

# **RECOGNITION OF HUMAN ACTIVITIES WITH SMART PHONE USING MACHINE LEARNING PROCESS**

Submitted in partial fulfillment of the requirements for the award of  
Bachelor of Engineering degree in Electronics and Communication Engineering

by

**S.FASIL (39130415)  
Y.ROHAN(39130526)**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING  
SCHOOL OF ELECTRICAL AND ELECTRONICS**

## **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC  
JEPPIAARNAGAR, RAJIV GANDHI SALAI, CHENNAI-600119**

**APRIL-2023**



# **SATHYABAMA**

INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC  
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119  
[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

## **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of S. FASIL (39130415) and Y. ROHAN (39130526) who carried out the project entitled **"RECOGNITION OF HUMAN ACTIVITIES WITH SMARTPHONE USING MACHINE LEARNING PROCESS"** under our supervision from July 2022 to April 2023.

**Internal Guide**

**Dr. R. PANDIAN M.E., Ph.D.**

Associate Professor, Department of ECE

**Head of the Department**

**Dr. T. RAVI, M.E., Ph.D.,**

**Submitted for Vivavoce Examination held on 19.04.2023**

**Internal Examiner**

**External Examiner**

## DECLARATION

We **S.FASIL(39130415)** and **Y.ROHAN(39130526)** hereby declare that the Project Report entitled “**RECOGNITION OF HUMAN ACTIVITIES WITH SMARTPHONE USING MACHINE LEARNING PROCESS**” done by us under the guidance of **Dr.R.PANDIAN, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

**DATE: 15.04.2023**

**PLACE: Chennai**

**SIGNATURE OF THE CANDIDATE(S)**

1.



2.



## **ACKNOWLEDGEMENT**

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D., Professor & Dean, School of Electrical and Electronics** and **Dr. T. RAVI, Ph.D., Professor & Head, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to our **Project Guide Dr. R. PANDIAN, M.E., Ph.D.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

## ABSTRACT

Recognition Human activity requires to predict the action of a person based on sensor-generated data. It has attracted major interest in the past few years, thanks to the large number of applications enabled by modern ubiquitous computing devices. It classify data into activity like Walking, walking up stairs, walking down stairs, sitting, standing, laying are recognized. Sensor data generated using its accelerometer and gyroscope, the sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters. 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. a vector of features was obtained by calculating variables from the time and frequency domain. The aim is to predict machine learning based techniques for Human Activity Recognition results in best accuracy. The analysis of dataset by supervised machine learning technique (SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. To propose a machine learning-based method to accurately predict the stock price Index value by prediction results in the form of stock price increase or stable state best accuracy from comparing supervise classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given transport traffic department dataset with the evaluation. dataset with the evaluation classification report, identify the confusion matrix and to categorizing data from priority and the result shows that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with precision, Recall and F1 Score.

## TABLESOF CONTENTS

| CHAPTER<br>NO | TITLE                         | PAGENO    |
|---------------|-------------------------------|-----------|
|               | <b>ABSTRACT</b>               | <b>v</b>  |
|               | <b>TABLEOFCONTENTS</b>        | <b>vi</b> |
|               | <b>LISTOF FIGURES</b>         | <b>ix</b> |
|               | <b>LISTOF TABLES</b>          | <b>x</b>  |
| <b>1.</b>     | <b>INTRODUCTION</b>           | <b>1</b>  |
|               | 1.1Outline                    | 1         |
|               | 1.2DataScience                | 1         |
|               | 1.3ArtificialIntelligence     | 2         |
|               | 1.4Machine Learning           | 2         |
|               | 1.5Objective                  | 5         |
|               | 1.6ProjectGoals               | 5         |
|               | 1.7Scope                      | 6         |
|               | 1.8ProblemStatement           | 6         |
| <b>2.</b>     | <b>LITERATURESURVEY</b>       | <b>7</b>  |
| <b>3.</b>     | <b>METHODOLOGY</b>            | <b>12</b> |
|               | 3.1Python                     | 12        |
|               | 3.1.1DesignPhilosophy&Feature | 12        |
|               | 3.1.2Syntaxand Semantics      | 14        |
|               | 3.1.3StatementsandControlFlow | 15        |
|               | 3.1.4Expressions              | 16        |
|               | 3.1.5Method                   | 19        |

|           |                              |           |
|-----------|------------------------------|-----------|
| 3.1.6     | Typing                       | 20        |
| 3.1.7     | Working Process              | 20        |
| 3.2       | Existing System              | 22        |
| 3.3       | Proposed System              | 22        |
| 3.4       | Project Requirement          | 24        |
| 3.4.1     | Functional Requirements      | 24        |
| 3.4.2     | Non-Functional Requirements  | 24        |
| 3.4.3     | Environment Requirements     | 25        |
| 3.5       | Preparing the Data Set       | 25        |
| 3.5.1     | Attribute Information        | 26        |
| 3.6       | Software Description         | 27        |
| 3.6.1     | Anaconda Navigator           | 27        |
| 3.6.2     | Jupyter's Notebook           | 31        |
| 3.7       | System Architecture          | 34        |
| 3.8       | Workflow Diagram             | 35        |
| 3.9       | Use Case Diagram             | 36        |
| 3.10      | Class Diagram                | 37        |
| 3.11      | Activity Diagram             | 38        |
| 3.12      | Sequence Diagram             | 39        |
| 3.13      | Entity Relationship Diagram  | 40        |
| <b>4.</b> | <b>MODULES AND ALGORITHM</b> | <b>41</b> |
| 4.1       | List of Modules              | 41        |
| 4.2       | Module Description           | 41        |
| 4.2.1     | Data Pre-Processing          | 41        |

|           |                                          |           |
|-----------|------------------------------------------|-----------|
| 4.2.2     | DataValidation/Cleaning/PreparingProcess | 44        |
| 4.2.3     | Explorationdataanalysisofvisualization   | 45        |
| 4.2.4     | ComparingAlgorithmwithPrediction         | 48        |
| 4.2.5     | PredictionResultbyAccuracy               | 49        |
| 4.3       | AlgorithmmandTechniques                  | 51        |
| 4.3.1     | AlgorithmExplanation                     | 51        |
| 4.3.2     | UsedPythonPackages                       | 51        |
| 4.3.3     | Logistic Regression                      | 52        |
| 4.3.4     | Logistic Function                        | 54        |
| 4.3.5     | LogisticRegressionEquation               | 55        |
| 4.3.6     | Decision Tree                            | 57        |
| 4.3.7     | Entropy                                  | 59        |
| 4.3.8     | Split Creation                           | 61        |
| 4.3.9     | BuildingaTree                            | 62        |
| 4.3.10    | Prediction                               | 62        |
| 4.3.11    | Assumption                               | 62        |
| 4.3.12    | RandomForestAlgorithm                    | 63        |
| <b>5.</b> | <b>RESULTSANDDISCUSSION</b>              | <b>70</b> |
| 5.1       | Deploy                                   | 70        |
| 5.1.1     | Flask(Web Framework)                     | 70        |
| <b>6.</b> | <b>CONCLUSIONANDFUTUREWORK</b>           | <b>72</b> |
| 6.1       | Conclusion                               | 72        |
| 6.2       | FutureWork                               | 72        |
|           | <b>REFERENCE</b>                         | <b>73</b> |

## LISTOFFIGURES

| SL.NO   | TITLE                         | PAGE.NO |
|---------|-------------------------------|---------|
| 1.4     | PROCESSOFMACHINELEARNING      | 3       |
| 3.3     | ARCHITECTUREOFPROPOSEDMODEL   | 23      |
| 3.5     | DETAILSOFDATASET              | 26      |
| 3.6.1   | ANACONDANAVIGATOR             | 29      |
| 3.7     | SYSTEMARCHITECTURE            | 34      |
| 3.8     | WORKFLOWDIAGRAM               | 35      |
| 3.9     | USECASEDIAGRAM                | 36      |
| 3.10    | CLASSDIAGRAM                  | 37      |
| 3.11    | ACTIVITYDIAGRAM               | 38      |
| 3.12    | SEQUENCEDIAGRAM               | 39      |
| 3.13    | ER-DIAGRAM                    | 40      |
| 4.2.1.1 | DATAUSEDFORUNBIASEDEVALUATION | 41      |
| 4.2.1.2 | DETAILEDSTATISTICS            | 42      |
| 4.2.1.3 | CORRELATION                   | 43      |
| 4.2.1.4 | WORKFLOWOFDATAPRE-PROCESSING  | 44      |
| 4.2.3.1 | DATAVISUALIZATION             | 45      |
| 4.2.3.2 | PIECHARTOFDATA VISUALIZATION  | 46      |
| 4.2.3.3 | HEATMAPPLOTDIAGRAM            | 47      |
| 4.2.3.4 | WORKFLOWOFDATAVISUALIZATION   | 48      |
| 4.3.3.1 | LABLEENCODING                 | 52      |
| 4.3.3.2 | REPORTOFLOGISTICRESULT        | 54      |
| 4.3.4   | S-FORMCURVE                   | 55      |

|          |                                 |    |
|----------|---------------------------------|----|
| 4.3.5    | WORKFLOWOFIMPLEMENTINGALGORITHM | 56 |
| 4.3.6    | DECISIONTREE                    | 57 |
| 4.3.6.1  | PARENTNODE                      | 58 |
| 4.3.7.1  | GRAPH                           | 59 |
| 4.3.7.2  | INFORMATIONGAINOFENTROPY        | 60 |
| 4.3.9    | WORKFLOWOFDECISIONTREEALGORITHM | 63 |
| 4.3.12.1 | ACCURACYCOMPARISON              | 63 |
| 4.3.12.2 | WORKINGOFRANDOMFORESTALGORITHM  | 64 |
| 4.3.12.3 | WORKFLOWOFRANDOMFORESTALGORITHM | 68 |
| 4.3.12.4 | SUPPORTVECTORMACHINES           | 69 |
| 5.1      | RESULTS                         | 71 |

#### LISTOFTABLES

| SL.NO | TITLE            | PAGE.NO |
|-------|------------------|---------|
| 3.5.1 | DETAILSOFDATASET | 26      |

# CHAPTER - I

## INTRODUCTION

### 1.1 OUTLINE:

Human Activity Recognition or HAR for short, is the problem of predicting what a person is doing based on a trace of their movement using sensors. Movements are often normal indoor activities such as standing, sitting, jumping, and going upstairs. Sensors are often located on the subject such as a smartphone or vest and often record accelerometer data in three dimensions (x, y, z). The idea is that once the subject's activity is recognized and known, an intelligent computer system can then offer assistance. It is a challenging problem because there is no clear analytical way to relate the sensor data to specific actions in a general way. It is technically challenging because of the large volume of sensor data collected (e.g., tens or hundreds of observations per second) and the classical use of hand-crafted features and heuristics from this data in developing predictive models. More recently, deep learning methods have been achieving success on HAR problems given their ability to automatically learn higher order features.

### 1.2 DATA SCIENCE:

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and 2 unstructured data and apply knowledge and actionable insights from data across a broad range of application domains. The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux. The term "data science" was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market. Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data. Data science can be defined as a blend of mathematics,

business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

### **1.3 ARTIFICIAL INTELLIGENCE:**

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI Textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

### **1.4 MACHINE LEARNING:**

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feeds the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated into three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It is provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning

dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modelling is the task of approximating a mapping function from input variables( $X$ ) to discrete output variables( $y$ ). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This dataset may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are speech recognition, handwriting recognition, bio metric identification, document classification etc.



**Fig1.4: Process of Machine learning**

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where we have input variables ( $X$ ) and an output variable ( $y$ ) and use an algorithm to learn the mapping function from the input to the output  $y = f(X)$ . The goal is to approximate the mapping function so well that when you have new input data ( $X$ ) that you can predict the output variables ( $y$ ) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised

learning requires that the data used to train the algorithm is already labelled with correct answers.

Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as "red" or "blue".

Human activities are often normal indoor activities such as standing, walking, laying, sitting. Sensors are often located on the subjects such as a smartphone or vest and often record accelerometer data in three dimensions (x, y, z). The idea is that once the subject's activity is recognized and known, an intelligent computer system can then offer assistance. It is a challenging problem because there is no clear analytical way to relate the sensor data to specific actions in a general way.

It is technically challenging because of the large volume of sensor data collected (e.g. tens or hundreds of observations per second) and the classical use of hand crafted features and heuristics from this data in developing predictive models. Human activities have been commonly used to define human behavioural patterns. The availability of sensors in mobile platforms has enabled the development of a variety of practical applications for several areas of knowledge such as:

- Health—through fall detection systems, elderly monitoring, and disease prevention.
- Internet of Things and Smart Cities—through solutions used to recognize and monitor domestic activities and electrical energy saving.
- Security—through individual activity monitoring solutions, crowd anomaly detection, and object tracking.

- Transportation—through solutions related to vehicle and pedestrian navigation. For this reason, the development of solutions that recognize human activities (HAR) through computational technologies and methods has been explored in recent years

For this reason, the development of solutions that recognize human activities (HAR) through computational technologies and methods has been explored in recent years.

