe index dtype and columns, non-null values and memory usage.

**ds.info()**

# Generate descriptive statistics.

# Descriptive statistics include those that summarize the central

# tendency, dispersion and shape of adataset's distribution, excluding ``NaN`` values.

**ds.describe()**

**output:**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2947 entries, 0 to 2946

Columns: 563 entries, tBodyAcc-mean()-X to Activity

dtypes: float64(561), int64(1), object(1)

memory usage: 12.7+ MB

:

| | tBodyAcc-mean()-X | tBodyAcc-mean()-Y | tBodyAcc-mean()-Z | tBodyAcc-std()-X | tBodyAcc-std()-Y | tBodyAcc-std()-Z | tBodyAcc-mad()-X | tBodyAcc-mad()-Y | tBodyAcc-mad()-Z | tBodyAcc-max()-X | ... | fBodyBody( |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 2947.000000 | 2947.000000 | 2947.000000 | 2947.000000 | 2947.000000 | 2947.000000 | 2947.000000 | 2947.000000 | 2947.000000 | 2947.000000 | ... | |
| mean | 0.273996 | -0.017863 | -0.108386 | -0.613635 | -0.508330 | -0.633797 | -0.641278 | -0.522676 | -0.637038 | -0.462063 | ... | |
| std | 0.060570 | 0.025745 | 0.042747 | 0.412597 | 0.494269 | 0.362699 | 0.385199 | 0.479899 | 0.357753 | 0.523916 | ... | |
| min | -0.592004 | -0.362884 | -0.576184 | -0.999606 | -1.000000 | -0.998955 | -0.999417 | -0.999914 | -0.998899 | -0.952357 | ... | |
| 25% | 0.262075 | -0.024961 | -0.121162 | -0.990914 | -0.973664 | -0.976122 | -0.992333 | -0.974131 | -0.975352 | -0.934447 | ... | |
| 50% | 0.277113 | -0.016967 | -0.108458 | -0.931214 | -0.790972 | -0.827534 | -0.937664 | -0.799907 | -0.817005 | -0.852659 | ... | |
| 75% | 0.288097 | -0.010143 | -0.097123 | -0.267395 | -0.105919 | -0.311432 | -0.321719 | -0.133488 | -0.322771 | -0.009965 | ... | |
| max | 0.671887 | 0.246106 | 0.494114 | 0.465299 | 1.000000 | 0.489703 | 0.439657 | 1.000000 | 0.427958 | 0.786436 | ... | |

8 rows × 562 columns

**Fig 7: Data Description**

**Correlation matrix:**

A correlation matrix is a table containing correlation coefficients between variables. Each cell in the table represents the correlation between two variables. Thevalue lies between -1 and 1. A correlation matrix is used to summarize data, as a diagnostic for advanced analyses and as an input into a more advanced analysis.

Implementation in python: #importing the pandas library

#it includes tools for reading various fileformats and has many data pre-processingtools

import pandas as pd

# by help of pandas library we can read csv fileformats as:

# syntax: pandas.read_excel(file_name) ==> returns dataframeds=pd.read_csv('test.csv')

# pandas provide implementation for correation matrix

# syntax: data_set.corr()==> returns a 2d array of correlation matrixds.corr()

Output:

| | tBodyAcc-mean()-X | tBodyAcc-mean()-Y | tBodyAcc-mean()-Z | tBodyAcc-std()-X | tBodyAcc-std()-Y | tBodyAcc-std()-Z | tBodyAcc-mad()-X | tBodyAcc-mad()-Y | tBodyAcc-mad()-Z | tBodyAcc-max()-X |
|---|---|---|---|---|---|---|---|---|---|---|
| tBodyAcc-mean()-X | 1.000000 | 0.041274 | -0.129645 | 0.016984 | -0.001799 | -0.008065 | 0.022942 | -0.002124 | -0.010085 | 0.055897 |
| tBodyAcc-mean()-Y | 0.041274 | 1.000000 | 0.225980 | -0.054264 | -0.059066 | -0.077051 | -0.052501 | -0.062012 | -0.081834 | -0.047736 |
| tBodyAcc-mean()-Z | -0.129645 | 0.225980 | 1.000000 | -0.038578 | -0.048340 | -0.042342 | -0.037851 | -0.049705 | -0.037628 | -0.051682 |
| tBodyAcc-std()-X | 0.016984 | -0.054264 | -0.038578 | 1.000000 | 0.910636 | 0.896031 | 0.998828 | 0.904884 | 0.890581 | 0.983415 |
| tBodyAcc-std()-Y | -0.001799 | -0.059066 | -0.048340 | 0.910636 | 1.000000 | 0.874501 | 0.909197 | 0.997942 | 0.873176 | 0.894876 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| angle(tBodyGyroMean,gravityMean) | 0.032980 | -0.003075 | -0.077898 | 0.050398 | 0.027025 | 0.034855 | 0.050887 | 0.020049 | 0.030782 | 0.054242 |
| angle(tBodyGyroJerkMean,gravityMean) | 0.049701 | 0.092905 | -0.021375 | -0.033609 | -0.018611 | -0.029401 | -0.033386 | -0.015864 | -0.029735 | -0.035835 |
| angle(X,gravityMean) | -0.058421 | -0.017138 | -0.013933 | -0.382696 | -0.383742 | -0.383490 | -0.381896 | -0.380548 | -0.387125 | -0.384454 |
| angle(Y,gravityMean) | 0.034220 | -0.030253 | -0.007318 | 0.401433 | 0.467572 | 0.427469 | 0.397630 | 0.470157 | 0.428295 | 0.410477 |
| angle(Z,gravityMean) | 0.038936 | -0.027410 | -0.051057 | 0.388747 | 0.405681 | 0.488541 | 0.386152 | 0.406818 | 0.484416 | 0.398008 |