In this sense, the HAR problem has previously been treated as a typical pattern recognition problem, and more specifically, a classification problem, that is, to identify the activity being performed by an individual at a given moment. Smartphones have been commonly employed to develop HAR solutions because of the ubiquitous capability and diversity of sensors embedded in such devices. Smartphones are included in the scope of wearable computing, and these devices are considered part of mobile computing based HAR systems. The advantage of smartphones over other wearable devices is associated with their ability to capture and process data, transmit and receive data, and connect with other devices or sensors available in the physical environment. Inertial sensors such as the accelerometer and gyroscope are most commonly used to capture information related to acceleration and direction of movement of the human body, respectively.

### **1.5 OBJECTIVES:**

The goal is to develop a machine learning model for real-time Human activity recognition, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

### **1.6 PROJECT GOALS:**

- Exploration data analysis of variable identification
  1. Loading the given dataset
  2. Import required libraries packages
  3. Analyse the general properties
  4. Find duplicate and missing values
  5. Checking unique and count values

- Uni-variata data analysis
  1. Rename, add data and drop the data
  2. To specify data type
- Exploration data analysis of bi-variate and multi-variate Plot diagram of pair plot, heatmap, bar chart and Histogram
- Method of Outlier detection with feature engineering
  1. method of Outlier detection with feature engineering
  2. Pre-processing the given dataset
  3. Splitting the test and training dataset
  4. Comparing the Decision tree and Logistic regression model and random forest etc
- Comparing algorithm to predict the result
  1. Based on the best accuracy

## 1.7 SCOPE:

The scope of this project is to investigate a dataset of smartphone sensor values and meteorological sector using machine learning technique. To identifying the Human behaviour of their regular activity tracing is not a easy one and try to reduces the risk factor behind the Human activity prediction, to using different algorithms and methodology based on our smartphone sensors dataset we predict the human activities.

## 1.8 PROBLEM STATEMENT:

Nowadays maximum of people are using smartphones, with the help of smartphone sensors like accelerometer and gyro, we can find the activities of the human, which is help to find out how the Activity people is inactive like walking, running, sitting. Some people are not active in real life, those people are have obesity and some other health issues. We can use this also some security purpose and Transportation.

## **CHAPTER - II**

### **LITERATURE SURVEY**

#### **GENERAL**

A literature review is a body of text that aims to review the critical points of current knowledge on and or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant to them. Loan default trends have been long studied from a socio-economic standpoint. Most economics surveys believe in empirical modelling of these complex systems in order to be able to predict the loan default rate for a particular individual. The use of machine learning for such tasks is a trend which is observing now. Some of the surveys to understand the past and present perspective of loan approval or not.

#### **REVIEW OF LITERATURE SURVEY**

Human activity recognition is an embryonic research field in smart environments. The performance of the human activities can be identified by extracting the features from the raw data sensors after pre-processing and segmentation steps. Feature selection plays an important role in activity recognition, hence appropriate features has to be chosen, dimensionality reductions has to be applied if necessary before passing it to the classifier. [1]

This paper provides a comprehensive survey for activity recognition in video surveillance is done by Sarvesh Vishwakarma, Anupam Agrawal on October (2012). It starts with a description of simple and complex human activity, and various applications. The applications of activity recognition are manifold, ranging from visual surveillance through content based retrieval to human computer interaction. The organization of this paper covers all aspects of the general framework of human activity recognition. Then it summarizes and categorizes recent-published research progresses under a general framework. In this survey, a brief overview of all pre-processing (i.e., detection, classification, and tracking) steps has been included. There are many limitations and open challenges, which we have highlighted by providing the comparison. Motion detection in dynamic scenes is a difficult task in the presence of illumination and weather changes, detection of a shadow, self-occlusion, and complete occlusion. Fast and accurate methods are still needed for segmentation techniques to affect the performance of latter stages. Description based approaches are doing well to recognize high level activities whose sub events are organized concurrently and occurring in a sequential manner in comparison to the statistical or syntactic approaches. The statistical and syntactic approaches can effectively handle the activity video polluted with noise. [2]

The true causes of the obesity epidemic are not well understood and there are few longitudinal population based data published examining this issue by many authors like P Timothy S. Church, Diana M. Thomas, Catrine Tudor-Locke, Peter T. Katzmarzyk, Conrad P. Earnest, Ruben Q. Rodarte, Corby K. Martin, Steven N on May (2011). The objective of this analysis was to examine trends in occupational physical activity during private industry since 1960 using data from the U.S. Bureau of Labor Statistics. Mean body weight was derived from the U.S. National Health and Nutrition Examination Surveys (NHANES). In the early 1960's almost half the jobs in private industry in the U.S. Since 1960 the estimated mean daily energy expenditure due to work related physical activity has dropped by more than 100 calories in both women and men. 15 Energy balance model predicted weights based on change in occupation related daily energy expenditures since 1960. Given a baseline weight of 76.9 kg in 1960-02, we estimated that a 142 calories reduction would result in an increase

in mean weight to 89.7 kg, which closely matched the mean NHANES weight of 91.8 kg in 2003-06. we estimate that daily occupation-related energy expenditure has decreased by more than 100 calories, and this reduction in energy expenditure accounts for a significant portion of the increase in mean U.S. body weights for women and men. [3]

Detecting human beings accurately in a visual surveillance system is crucial for diverse application areas including abnormal event detection, human gait characterization, congestion analysis, person identification, gender classification and fall detection for elderly people by Sarvesh Vishwakarma·Anupam Agrawal on May (2011). The first step of the detection process is to detect an object which is in motion. Object detection could be performed using background subtraction, optical flow and spatio-temporal filtering techniques. Once detected, a moving object could be classified as a human being using shape-based, texture-based or motion-based features. A comprehensive review with comparisons on available techniques for detecting human beings in surveillance videos is presented in this paper. At the end of this paper, a discussion is made to point the future work needed to improve the human detection process in surveillance videos. These include exploiting a multiview approach and adopting an improved model based on localized parts of the image. [4]

Vision based human motion recognition has fascinated many researchers due to its critical challenges and a variety of applications. The applications range from simple gesture recognition to complicated behaviour understanding in surveillance system by the authors Geetanjali Vinayak Kale, Varsha Hemant Patil on 2nd July (2016). This leads to major development in the techniques related to human motion representation and recognition. This paper discusses applications, general framework of human motion recognition, and the details of each of its components. 16 The paper emphasizes on human motion representation and the recognition methods along with their advantages and disadvantages. This study also discusses these selected literature, popular datasets, and concludes with the challenges in the domain along with a future direction. Hierarchical approaches have shown great success for the recognition of complicated actions and interactions. Techniques like bag-of-words, and HMM that

has shown success in speech and text recognition and is successfully applied for action recognition. Advances in fields like Artificial Intelligence and Machine Learning need to be applied for human motion recognition. Now, with efforts of the research community in human motion recognition, products with intelligent camera systems for social and commercial applications should be made available in the market. [5]

The greater part of these applications require a robotized acknowledgment of abnormal state exercises, made out of various straightforward (or nuclear) activities of people. This article gives a point-by-point diagram of different best-in-class investigation papers on human action acknowledgment. A methodology-based scientific classification is picked that analyzes the points of interest and restrictions of each methodology. Ene et al. (2016) discussed the usage of Low-cost RGB-D sensors in surveillance, human-computer interaction and computed features from the skeleton joints. Histogram was developed from the skeleton features which was denoted as various key poses. The current methodologies concentrated on minor AI estimates which don't give the best outcomes when contrasted with the presently accessible arrangements. They are inclined to overfitting, underfitting, on account of choice trees in Random Forest. [6]

Ms. S. Roobini et al. (2019) used a deep learning approach to identify human motion. And also compared Convolutional Neural Network with Long-Short Term Memory and Recurrent Neural Network with Long-Short Term Memory, finally proved that Recurrent Neural Network with Long Short Term Memory (RNNLSTM) provides better accuracy with lower mean absolute percentage error. Hence they suggested RNNLSTM can be used to reduce the human loss of lives in recognizing the activities of human in the real world. [7]

Recognizing human activities from video sequences or still images is a challenging task due to problems, such as background clutter, partial occlusion, changes in scale, viewpoint, lighting, and appearance. This is done by Michalis Vrigkas<sup>1</sup>, Christophoros Nikou<sup>1</sup> and Ioannis A. Kakadiaris on 16th Nov (2020). Many applications, including video surveillance systems, human-computer interaction, and robotics for human behaviour characterization, require a multiple activity recognition system. In this work, we provide a detailed review of recent and state-of-the-art research advances in the

field of human activity classification. We propose a categorization of human activity methodologies and discuss their advantages and limitations. In particular, Moreover, we provide a comprehensive analysis of the existing, publicly available human activity classification datasets and examine the requirements for an ideal human activity recognition dataset. A comprehensive review of existing human activity classification benchmarks was also presented and we examined the challenges of data acquisition to the problem of understanding human activity. Finally, we provided the characteristics of building an ideal human activity recognition system. [8]

## CHAPTER - III

### METHODOLOGY

#### 3.1 PYTHON INTRODUCTION:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming.

It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020. Python consistently ranks as one of the most popular programming languages.

##### **3.1.1 Design Philosophy & Feature:**

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Python's design offers some support for functional programming in the Lisp

tradition. It has filter, map and reduce functions, list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture.

"Python's developers strive to avoid premature optimization and reject patches to non-critical parts of the C-Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which

translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim to keep the language fun to use. This is reflected in its name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wider range of meanings related to program style. To say that code is *pythonic* is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonistas*.

### **3.1.2 Syntax and Semantics:**

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

*Main Article: Python syntax and semantics & Indentation*

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the *off-side rule*, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

### **3.1.3 Statements and Control Flow:**

Python's statements include (among others):

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII) - like behavior and replaces a common try/finally idiom.
- The break statement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.

- The `pass` statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The `assert` statement, used during debugging to check for conditions that should apply.
- The `yield` statement, which returns a value from a generator function and `yield` is also an operator. This form is used to implement co-routines.
- The `return` statement, used to return a value from a function.
- The `import` statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (`=`) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type. Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.[80][81] However, better support for co-routine-like functionality is provided, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed uni-directionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

#### **3.1.4 Expressions:**

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer `//`)

division) // and floating-point/division. Python also uses the `**` operator for exponentiation.

- From Python 3.5, the new `@` infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.
- From Python 3.8, the syntax `=`, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.
- In Python, `==` compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the `equals()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example `A<=B<=C`.
- Python uses the words `and`, `or`, `not` for its boolean operators rather than the symbolic `&&`, `||`, `!` used in Java and C.
- Python has a type of expression termed a list comprehension as well as a more general expression termed a generator expression.
- Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.
- Conditional expressions in Python are written as `x if c else y` (different in order of operands from the `c ? x : y` operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as `[1, 2, 3]`, are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as `(1, 2, 3)`, are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The `+` operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable `t` initially equal to `(1, 2, 3)`, executing `t = t + (4, 5)` first evaluates `t + (4, 5)`, which yields `(1, 2, 3, 4, 5)`, which is then assigned back to `t`,

thereby effectively "modifying the contents" of `t`, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.

- Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equals sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

- Python has a "string format" operator `%`. This functions analogously to `printf` format strings in C, e.g. `"spam=%segs=%d"%( "blah", 2)` evaluates to `"spam=blahegs=2"`. In Python 3 and 2.6+, this was supplemented by the `format()` method of the `str` class, e.g. `"spam={0} eggs={1}".format("blah", 2)`. Python 3.6 added "f-strings": `blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'`

- Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. `"spam" + "eggs"` returns `"spameggs"`. Even if your strings contain numbers, they are still added as strings rather than integers. E.g. `"2" + "2"` returns `"2"`.

- Python has various kinds of string literals:
  - o Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (`\`) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".
  - o Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.

- Python has array index and array slicing expressions on lists, denoted as `a[Key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including,

the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example `a[:]` returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops
- Conditional expressions vs. if blocks
- The `eval()` vs. `exec()` built-in functions (in Python 2, `exec` is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as `a=1` cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator `=` for an equality operator `==` in conditions: `if (c==1) {...}` is syntactically valid (but probably unintended) C code but `if c=1:...` causes a syntax error in Python.

### **3.1.5 Methods:**

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter access instance data, in contrast to the implicit `self` (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this Python also provides methods, sometimes called d-under methods due to their names beginning and ending with double-underscores, to extend the functionality of custom classes to support native functions such as `print`, `length`, comparison, support for arithmetic operations, type conversion, and many more.

### **3.1.6 Typing:**

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, `SpamClass()` or `EggsClass()`), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta-programming and reflection. Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from `object` and are instances of `type`). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, Python. An experimental optional static type checker named `mypy` supports compile-time type checking.

### **3.1.7 Working Process:**

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization
- machine learning models, pick the best and build confidence that the accuracy is reliable

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

- Installing the Python Anaconda platform.
- Loading the dataset.
- Summarizing the dataset.
- Visualizing the dataset.
- Evaluating some algorithms.
- Making some predictions

### 3.2 EXISTING SYSTEM:

This paper proposes and develops a cascaded deep neural network (CDNN) to analyze data, collected using the sensors of smart-phones, to accurately localize an object in an indoor environment. There are many existing studies that have attempted to identify the location of an inhabitant in a room through the analysis of the radio signal strength (RSS), with varying success. The strength of the RSS varies with distance and the presence of obstacles within the line of sight. As a result, an automated system using RSS signal in one environment might not work in another one. In this paper therefore, we propose and develop a different localization method based on data collected from different sensors embedded in a smart-phone. To analyze and predict the exact location within a very short distance (say a 1 to 1.5 m radius), we develop a novel CDNN. The indoor localization of objects has a lot of application inside offices, hospitals and public places. The proposed CDNN suffers from space and computational complexities, specially for training each of the DNNs in the CDNN. We also plan to improve the CDNN structure such that the number of DNNs can be reduced without affecting the localization accuracy.