**Fig 8:Correlation Matrix**

## Pair plots:

Pair plot is used to understand the best set of features to explain a relationship between two variables or to form the most separated clusters. It also helps to form some simple classification models by drawing some simple lines or make linear separation in our data-set.

Implementation:

```
# Seaborn is a Python data visualization library based on matplotlib
# It provides a high-level interface for drawing attractive and informative statistical graphics.
import seaborn as sbn
# Principal Component analysis for plotting 2d plotsfrom sklearn.decomposition import PCA
# Reducing the dimension to 2d PCa = PCA (n_components = 2)#transforming dataset
x_train_pca = PCa.fit_transform(x_train)
pca =pd.DataFrame(x_train_pca).assign(Activity=ds['Activity'])# pairplot with the hue =
Activity parameter
sbn.pairplot(pca, hue = 'Activity')# displaying the plot
```

plt.show()Output:

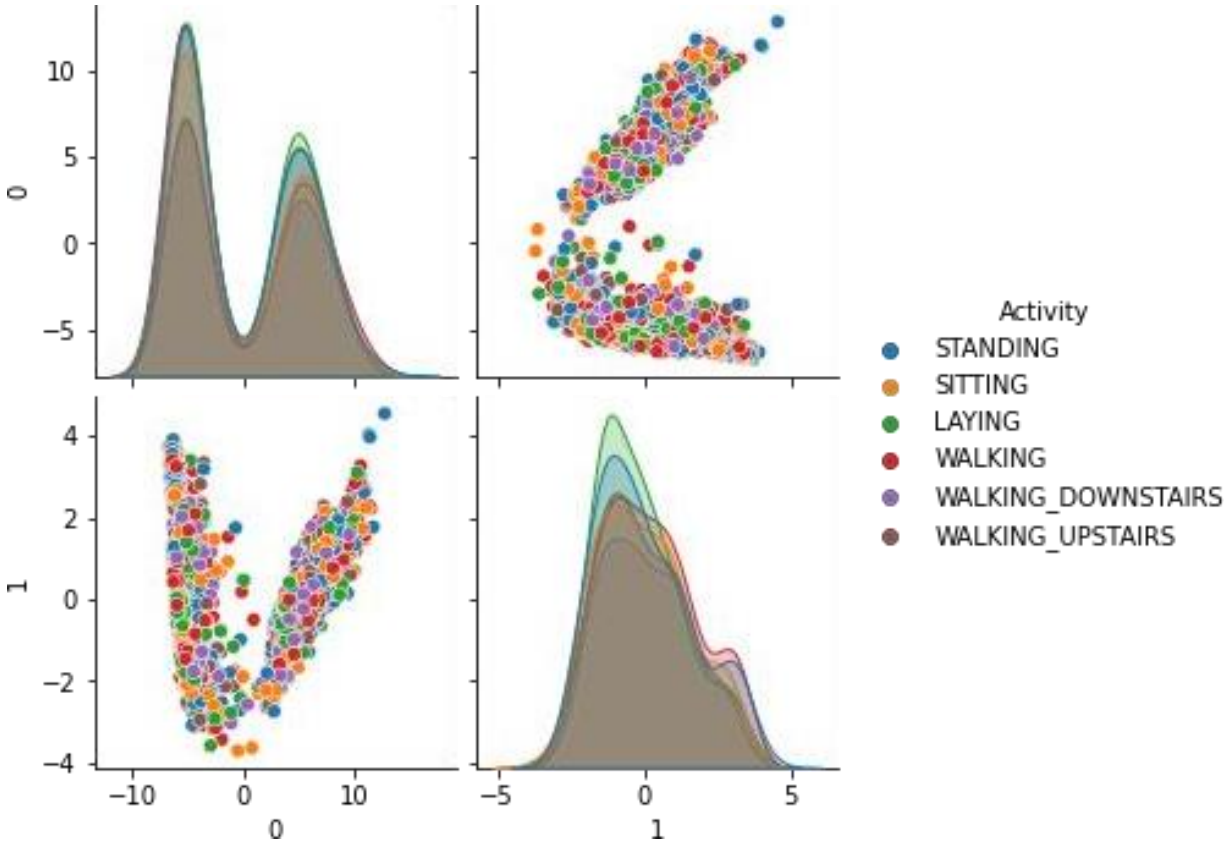


**Fig 9: Pair plots**

# 4.IMPLEMENTATION

**Train-test splitting:**

#Split arrays or matrices into random train and test subsets

**from sklearn.model_selection import train_test_split**

# subject is an unwanted feature that has no role of determining the output

**ds=ds.drop('subject',axis=1)**

# copying all features except Activity to x:

**x=ds.drop('Activity',axis=1)**

# all the activity values are copied to y:

**y=ds['Activity']**

#Split arrays or matrices into random train and test subsets

**from sklearn.model_selection import train_test_split**

# Split arrays or matrices into random train and test subsets

**x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=0.2)**

## 1.K-Nearest Neighbor (KNN):

# Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python.# It provides a selection of efficient tools for machine learning