#### 3.2.1 Drawbacks:

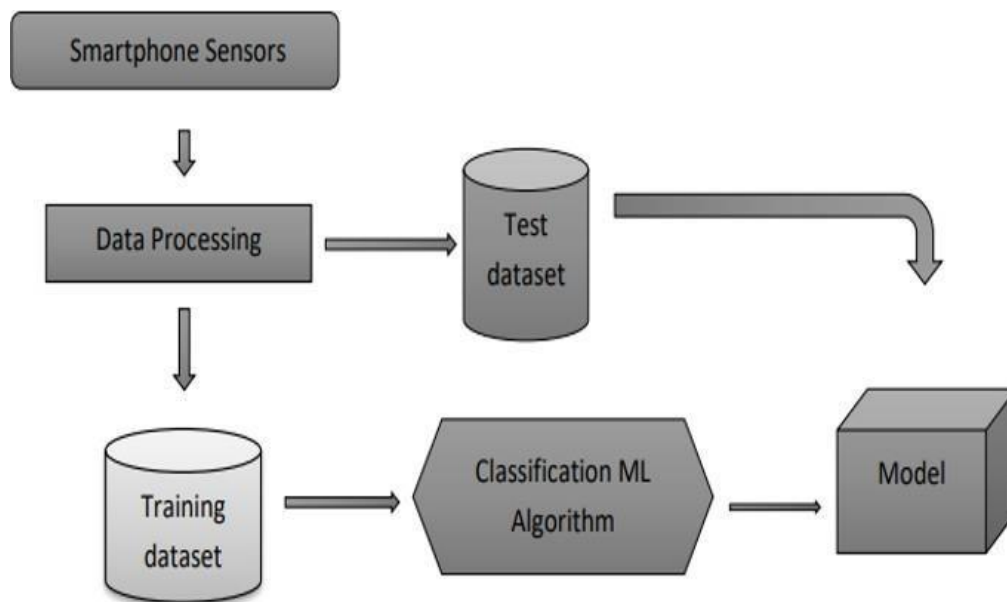
- The CDNN achieves only 80.41% and 74.14% localization accuracies for the training and testing data.
- Evidently proves the difficulty of localizing the exact position of the object within a very short distance/radius (say a 1 to 1.5 m radius).

### 3.3 PROPOSED SYSTEM:

The process of human activities recognition is very similar to a general purpose pattern recognition system and corresponds to a set of steps ranging from data collection to activities classification. This process involves a set of transformations of the raw data extracted from sensors to generate efficient classification models of human activities. The HAR methodology for smartphones equipped with inertial sensors can approach based on machine learning techniques as shallow algorithms (e.g., SVM, decision tree, Random forest). To find several information's like, variable

identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset.

- While applying photo graphic based method is critical to evaluate parameters and it's taken data size is high
- To overcome this method to implement machine learning approach by user interface of GUI application
- Multiple datasets from different sources would be combined to form a generalized dataset, and then different machine learning algorithms would be applied to extract patterns and to obtain results with maximum accuracy.



***Fig:3.3:ArchitectureofProposedModel***

### **3.3.1 Advantages:**

- These reports are to the investigation of applicability of machine learning techniques for human activity recognition using smartphone sensors.
- Finally, it highlights some observations on future research issues, challenges, and needs.

### **3.4 PROJECT REQUIREMENTS:**

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
  - A. Hardware requirements
  - B. software requirements

#### **3.4.1 Functional Requirements:**

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn

#### **3.4.2 Non-Functional Requirements:**

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results

## 5. Predictionthe result

### **3.4.3 EnvironmentalRequirements:**

#### 1. SoftwareRequirements:

OperatingSystem: Windows

Tool: AnacondawithJupyterNotebook

#### 2. Hardwarerequirements:

Processor: PentiumIV/III

Harddisk: minimum80GB

RAM: minimum2 GB

### **3.5 PREPARINGTHEDATASET:**

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed four activities (WALKING, SITTING, STANDING,LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been videorecorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into bodyacceleration and gravity. The 33 gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cut off frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain

### 3.5.1 Attribute Information:

For each record in the dataset the following is provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 19-feature vector with time and frequency domain variables.
- Its activity label.
- An identifier of the subject who carried out the experiment. Table shows details of the datasets

| Variable         | Description                        |
|------------------|------------------------------------|
| SmartPhone       | REDMIK20PRO                        |
| Sensors          | Accelerometer and Gyroscope        |
| Axis             | 3-axis(x, y, z)                    |
| No.of volunteers | 30                                 |
| Volunteers Age   | 19-48                              |
| Features         | 19                                 |
| Activates        | WALKING, SITTING, STANDING, LAYING |

**Table 3.5.1: Detail of Dataset**

### 3.6 SOFTWARE DESCRIPTION:

Anaconda is a free and open-sourced distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the `conda install` command or using the `pip install` command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together.

Custom packages can be made using the `conda build` command, and can be shared with others by uploading them to Anaconda Cloud,[10] PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

#### 3.6.1 Anaconda Navigator:

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

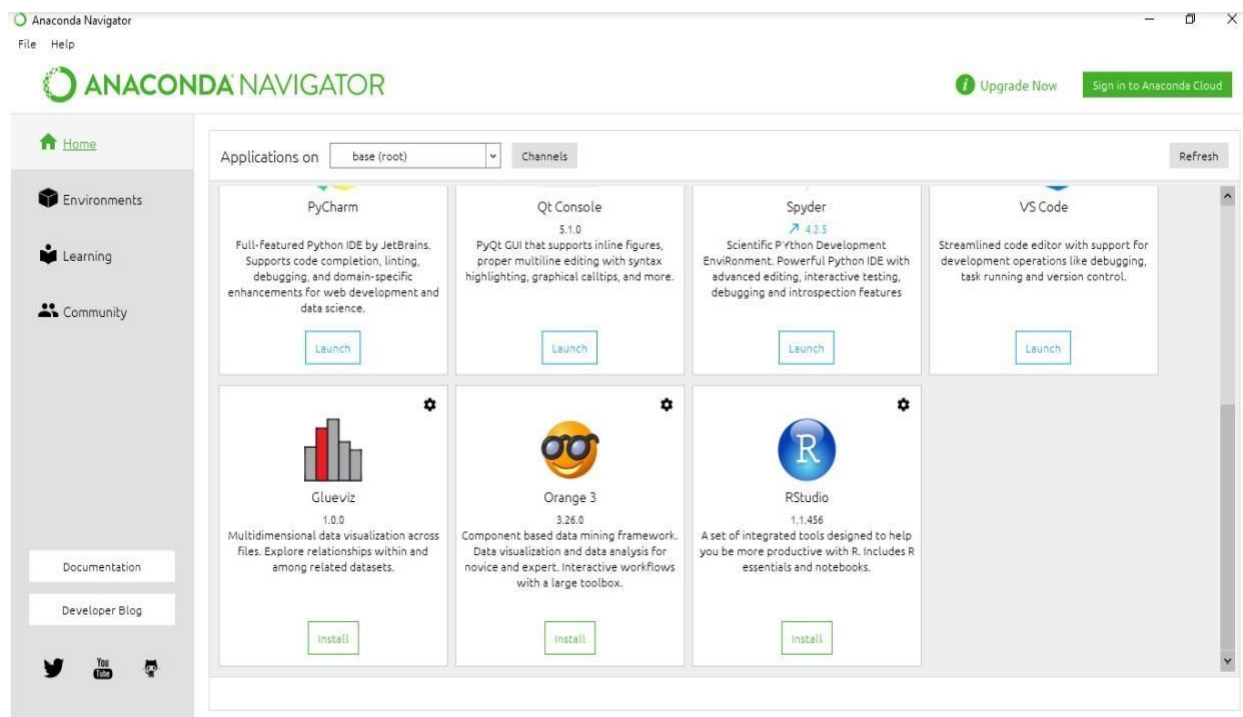
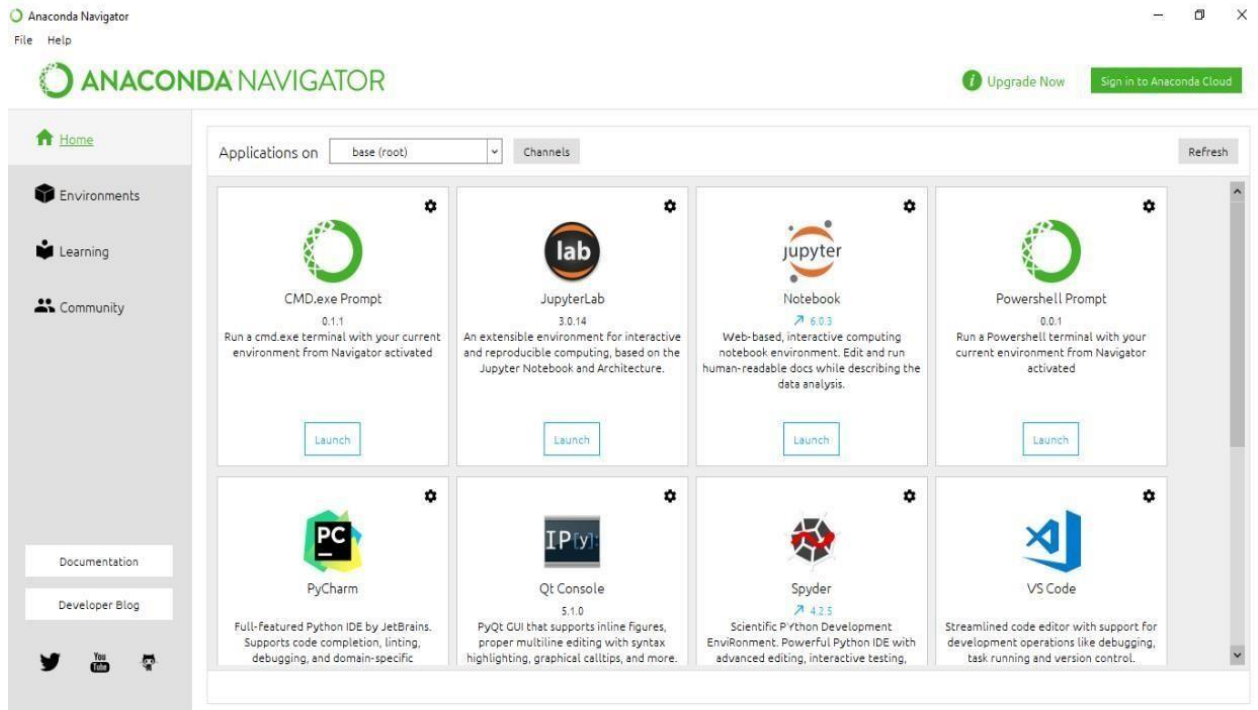
In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them -all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab36
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange3App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)



**Fig3.6.1:AnacondaNavigator**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with `conda list` on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and don't want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda

Conda:

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many software which will help you to build your machine learning project and deep learning project. These softwares have great graphical user interface and these will make your work easy to do. You can also use it to run your python script. These are the software carried by anaconda navigator.

### 3.6.2 Jupyter Notebook:

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Pérez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc...) as well as executable documents which can be run to perform data analysis.

The easiest way to install the Jupyter Notebook App is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called Anaconda.

*Launching Jupyter Notebook App:* The Jupyter Notebook App can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (cmd on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

### 3.6.2.1 Save notebooks:

Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

### 3.6.2.2 Executing a notebook:

Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- Launch the jupyter notebook app
- In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- Click on the menu Help -> User Interface Tour for an overview of the Jupyter Notebook App user interface.
- You can run the notebook document step-by-step (one cell a time) by pressing shift + enter.
- You can run the whole notebook in a single step by clicking on the menu Cell -> Run All.
- To restart the kernel (i.e. the computational engine), click on the menu Kernel -> Restart. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...)

*Purpose:* To support interactive data science and scientific computing across all programming languages.