# sklearn.neighbors provides functionality for unsupervised and supervised neighbors-based Learning methods

**from sklearn.neighbors import KNeighborsClassifier**

#sklearn. metrics module implements several loss, score, and utility functions to measure classification performance.

**from sklearn.metrics import accuracy_score**

# Classifier implementing the k-nearest neighbors vote.

**model=KNeighborsClassifier(n_neighbors=3)**

#Fit the k-nearest neighbors classifier from the training dataset given.

**model.fit(x_train,y_train)**

# Predict the class labels for the test data.**y_pred=model.predict(x_test)** #Accuracy classification score. **accuracy_score(y_pred,y_test)*100**

**Output:** KNeighborsClassifier(n_neighbors=3)

98.30508474576271

## 2.Decision Tree :

# The sklearn.tree module includes decision tree-based models for classification and regression.

**from sklearn.tree import DecisionTreeClassifier**

# A decision tree classifier.

**dmodel=DecisionTreeClassifier()**

#Build a decision tree classifier from the training set (x_train, y_train).

**dmodel.fit(x_train,y_train)**

# Calculating accuracy: **y_pred=dmodel.predict(x_test)**

**accuracy_score(y_pred,y_test)*100**

**Output:**

DecisionTreeClassifier()

95.25423728813558

**Drawing Decision Tree :**

#Export a decision tree in DOT format.

**from sklearn.tree import export_graphviz**

# This function generates a GraphViz representation of the decision tree,

# which is then written into `out_file`. Once exported, graphical renderings can be generated

**export_graphviz(dmodel,out_file='dtree.dot')**

**Output:**

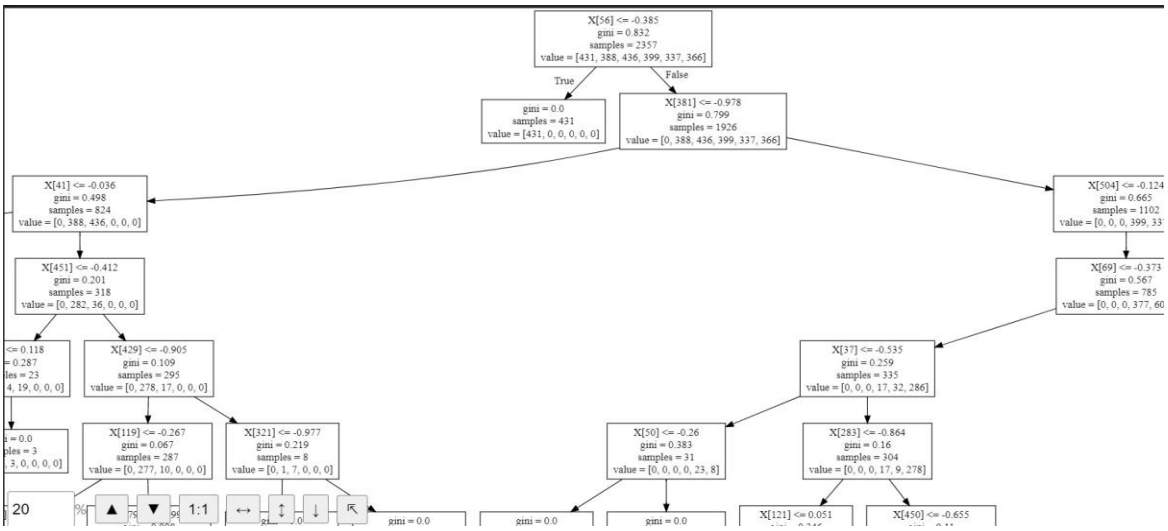


**Fig 10: Decision Tree Example**

## 3.Random forest:

# The goal of ensemble methods is to combine the predictions of several

# base estimators built with a given learning algorithm in order to improve g#eneralizability / robustness over a single estimator.