*File Extension:* An IPYNB file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

*Jupyter Notebook App:* The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as

described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

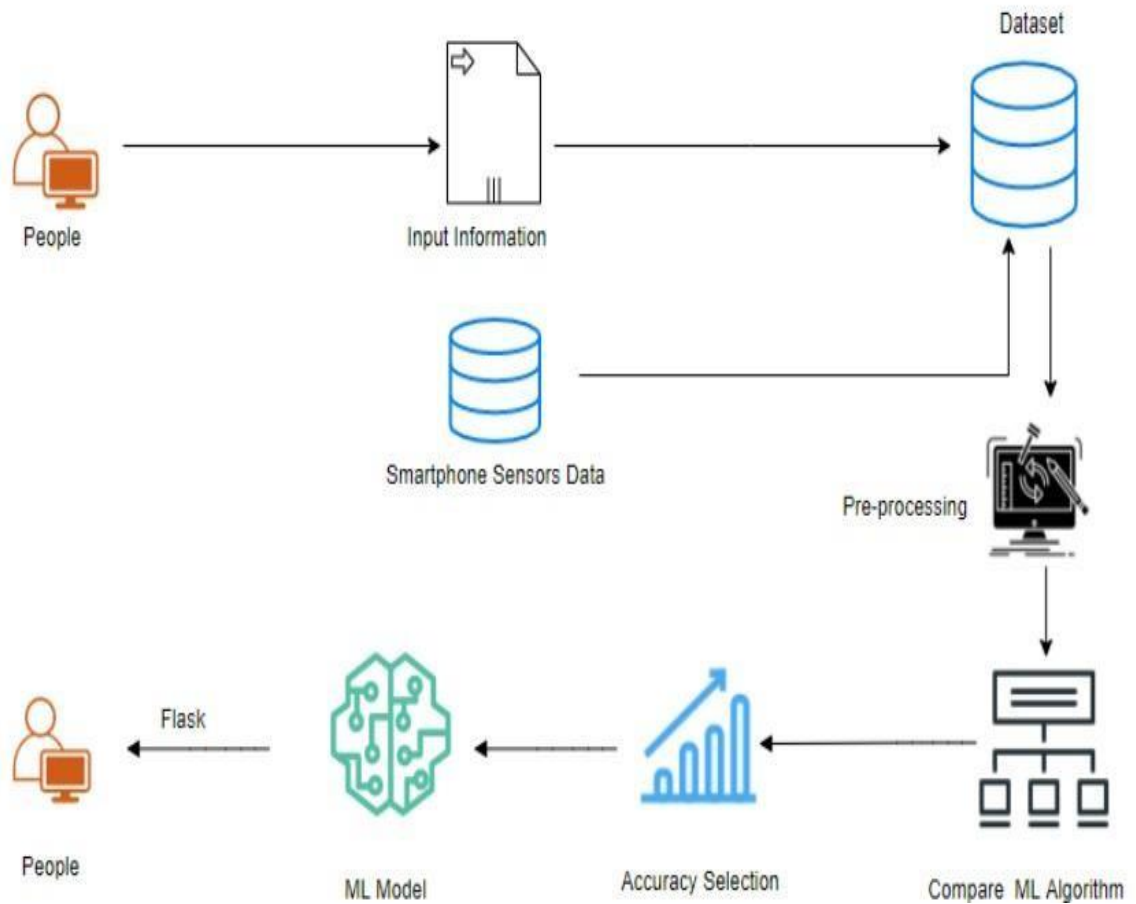
*kernel:*

A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The Python kernel, referenced in this guide, executes Python code. Kernels for many other languages exist (official kernels). When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down.

*Notebook Dashboard:*

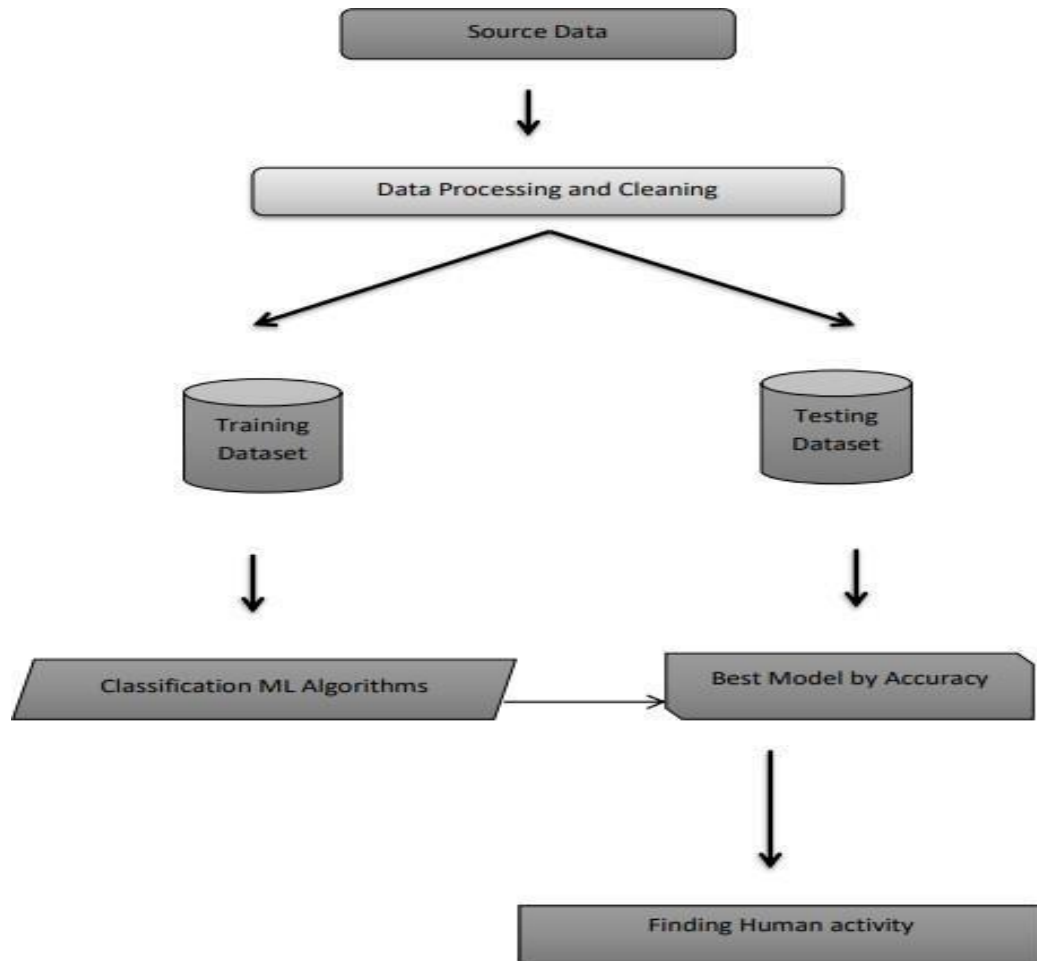
The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown). The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

### 3.7 SYSTEMARCHITECTURE:



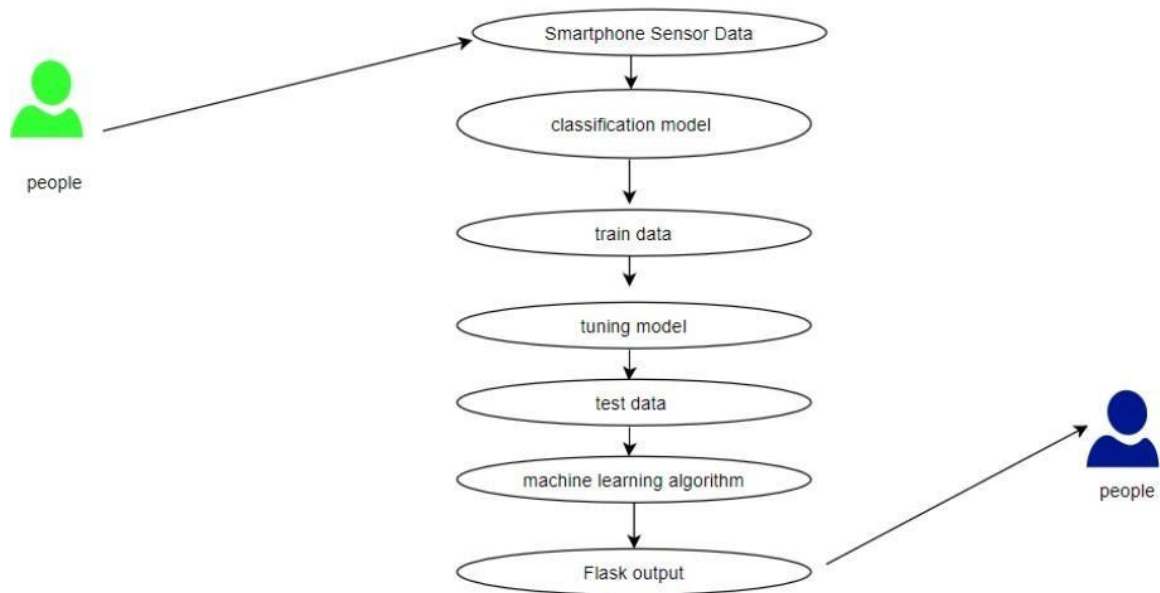
**Fig3.7: System Architecture**

### 3.8 WORKFLOWDIAGRAM:



***Fig3.8: Workflow Diagram***

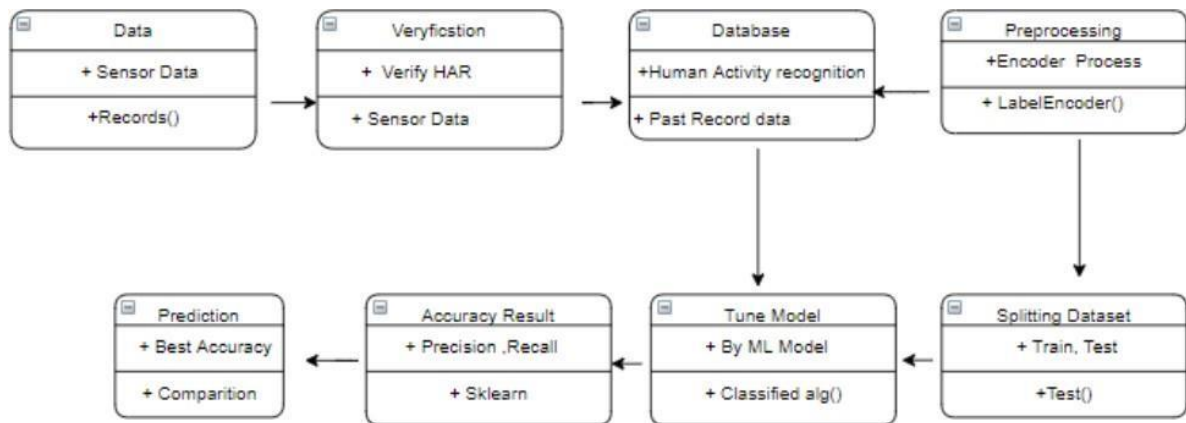
### 3.9 USECASEDIAGRAM:



**Fig3.9: UseCase Diagram**

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analysed the functionalities are captured in use cases. So, it can say that uses cases are nothing, but the system functionalities written in an organized manner

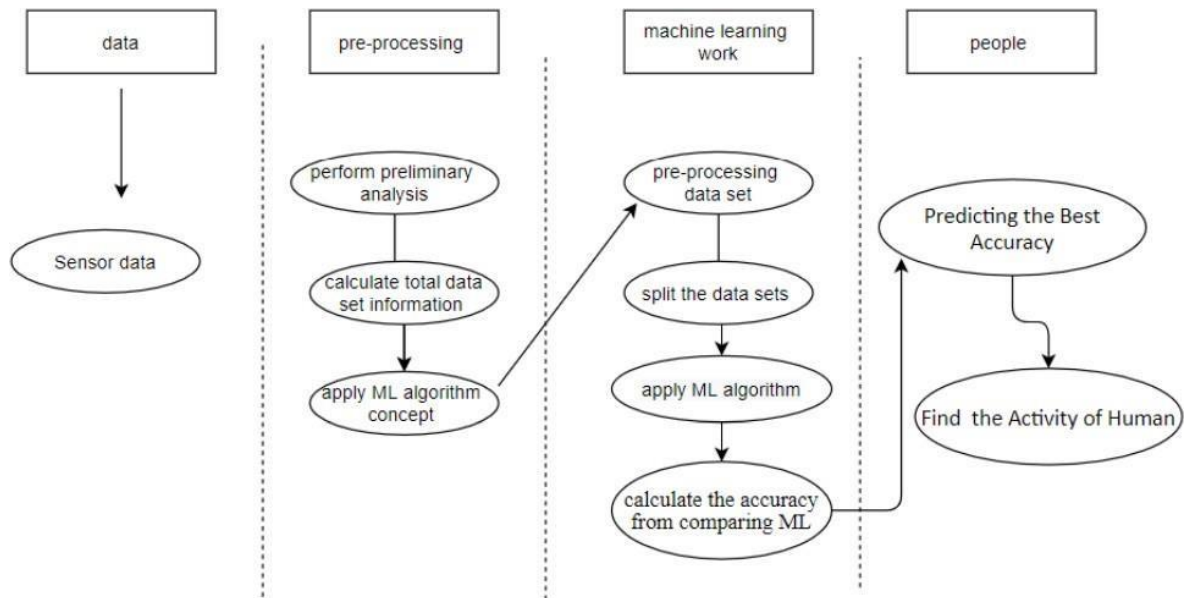
### 3.10 CLASSDIAGRAM:



**Fig3.10:Class Diagram**

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance. Responsibility (attributes and methods) of each class should be clearly identified for each class. Minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