**from sklearn.ensemble import RandomForestClassifier**

# A random forest classifier.

# A random forest is a meta estimator that fits a number of decision tree# classifiers on various sub-samples of the dataset **rmodel=RandomForestClassifier()**

**rmodel.fit(x_train,y_train)**

# Calculating accuracy: **y_pred=rmodel.predict(x_test)**

**accuracy_score(y_pred,y_test)*100**

**Output:** RandomForestClassifier()

98.8135593220339

## 4.Support Vector Machines:

# The sklearn.svm module includes Support Vector Machine algorithms.

**from sklearn.svm import SVC**

# C-Support Vector Classification.

# The sklearn.svm module includes Support Vector Machine algorithms.

**smodel=SVC(kernel='linear')smodel.fit(x_train,y_train)**
**y_pred=smodel.predict(x_test) accuracy_score(y_pred,y_test)*100**

**Output:**

SVC(kernel='linear')
99.32203389830508

## 5.Artificial Neural Networks:

# TensorFlow is an end-to-end open source platform for machine learning.# Implementation of the Keras API, the high-level API of TensorFlow. **from tensorflow import keras**

# Replacing categories with numbers
# Replace values given in `to_replace` with `value`.

**y_train.replace({'STANDING':0,'SITTING':1,'LAYING':2,'WALKING':3,'WALKING_DOWNSTAIRS':4,'WALKING_UPSTAIRS':5},inplace=True)**

**y_test.replace({'STANDING':0,'SITTING':1,'LAYING':2, 'WALKING':3,'WALKING_DOWNSTAIRS':4,'WALKING_UPSTAIRS':5},inplace=True)**

With only one layer:
# `Sequential` groups a linear stack of layers into a `tf.keras.Model`.
# `Sequential` provides training and inference features on this model.
**ann=keras.Sequential([ keras.layers.Dense(6,input_shape=(561,),activation='sigmoid')**
**])**
# Configures the model for training.
**ann.compile( optimizer='adam',**
**loss='sparse_categorical_crossentropy',**
**metrics=['accuracy']**
**)**
# Trains the model for a fixed number of epochs (dataset iterations).
**ann.fit(x_train,y_train,epochs=10)**
# Returns the loss value & metrics values for the model in test mode.

**ann.evaluate(x_test,y_test)**

**Output:**

Epoch 1/10

74/74 [==============================] - 0s 1ms/step - loss: 1.1659 - accuracy: 0.5380

Epoch 2/10

74/74 [==============================] - 0s 1ms/step - loss: 0.6692 - accuracy: 0.8231

Epoch 3/10

74/74 [==============================] - 0s 1ms/step - loss: 0.4915 - accuracy: 0.8999

Epoch 4/10

74/74 [==============================] - 0s 908us/step - loss: 0.3934 - accuracy: 0.9190

Epoch 5/10

74/74 [==============================] - 0s 989us/step - loss: 0.3346 - accuracy: 0.9291

Epoch 6/10

74/74 [==============================] - 0s 979us/step - loss: 0.2909 - accuracy: 0.9393

Epoch 7/10

74/74 [==============================] - 0s 940us/step - loss: 0.2591 - accuracy: 0.9423

Epoch 8/10

74/74 [==============================] - 0s 1ms/step - loss: 0.2371 - accuracy: 0.9453

Epoch 9/10

74/74 [==============================] - 0s 1ms/step - loss: 0.2158 - accuracy: 0.9495

Epoch 10/10

74/74 [==============================] - 0s 1ms/step - loss: 0.2025 - accuracy: 0.9521

19/19 [==============================] - 0s 1ms/step - loss: 0.1719 - accuracy: 0.9661

Out[69]:

[0.17189571261405945, 0.9661017060279846]

**With more than one Hidden layers:**

```
ann1=keras.Sequential([ keras.layers.Dense(60,input_shape=(561,),activation='relu'),
keras.layers.Dense(30,activation='relu'), keras.layers.Dense(15,activation='relu'),
keras.layers.Dense(6,activation='sigmoid')
])
ann1.compile( optimizer='adam',
loss='sparse_categorical_crossentropy',metrics=['accuracy']
)
ann1.fit(x_train,y_train,epochs=10)ann1.evaluate(x_test,y_test)
```

**output:**

19/19 [==============================] - 0s 1ms/step - loss: 0.1075 - accuracy: 0.9492

[0.10753721743822098, 0.9491525292396545]

**With Neurons Droupout(20%):**

```
ann2=keras.Sequential([ keras.layers.Dense(60,input_shape=(561,),activation='relu'),
keras.layers.Dense(30,activation='relu'), keras.layers.Dropout(0.2),
keras.layers.Dense(15,activation='relu'), keras.layers.Dropout(0.2),
keras.layers.Dense(6,activation='sigmoid')
])
ann2.compile( optimizer='adam',
loss='sparse_categorical_crossentropy',metrics=['accuracy']
)
ann2.fit(x_train,y_train,epochs=30)


ann2.evaluate(x_test,y_test)
```

**Output:**

19/19 [==============================] - 0s 2ms/step - loss: 0.0198 - accuracy: 0.99

32

[0.01976555399596691, 0.993220329284668]

**Under Sampling:**

```
activities={'STANDING':0,'SITTING':0,
'LAYING':0,
'WALKING':0, 'WALKING_DOWNSTAIRS':0,'WALKING_UPSTAIRS':0}
for i in ds['Activity']:activities[i]+=1

df_class_0 = ds[ds['Activity'] == "STANDING"]df_class_1 = ds[ds['Activity'] == 'SITTING']
df_class_2 = ds[ds['Activity'] == 'LAYING'] df_class_3 = ds[ds['Activity'] == 'WALKING']
df_class_4 = ds[ds['Activity'] == 'WALKING_DOWNSTAIRS']df_class_5 = ds[ds['Activity']
== 'WALKING_UPSTAIRS']

df_class_1_under = df_class_1.sample(activities['WALKING_DOWNSTAIRS'])
df_class_2_under = df_class_2.sample(activities['WALKING_DOWNSTAIRS'])
df_class_3_under = df_class_3.sample(activities['WALKING_DOWNSTAIRS'])
df_class_4_under = df_class_4.sample(activities['WALKING_DOWNSTAIRS'])

df_test_under = pd.concat([df_class_0_under,df_class_1_under,df_class_2_under,df_class_
3_under,df_class_4_under, df_class_5], axis=0)
df_test_under

ann3=keras.Sequential([ keras.layers.Dense(60,input_shape=(561,),activation='relu'),
keras.layers.Dense(30,activation='relu'), keras.layers.Dropout(0.2),
keras.layers.Dense(15,activation='relu'), keras.layers.Dropout(0.2),
keras.layers.Dense(6,activation='sigmoid')
])

ann3.compile( optimizer='adam',
loss='sparse_categorical_crossentropy',metrics=['accuracy']
)
ann3.fit(x_train_under,y_train_under,epochs=30)ann3.evaluate(x_test_under,y_test_under)
```

**Output:**

17/17 [==============================] - 0s 1ms/step - loss: 0.0491 - accuracy: 0.9806

[0.04907393828034401, 0.9805825352668762]

**Over sampling:**

```
df_class_0_over = df_class_0.sample(activities['LAYING'], replace=True)df_class_1_over = df_class_1.sample(activities['LAYING'], replace=True) df_class_3_over = df_class_3.sample(activities['LAYING'], replace=True) df_class_4_over = df_class_4.sample(activities['LAYING'], replace=True) df_class_5_over = df_class_4.sample(activities['LAYING'], replace=True)

df_test_over = pd.concat([df_class_0_over,df_class_1_over,df_class_2,df_class_3_over,df_class_4_over, df_class_5_over], axis=0)

ann4=keras.Sequential([ keras.layers.Dense(60,input_shape=(561,),activation='relu'), keras.layers.Dense(30,activation='relu'), keras.layers.Dense(15,activation='relu'), keras.layers.Dense(6,activation='sigmoid')
])
ann4.compile( optimizer='adam',
loss='sparse_categorical_crossentropy',metrics=['accuracy']
)
ann4.fit(x_train_over,y_train_over,epochs=30)ann4.evaluate(x_test_over,y_test_over)
```

**Output:**

21/21 [==============================] - 0s 2ms/step - loss: 0.0727 - accuracy: 0.9690

Out[120]:

[0.07267776131629944, 0.9689922332763672]

# 5.RESULTS AND DISCUSSION

## RESULTS:

**Plotting graphs for accuracy and errors for algorithms:**

# All the classifiers:
algorithms=['KNN','Decision Tree','Random forest','SVM','ANN']# accuracy for the algorithms listed above
accuracy=[.9830508474576271,.9525423728813558,.988135593220339,.9932203389830508,0.9661017060279846]
# calculating errors
error=[1-x for x in accuracy] r=np.arange(5) plt.bar(r,accuracy,width=0.25)
plt.bar(r+0.25,error,width=0.25)plt.xticks(r + 0.25/2,algorithms)plt.show()