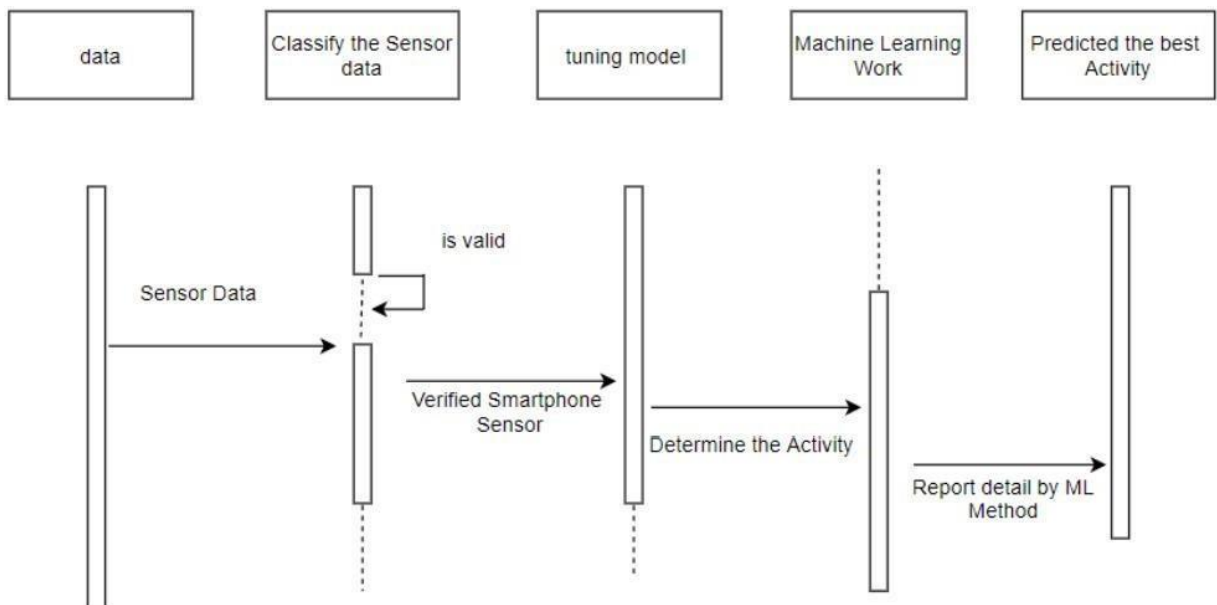
### 3.11 ACTIVITYDIAGRAM:



**Fig3.11:Activity Diagram**

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams look like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

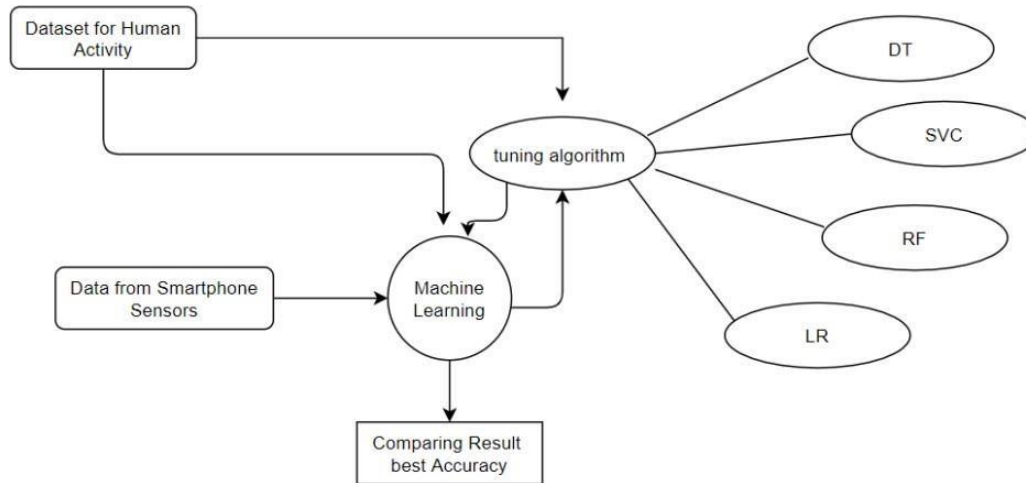
### 3.12 SEQUENCEDIAGRAM:



**Fig3.12:SequenceDiagram**

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequencediagramsarethemostpopular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data modelsareinmyopinionthelmostimportantdesign-levelmodelsformodernbusiness application development.

### 3.13 ENTITYRELATIONSHIPDIAGRAM(ERD):



**Fig3.13:EntityRelationshipDiagram(ERD)**

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modelling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

## CHAPTER – IV

### MODULES AND ALGORITHMS

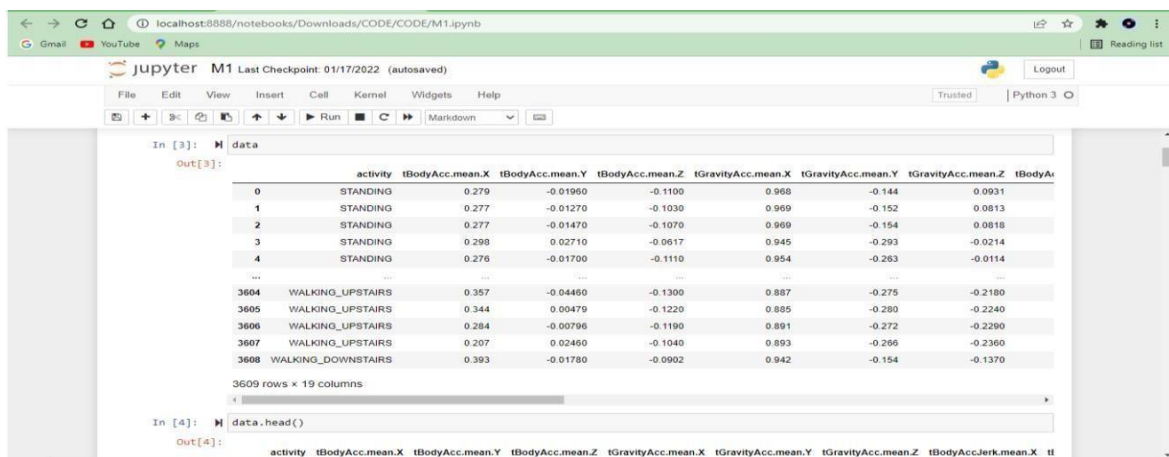
#### 4.1 LIST OF MODULES:

- Data Pre-processing
- Data Analysis of Visualization
- Comparing Algorithm with prediction in the form of best accuracy result
- Deployment Using Flask

#### 4.2 MODULE DESCRIPTION:

##### 4.2.1 Data Pre-processing:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.



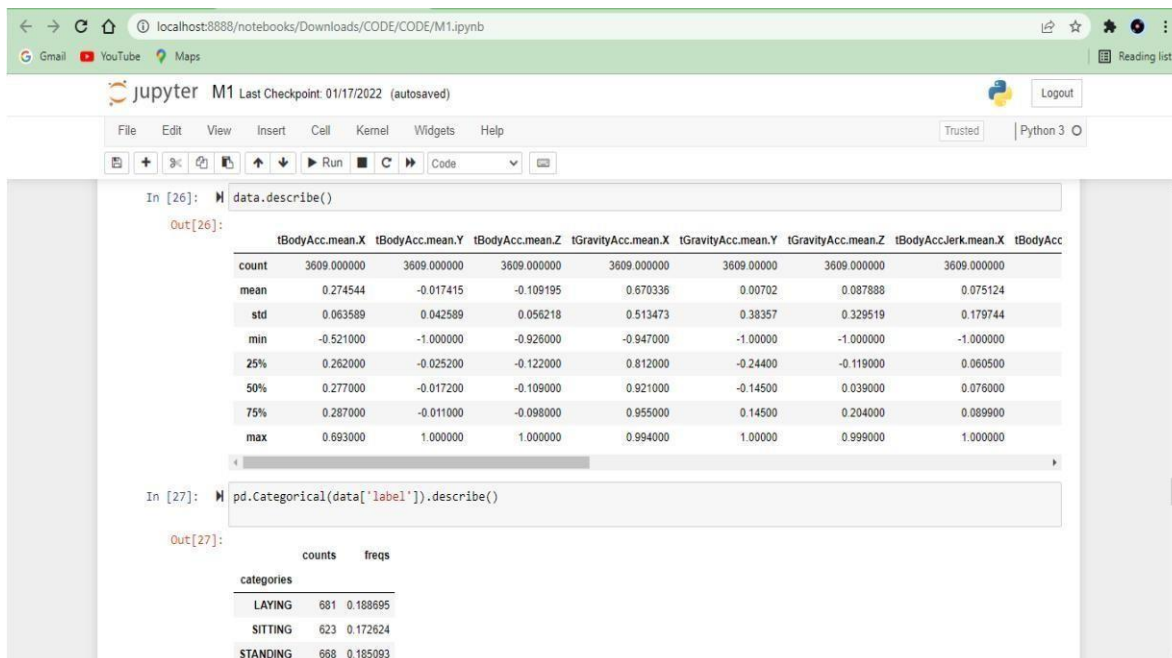
The screenshot shows a Jupyter Notebook interface with a dataset loaded into a variable named 'data'. The dataset has 3609 rows and 19 columns. The first few rows of the data are as follows:

|      | activity           | tBodyAcc.mean.X | tBodyAcc.mean.Y | tBodyAcc.mean.Z | tGravityAcc.mean.X | tGravityAcc.mean.Y | tGravityAcc.mean.Z | tBodyAcc... |
|------|--------------------|-----------------|-----------------|-----------------|--------------------|--------------------|--------------------|-------------|
| 0    | STANDING           | 0.279           | -0.01960        | -0.1100         | 0.968              | -0.144             | 0.0931             |             |
| 1    | STANDING           | 0.277           | -0.01270        | -0.1030         | 0.969              | -0.152             | 0.0813             |             |
| 2    | STANDING           | 0.277           | -0.01470        | -0.1070         | 0.969              | -0.154             | 0.0818             |             |
| 3    | STANDING           | 0.298           | 0.02710         | -0.0617         | 0.945              | -0.293             | -0.0214            |             |
| 4    | STANDING           | 0.276           | -0.01700        | -0.1110         | 0.954              | -0.263             | -0.0114            |             |
| ...  | ...                | ...             | ...             | ...             | ...                | ...                | ...                | ...         |
| 3604 | WALKING_UPSTAIRS   | 0.357           | -0.04460        | -0.1300         | 0.887              | -0.275             | -0.2180            |             |
| 3605 | WALKING_UPSTAIRS   | 0.344           | 0.00479         | -0.1220         | 0.885              | -0.280             | -0.2240            |             |
| 3606 | WALKING_UPSTAIRS   | 0.284           | -0.00796        | -0.1190         | 0.891              | -0.272             | -0.2290            |             |
| 3607 | WALKING_UPSTAIRS   | 0.207           | 0.02460         | -0.1040         | 0.893              | -0.266             | -0.2360            |             |
| 3608 | WALKING_DOWNSTAIRS | 0.393           | -0.01780        | -0.0902         | 0.942              | -0.154             | -0.1370            |             |

The dataset is displayed as a table with 3609 rows and 19 columns. The first few rows show data for 'STANDING' activity, and the last few rows show data for 'WALKING\_UPSTAIRS' and 'WALKING\_DOWNSTAIRS' activities. The columns include activity type, mean acceleration (tBodyAcc.mean.X, tBodyAcc.mean.Y, tBodyAcc.mean.Z), mean gravity acceleration (tGravityAcc.mean.X, tGravityAcc.mean.Y, tGravityAcc.mean.Z), and mean jerk (tBodyAccJerk.mean.X, tBodyAccJerk.mean.Y, tBodyAccJerk.mean.Z).

**Fig4.2.1.1: Data used for Unbiased Evaluation.**

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model. A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

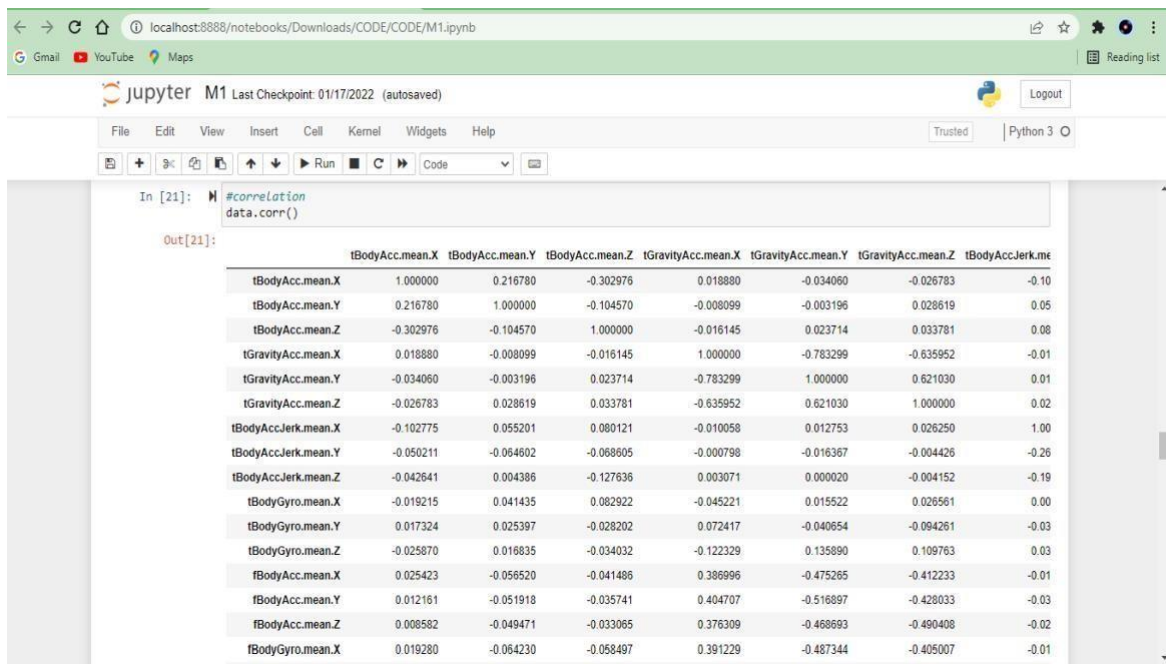


**Fig4.2.1.2:DetailedStatistics**

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values and do some basic imputation and detailed statistical approach for dealing with missing data. Before,

joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.