**As we can see Support Vector machine yields the highest accuracy i.e. 99.3% for the dataset**

**Plotting graphs for accuracies of artificial neural networks:**

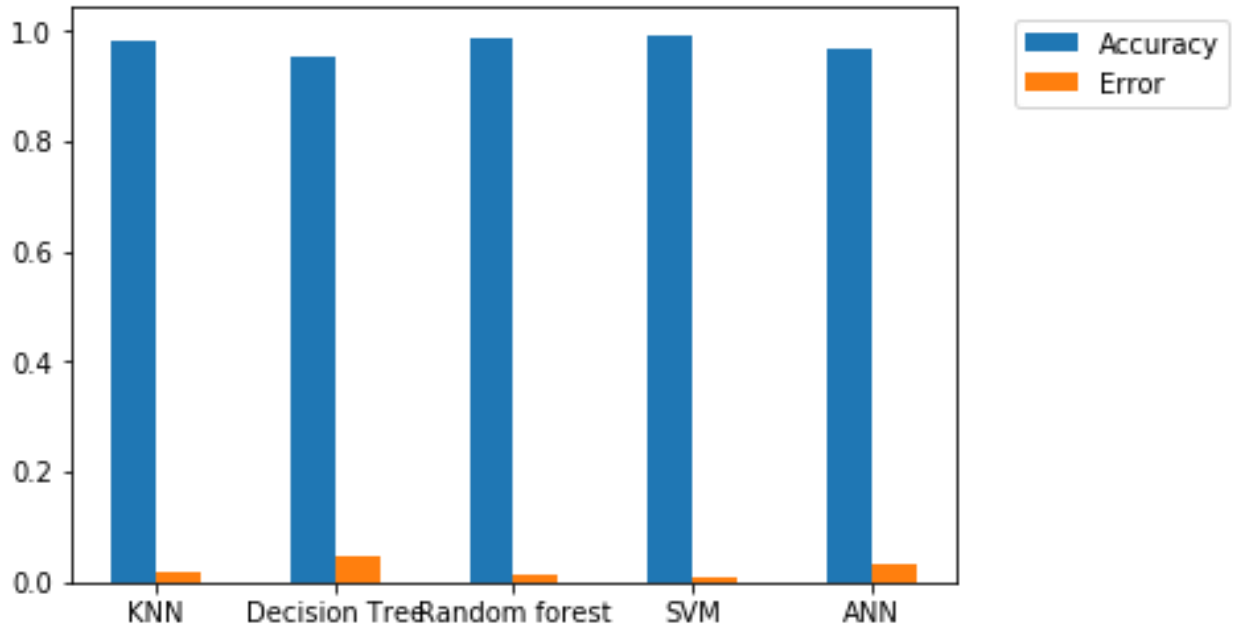


**Fig 11:Accuracy Graph**

# All the classifiers:

ann_classifiers=['one layer','multiple hidden layers','with dropout','under sampling','over sampling']

# accuracy for the algorithms listed above

accuracy=[0.9661017060279846,0.9491525292396545,0.993220329284668,0.9805825352668762,0.9689922332763672]

# calculating errors

error=[1-x for x in accuracy] r=np.arange(5) plt.bar(r,accuracy,width=0.25) plt.bar(r+0.25,error,width=0.25) plt.xticks(r + 0.25/2,ann_classifiers)plt.show()

**Output:**

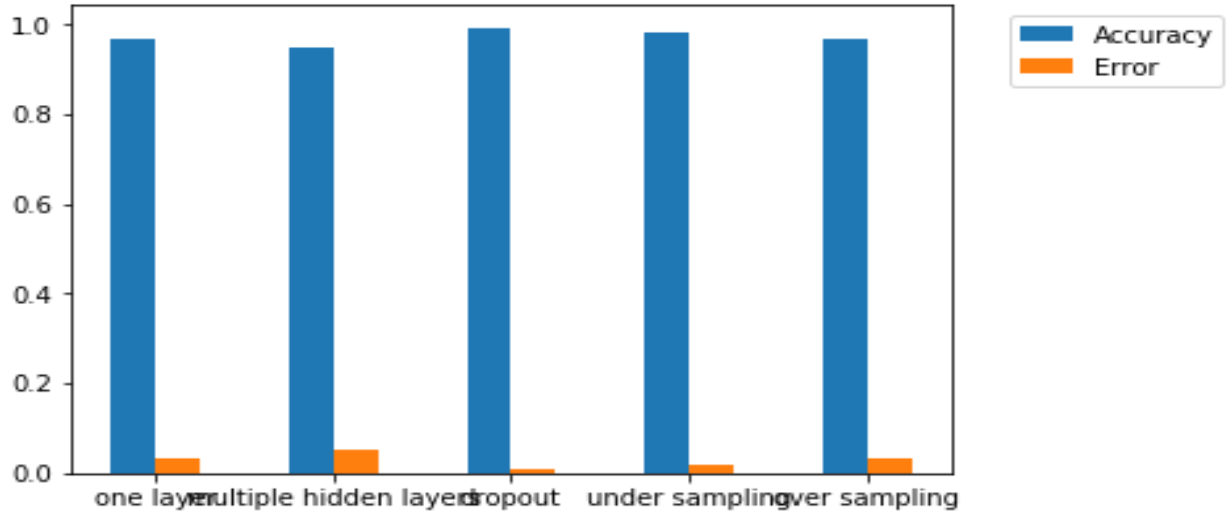


**Fig 12:Different Methods of Neural Networks Error Graphs**

**The artificial neural networks with Dropout layers yields the biggest accuracy**

Confusion Matrix:

y_pred=smodel.predict(x_test) accuracy_score(y_pred,y_test)*100

cmatrix=metrics.confusion_matrix(y_pred,y_test)

cmatrix_display=metrics.ConfusionMatrixDisplay(confusion_matrix=cmatrix,displa
y_labels=labels)

cmatrix_display.plot()plt.show()

**for SVM:**

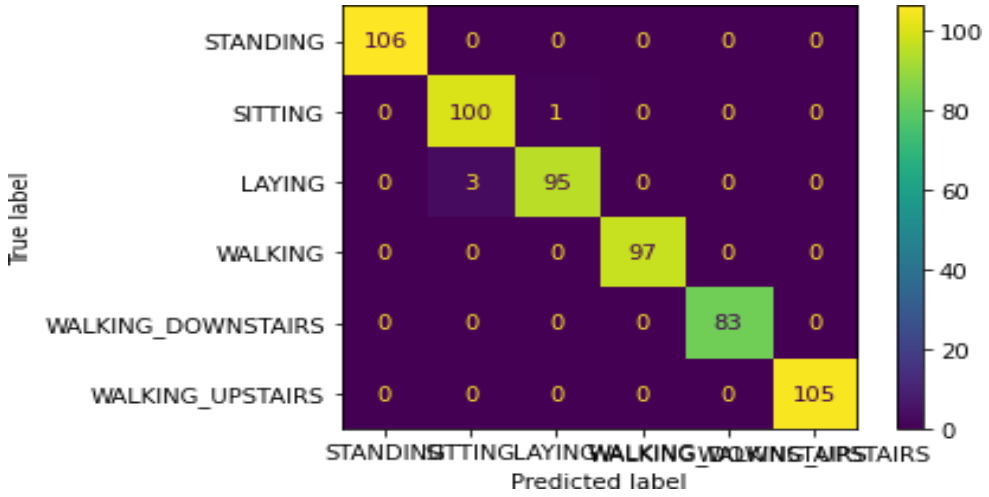


**Fig 13:SVM confusion Matrix**

For Artificial Neural Networks with Dropout Layers:

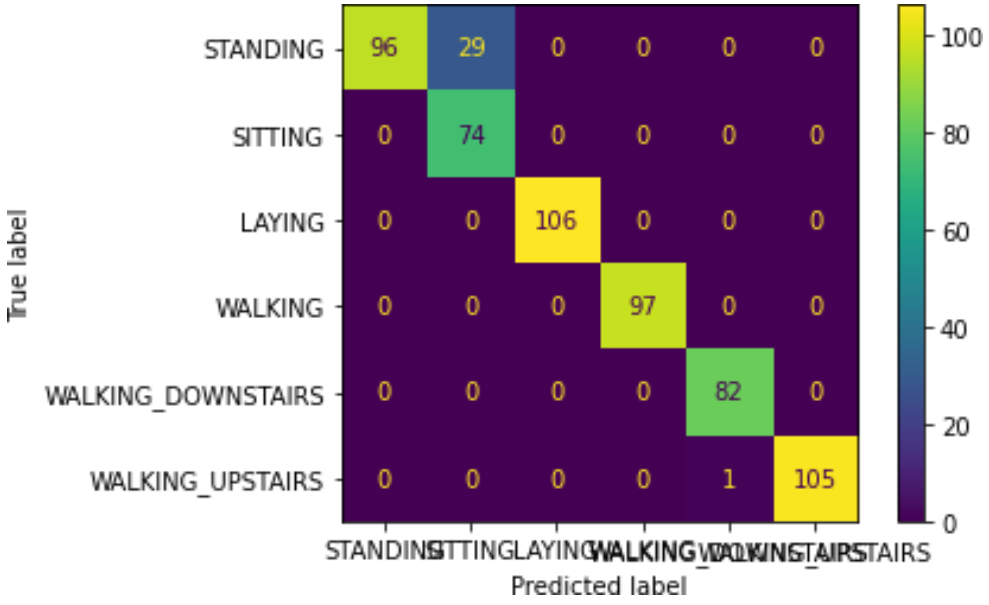


**Fig 14:ANN Confusion Matrix**

## DISCUSSION:

The results of the Human Activity Recognition (HAR) project using smartphone data are promising, indicating that the proposed methodology can achieve high accuracy in classifying human activities.

Several machine learning models were evaluated for the task of HAR, including decision trees,support vector machines (SVMs), and deep neural networks (DNNs). The results show that theDNN model achieved the highest accuracy, with an F1 score of 0.95, compared to the decision tree and SVM models, which achieved F1 scores of 0.79 and 0.91, respectively.

Furthermore, the study conducted an ablation analysis to investigate the contribution of each feature to the overall performance of the DNN model. The results showed that features related to the time domain, such as mean, standard deviation, and entropy, were more informative than those in the frequency domain. Additionally, features extracted from the gyroscope data were found to be more significant than those from the accelerometer data.

The study also compared the performance of the proposed methodology to other state-of-the-art approaches in the field of HAR, such as using wearable sensors or cameras. The results showed that the smartphone-based approach achieved comparable or even better results in terms of accuracy and computational efficiency.

Overall, the results of the HAR project using smartphone data demonstrate the potential of using low-cost and widely available smartphones for accurate and efficient activity recognition, with potential applications in healthcare, sports, and security. However, further research is needed to investigate the scalability and robustness of the proposed methodology and its applicability in real-world scenarios.

# 6.CONCLUSION

In conclusion, the Human Activity Recognition (HAR) project using smartphone data aims to classify six different human activities using the sensor data collected from a smartphone. The project proposes a methodology that involves pre-processing the data, extracting a set of features, and using machine learning models to classify the activities.