**Fig4.2.1.3: Correlation**

- Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:
  - import libraries for access and functional purpose and read the given dataset
  - General Properties of Analyzing the given dataset
  - Display the given dataset in the form of dataframe
  - show columns
  - shape of the dataframe
  - To describe the dataframe
  - Checking data type and information about dataset

- Checkingforduplicatedata
- Checking Missing valuesofdata frame
- Checkinguniquevaluesof dataframe
- Checkingcountvaluesofdataframe
- Rename anddropthegivendataframe
- Tospecifythetypeofvalues
- Tocreteextracolumns

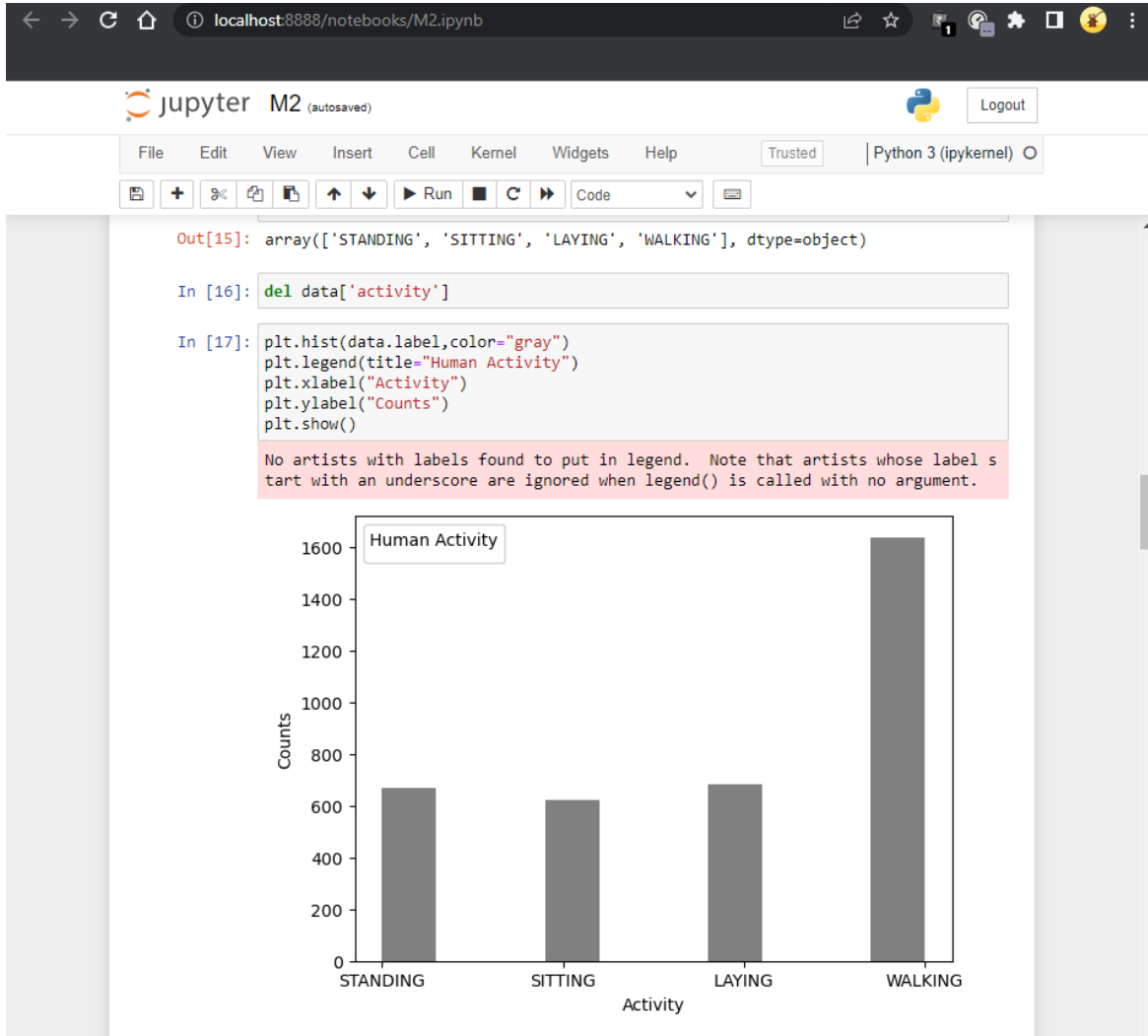


***Fig4.2.1.4:WorkflowofDatapre-processing***

#### **4.2.2 DataValidation/Cleaning/PreparingProcess:**

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held backfromtraining your model that is used to give an estimate of model skill while tuning model's and proceduresthatyoucanusetomakethebestuseofvalidationandtestdatasetswhen evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primarygoalofdatacleaningistodetectandremoveerrorsandanomaliestoincrease the value of data in analytics and decision making.

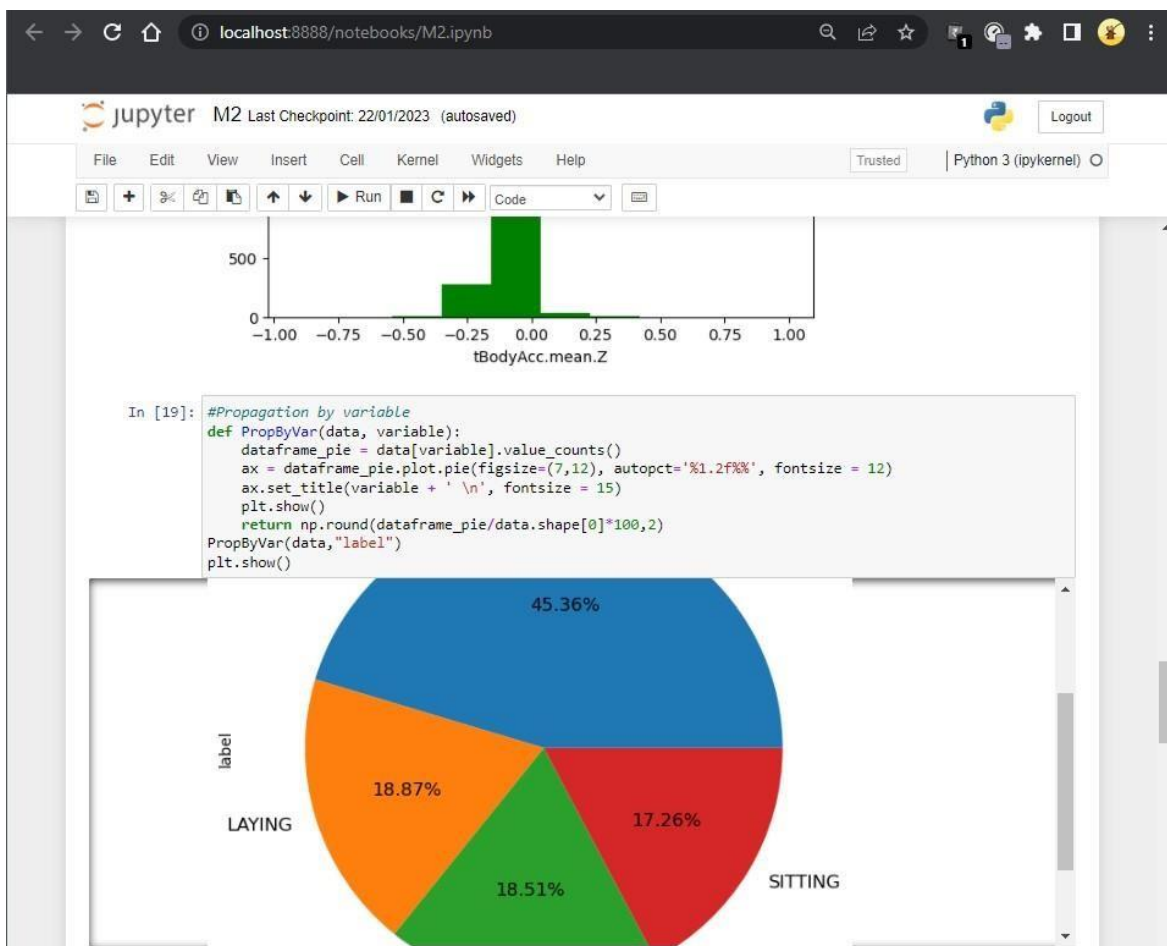
### 4.2.3 Exploration data analysis of visualization:



**Fig4.2.3.1: Data visualization**

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholder than measures of association or significance. Data visualization and exploratory data analysis are

whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.



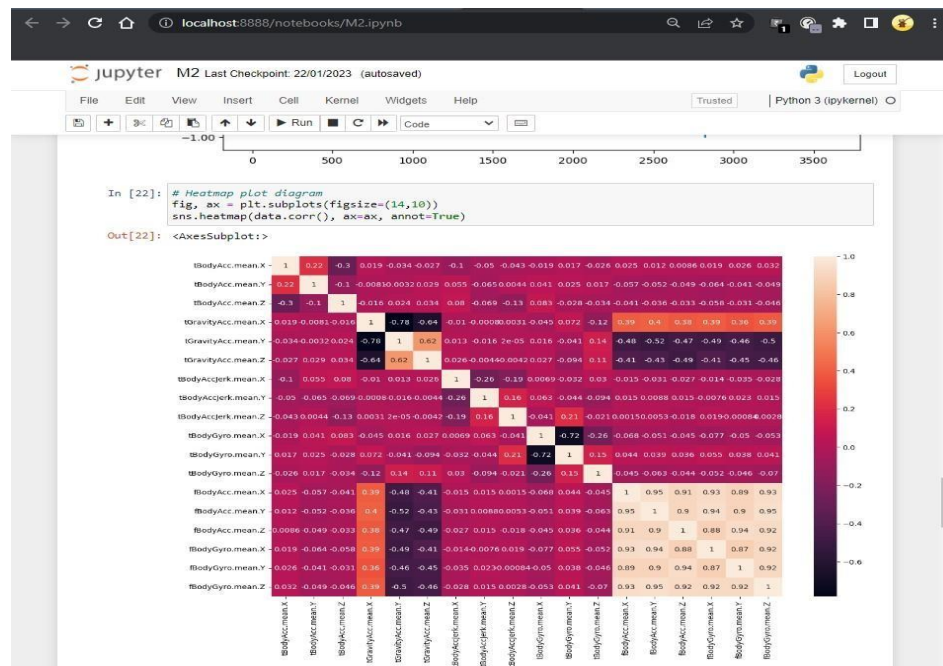
**Fig4.2.3.2: PiechartofData visualization**

Sometimes data does not make sense until it can look at a visual form, such as with charts and plots. Being able to quickly visualize data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.

- How to summarize data distributions with histograms and box plots. Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format which is not feasible for the analysis. To achieve better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset



**Fig4.2.3.3:HeatmapPlot Diagram**

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive. False Negatives (FN): A person whose default predicted as payer. When actual class is yes but predicted class is in

no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

**True Positives (TP):** A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

**True Negatives (TN):** A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing



***Fig4.2.3.4: Workflow of Data visualization***

#### **4.2.4 Comparing Algorithm with prediction in the form of best accuracy result:**

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and 58 different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization

methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 4 different algorithms are compared:

- LogisticRegression
- DecisionTree
- RandomForest
- SupportVector Machine

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same split to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to do one pre-processing, linear model 59 with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

#### **4.2.5 Prediction result by accuracy:**

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

TruePositiveRate(TPR) =  $TP / (TP + FN)$  False

Positive rate(FPR) =  $FP / (FP + TN)$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observations to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct. (When the model predicts default: how often is correct?)

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

GeneralFormula:

$$F\text{-Measure} = \frac{2TP}{2TP + FP + FN}$$

F1-ScoreFormula:

$$F1\text{Score} = \frac{2 * (\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

### **4.3 ALGORITHM AND TECHNIQUES:**

#### **4.3.1 Algorithm Explanation:**

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This dataset may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

#### **4.3.2 Used Python Packages:**

sklearn:

- In python, sklearn is a machine learning package which includes a lot of ML algorithms.
- Here, we are using some of its modules like `train_test_split`, `DecisionTreeClassifier` or `Logistic Regression` and `accuracy_score`.