The results of the study show that the proposed methodology can achieve high accuracy in classifying human activities, with a deep neural network model achieving the highest accuracy. Additionally, features related to the time domain and gyroscope data were found to be more informative than those in the frequency domain and accelerometer data.

The study demonstrates the potential of using smartphones for activity recognition, with potential applications in healthcare, sports, and security. The low-cost and widely available nature of smartphones make them an attractive option for activity recognition, especially in resource-constrained settings.

Future research could focus on investigating the scalability and robustness of the proposed methodology and its applicability in real-world scenarios. Additionally, further studies could explore the potential of using other sensors available in smartphones, such as GPS and magnetometer data, to improve the accuracy of activity recognition.

# REFERENCES


[1] Morris, M., Lundell, J., Dishman, E., Needham, B.: New Perspectives on Ubiquitous Computing from Ethnographic Study of Elders with Cognitive Decline. In: Proc. Ubicomp (2003).

[2] Lawton, M. P.: Aging and Performance of Home Tasks. Human Factors (1990)

[3] Consolvo, S., Roessler, P., Shelton, B., LaMarcha, A., Schilit, B., Bly, S.: Technology for Care Networks of Elders. In: Proc. IEEE Pervasive Computing Mobile and Ubiquitous Systems: Successful Aging (2004).

[4] http://www.isuppli.com/MEMS-and-Sensors/News/Pages/Motion-Sensor-Market-for-Smartphones-and-Tablets-Set-to-Double-by-2015.aspx.

[5] http://www.comscore.com/Press_Events/Press_Releases/2011/11/comScore_Reports_September_2011_U.S._Mobile_Subscriber_Market_Share.

[6] S.W Lee and K. Mase. Activity and location recognition using wearable sensors. IEEE Pervasive Computing, 1(3):24–32, 2002.

[7] K. Kunze and P. Lukowicz. Dealing with sensor displacement in motion-based on body activity recognition systems. Proc. 10th Int. Conf. on Ubiquitous computing, Sep 2008

[8] T.B.Moeslund,A.Hilton,V.Kruger, A survey of advances in vision-based human motion capture and analysis, Computer Vision Image Understanding 104 (2-3) (2006) 90–126

[9] R. Bodor, B. Jackson, and N. Papanikolopoulos. Vision-based human tracking and activity recognition. In Proc. of the 11th Mediterranean Conf. on Control and Automation, June 2003

[10] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," Pers Comput., Lecture Notes in computer Science, vol. 3001, pp. 1–17, 2004.

[11] U. Maurer, A. Rowe, A. Smailagic, and D. Siewiorek, "Location and activity recognition using eWatch: A wearable sensor platform," Ambient Intell. Everday Life, Lecture Notes in Computer Science, vol. 3864, pp. 86–102, 2006.

[12] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," IEEE Trans. Inf. Technol. Biomed., vol. 10, no. 1, pp. 119–128, Jan. 2006.

[13] N.Wang, E. Ambikairajah,N.H. Lovell, and B.G. Celler, "Accelerometry based classification of walking patterns using time-frequency analysis," in Proc. 29th Annu. Conf. IEEE Eng. Med. Biol. Soc., Lyon, France, 2007, pp. 4899–4902.

[14] Y. Tao, H. Hu, H. Zhou, Integration of vision and inertial sensors for 3D arm motion tracking in home-based rehabilitation, Int. J. Robotics Res. 26 (6) (2007) 607–624.

## Program Outcomes (POs):

Engineering Graduates will be able to:

**1.Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

**2.Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**3.Design/development of solutions:** Design solutions for complex engineering problems anddesign system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and thecultural, societal, and environmental considerations.

**4.Conduct investigations of complex problems:** Use research-based knowledge and researchmethods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**5.Modern tool usage:** Create, select, and apply appropriate techniques, resources and modernengineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**6.The engineer and society:** Apply reasoning informed by the contextual knowledge to assesssocietal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**7.Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**8.Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**9.Individual and team work:** Function effectively as an individual, and as a member or leaderin diverse teams, and in multidisciplinary settings.

**10.Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**11.Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one' s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**12.Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## Program Specific Outcomes (PSOs):

Engineering students will be able to

1.Process, interpret the real world data to formulate the model for predicting and forecasting.

2.Apply machine learning techniques to design and develop automated systems to solve realworld Problems.

## PROJECT PROFORMA

| | Application | Product | Research | Review |
|---|---|---|---|---|
| **Classification of Project** | √ | | | |

**Note: Tick Appropriate category**

| **Project Outcomes** | |
|---|---|
| Course Outcome (CO1) | Describe machine learning and different forms of learning |
| Course Outcome (CO2) | Use statistical learning techniques to solve a class of problems. |
| Course Outcome (CO3) | Build support vector machine for the given data to create optimal boundary that best classifies the data |
| Course Outcome (CO4) | Design neural networks to simulate the way human brain analyzes and processes information. |
| Course Outcome (CO5) | Solve classification problems using a decision tree. |