NumPy:

- It is a numeric python module which provides fast math functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

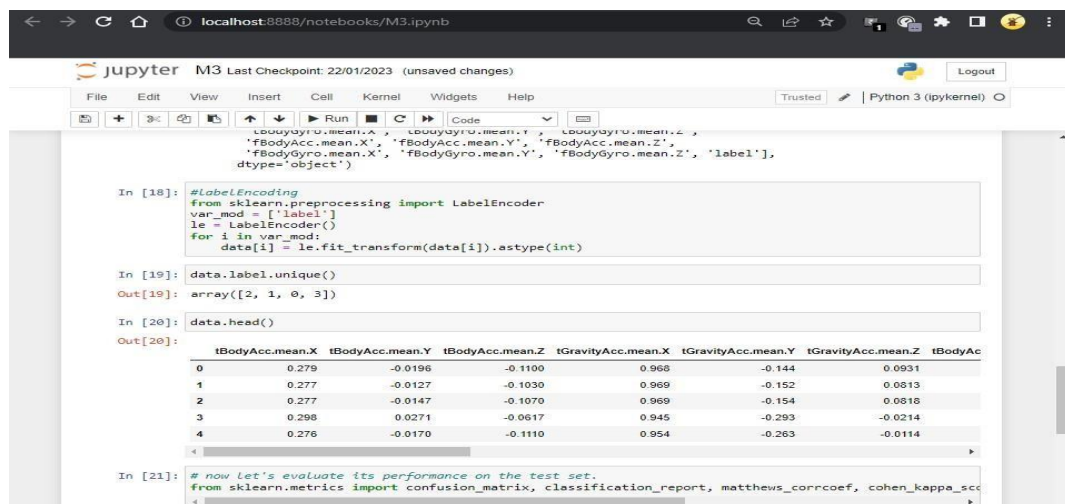
- Used to read and write different files.
- Data manipulation can be done easily with dataframes.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with dataframes.

### 4.3.3 Logistic Regression:

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (failure etc).



```
In [18]: #LabelEncoding
from sklearn.preprocessing import LabelEncoder
var_mod = ['label']
le = LabelEncoder()
for i in var_mod:
    data[i] = le.fit_transform(data[i]).astype(int)

In [19]: data.label.unique()
Out[19]: array([2, 1, 0, 3])

In [20]: data.head()
Out[20]:
```

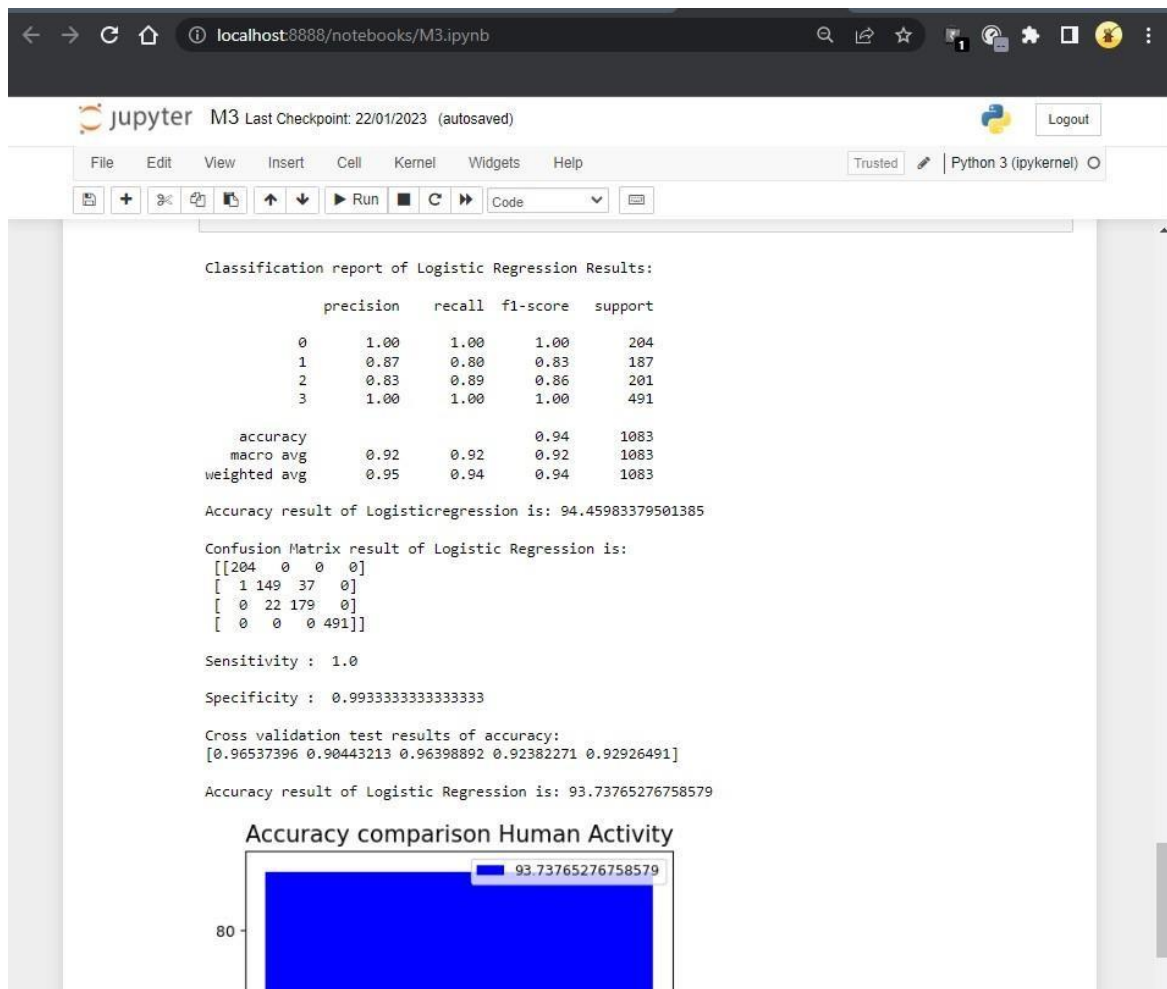
|   | tBodyAcc.mean.X | tBodyAcc.mean.Y | tBodyAcc.mean.Z | tGravityAcc.mean.X | tGravityAcc.mean.Y | tGravityAcc.mean.Z | tBodyAcc |
|---|-----------------|-----------------|-----------------|--------------------|--------------------|--------------------|----------|
| 0 | 0.279           | -0.0198         | -0.1100         | 0.968              | -0.144             | 0.0931             |          |
| 1 | 0.277           | -0.0127         | -0.1030         | 0.969              | -0.152             | 0.0813             |          |
| 2 | 0.277           | -0.0147         | -0.1070         | 0.969              | -0.154             | 0.0818             |          |
| 3 | 0.298           | 0.0271          | -0.0617         | 0.945              | -0.293             | -0.0214            |          |
| 4 | 0.276           | -0.0170         | -0.1110         | 0.954              | -0.263             | -0.0114            |          |

```
In [21]: # now Let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef, cohen_kappa_score
```

**Fig4.3.3.1:LabelEncoding**

In other words, the logistic regression model predicts  $P(Y=1)$  as a function of  $X$ .  
Logistic regression Assumptions:

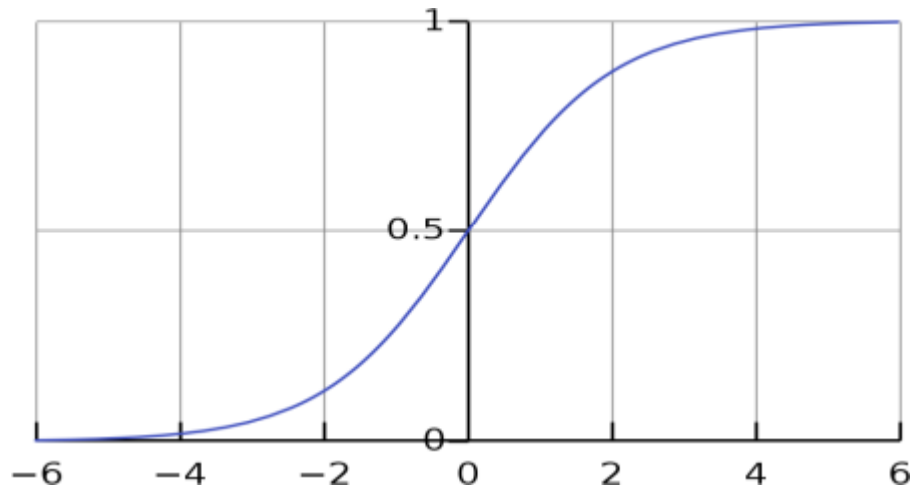
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



**Fig4.3.3.2: Report of logistic regression Result**

#### 4.3.4 Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.



**Fig4.3.4.:S-form Curve**

- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variables should not have multi-collinearity.

#### **4.3.5 Logistic Regression Equation:**

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n$$

- In Logistic Regression,  $y$  can be between 0 and 1 only, so for this let's divide the above equation by  $(1-y)$ :

$$\frac{y}{1-y}; 0 \text{ for } y=0, \text{ and infinity for } y=1$$

- but we need range between  $-\infty$  to  $+\infty$ , then take logarithm of the equation it will become:

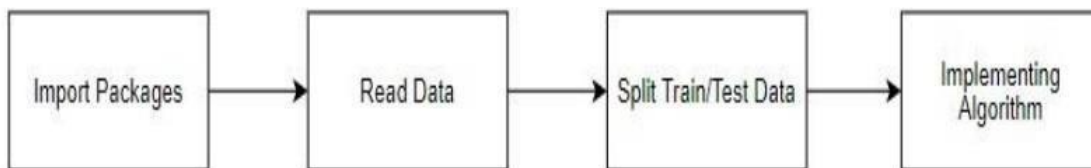
$$\log \left[ \frac{y}{1-y} \right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n$$

The above equation is the final equation for Logistic Regression. Type of

Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binary or Binomial**  
In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.
- **Multinomial**  
In such a kind of classification, dependent variable can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent "Type A" or "Type B" or "Type C".
- **Ordinal**  
In such a kind of classification, dependent variable can have 3 or more possible ordered types or the types having a quantitative significance. For example, these variables may represent "poor" or "good", "very good", "Excellent" and each category can have the scores like 0, 1, 2, 3



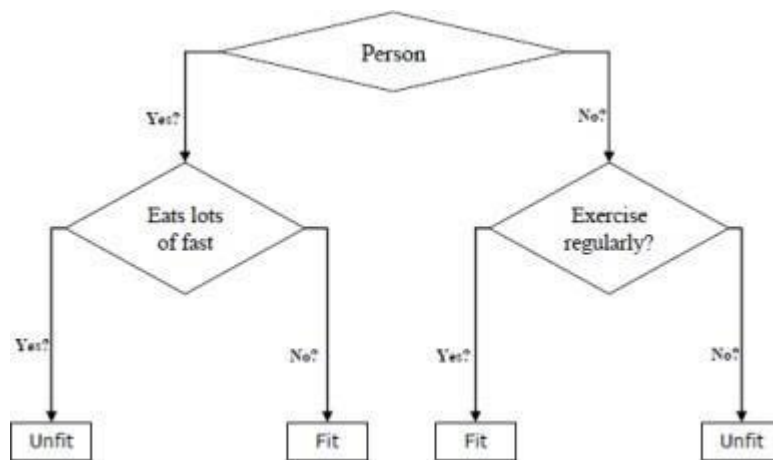
**Fig4.3.5: Workflow of Implementing Algorithm**

### 4.3.6 DecisionTree:

#### Introduction to Decision Tree

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decision trees are the most powerful algorithms that fall under the category of supervised algorithms.

They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we get the outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits, is given below –



**Fig4.3.6:DecisionTree**

In the above decision tree, the questions are decision nodes and final outcomes are leaves. We have the following two types of decision trees.

- Classification decision trees – In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.

- Regression decision trees – In this kind of decision trees, the decision variable is continuous.

## Implementing Decision Tree Algorithm Gini

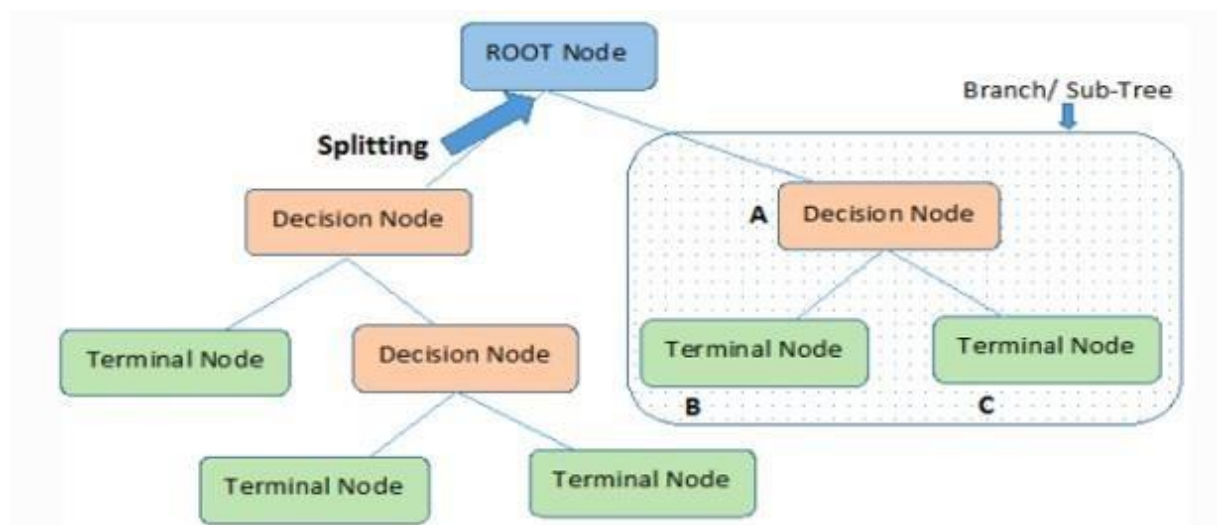
### Index

#### 4.3.6.1 Important Terminology related to Decision Tree:

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf/Terminal Node:** Nodes that do not split are called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.

**Branch/Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.

**Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.



**Fig4.3.6.1: Parent Node**

Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.

Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.

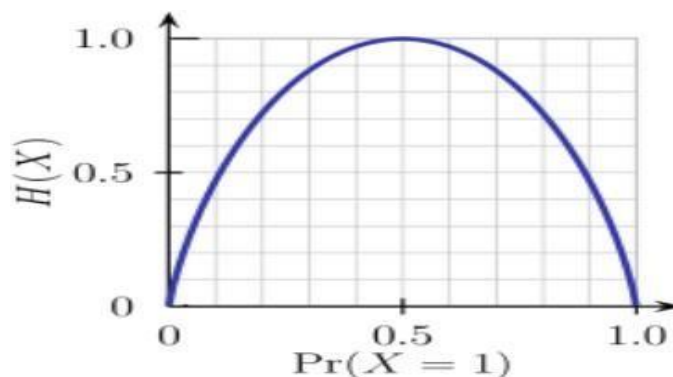
#### 4.3.6.2 Assumptions while creating Decision Tree:

Below are some of the assumptions we make while using Decision tree:

- In the beginning, the whole training set is considered as the root.
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- Records are distributed recursively on the basis of attribute values.
- Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

#### 4.3.7 Entropy:

Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random.



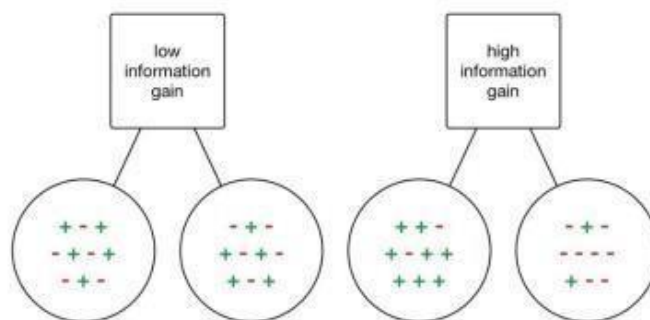
**Fig4.3.7.1: Graph**

From the above graph, it is quite evident that the entropy  $H(X)$  is zero when the probability is either 0 or 1. The Entropy is maximum when the probability is 0.5 because it projects perfect randomness in the data and there is no chance if perfectly determining the outcome. Mathematically Entropy for 1 attribute is represented as:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Where  $S \rightarrow$  Current state, and  $P_i \rightarrow$  Probability of an event  $i$  of state  $S$  or Percentage of class  $i$  in a node of state  $S$ .

Information gain or IG is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.



$$\text{Information Gain} = \text{Entropy}(\text{before}) - \sum_{j=1}^K \text{Entropy}(j, \text{after})$$

**Fig4.3.7.2: Information gain of Entropy**

Information gain is a decrease in entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain. Mathematically, IG is represented as:

$$\text{Information Gain}(T, X) = \text{Entropy}(T) - \text{Entropy}(T, X)$$

In a much simpler way, we can conclude that:

$$\text{Information Gain} = \text{Entropy}(\text{before}) - \sum_{j=1}^K \text{Entropy}(j, \text{after})$$

#### 4.3.8 Split Creation:

A split is basically including an attribute in the dataset and a value. We can create a split in a dataset with the help of the following three parts –

- Part 1: Calculating Gini Score – We have just discussed this part in the previous section.
- Part 2: Splitting a dataset – It may be defined as separating a dataset into two lists of rows having index of an attribute and a split value of that attribute. After getting the two groups – right and left, from the dataset, we can calculate the value of split by using Gini score calculated in first part. Split value will decide in which group the attribute will reside.
- Part 3: Evaluating all splits – Next part after finding Gini score and splitting dataset is the evaluation of all splits. For this purpose, first, we must check every value associated with each attribute as a candidate split. Then we need to find the best possible split by evaluating the cost of the split. The best split will be used as a node in the decision tree.

#### **4.3.9 Building a Tree:**

As we know that a tree has root node and terminal nodes. After creating the root node, we can build the tree by following two parts –

##### **Part 1: Terminal node**

creation While creating terminal nodes of decision tree, one important point is to decide when to stop growing tree or creating further terminal nodes. It can be done by using two criteria namely maximum tree depth and minimum node records as follows –

- **Maximum Tree Depth** – As name suggests, this is the maximum number of nodes in a tree after root node. We must stop adding terminal nodes once a tree reached at maximum depth i.e., once a tree got maximum number of terminal nodes. It may be defined as the minimum number of training patterns that a given node is responsible for. We must stop adding terminal nodes once tree reached at these minimum node records or below this minimum.

##### **Part 2: Recursive Splitting**

As we understood about when to create terminal nodes, now we can start building our tree. Recursive splitting is a method to build the tree. In this method, once a node is created, we can create the child nodes (nodes added to an existing node) recursively on each group of data, generated by splitting the dataset, by calling the same function again and again.

#### **4.3.10 Prediction:**

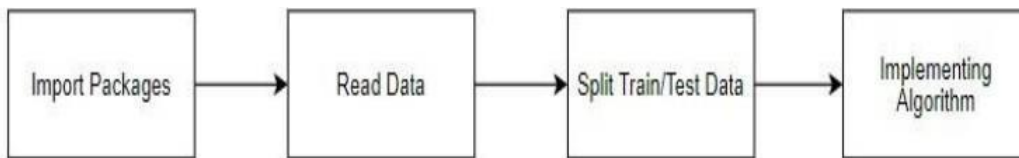
After building a decision tree, we need to make a prediction about it. Basically, prediction involves navigating the decision tree with the specifically provided row of data.

We can make a prediction with the help of recursive function, as did above. The same prediction routine is called again with the left or the child right nodes.

#### **4.3.11 Assumptions:**

The following are some of the assumptions we make while creating decision tree –

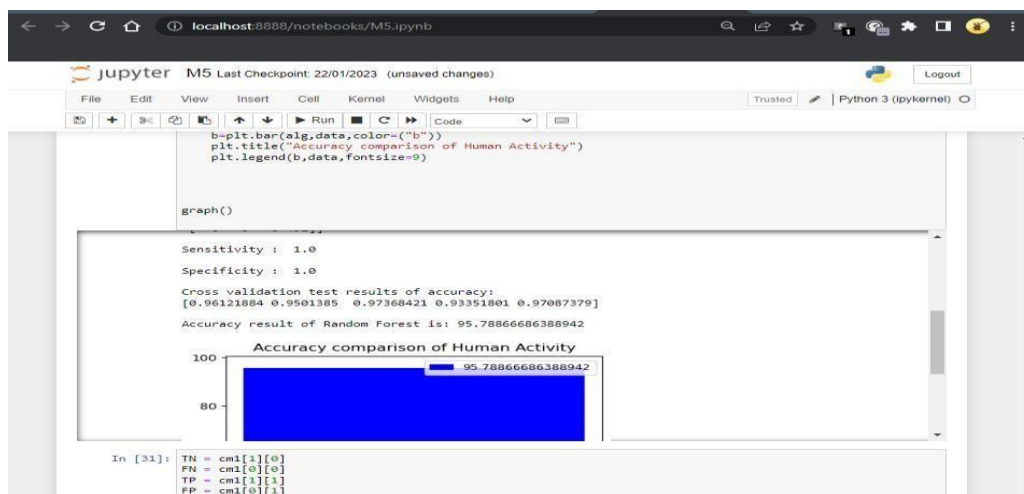
- While preparing decision trees, the training set is a root node.
- Decision tree classifier prefers the features values to be categorical. In case if you want to use continuous values then they must be done discretized prior to model building.
- Based on the attribute's values, the records are recursively distributed.
- Statistical approach will be used to place attributes at any node position i.e. as root node or internal node.



**Fig 4.3.9: Workflow of Decision Tree Algorithm**

#### 4.3.12 Random Forest Algorithm:

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 multiple classifiers to solve a complex problem and to improve the performance of the model.

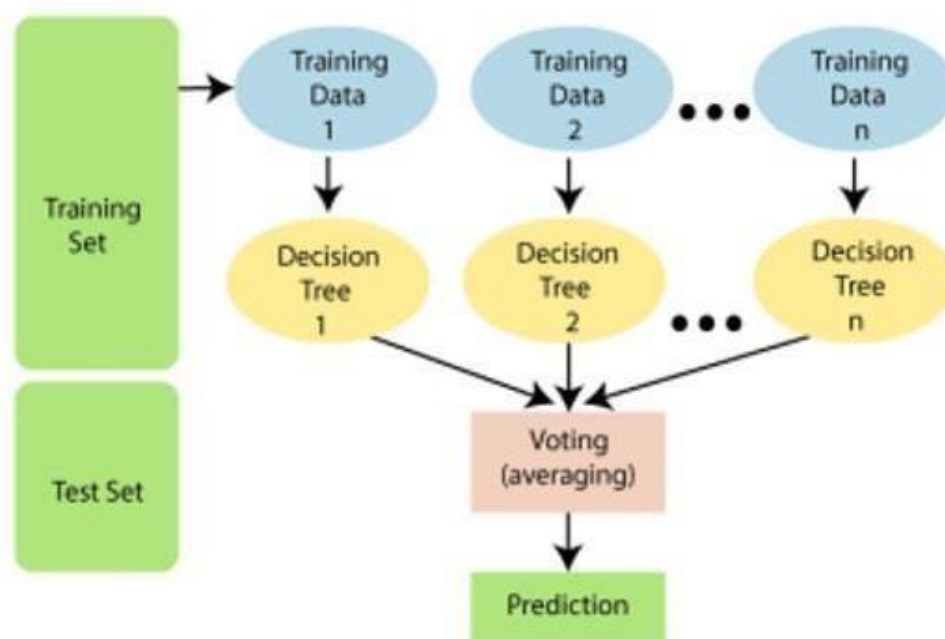


**Fig 4.3.12.1: Accuracy Comparison**

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 votes of 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.

The below diagram explains the working of the Random Forest algorithm:



***Fig4.3.12.2: Working of the Random Forest algorithm.***

#### 4.3.12.1 Assumptions for Random Forest:

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output.

- Therefore, below are two assumptions for a better Random forest classifier:  
There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result. 79
- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two phases: first is to create the random forest by combining  $N$  decision trees, and second is to make predictions for each tree created in the first phase.

The working process can be explained in the below steps and diagram: 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 new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:

#### 4.3.12.2 Implementation in Scikit-learn:

For each decision tree, Scikit-learn calculates a node's importance using Gini Importance, assuming only two child nodes (binary tree):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

- $ni_{sub}(j)$  = the importance of node  $j$
- $w_{sub}(j)$  = weighted number of samples reaching node  $j$
- $C_{sub}(j)$  = the impurity value of node  $j$
- $left(j)$  = child node from left split on node  $j$
- $right(j)$  = child node from right split on node  $j$

$sub()$  is being used as a subscript isn't available in Medium See

method `compute_feature_importances` in `_tree.pyx`

The importance for each feature on a decision tree is then calculated as:

$$fi_i = \frac{\sum_{j: \text{node } j \text{ splits on feature } i} ni_j}{\sum_{k \in \text{all nodes}} ni_k}$$

- $fi_{sub}(i)$  = the importance of feature  $i$
- $ni_{sub}(j)$  = the importance of node  $j$

These can then be normalized to a value between 0 and 1 by dividing by the sum of all feature importance values:

$$norm fi_i = \frac{fi_i}{\sum_{j \in \text{all features}} fi_j}$$

The final feature importance, at the Random Forest level, is its average over all the trees. The sum of the feature's importance value on each tree is calculated and divided by the total number of trees:

$$RFfi_i = \frac{\sum_{j \in \text{all trees}} \text{norm} fi_{ij}}{T}$$

- $RFfi_{sub}(i)$  = the importance of feature  $i$  calculated from all trees in the Random Forest model
- $\text{norm} fi_{sub}(ij)$  = the normalized feature importance for  $i$  in tree  $j$
- $T$  = total number of trees

See `method feature_importances_in_forest.py`.

Notation was inspired by this [Stack Exchange thread](#) which I found incredible useful for this post.

Implementation in Spark

For each decision tree, Spark calculates a feature's importance by summing the gain, scaled by the number of samples passing through the node:

$$fi_i = \sum_{j: \text{nodes } j \text{ splits on feature } i} s_j G_j$$

- $fi_{sub}(i)$  = the importance of feature  $i$
- $s_{sub}(j)$  = number of samples reaching node  $j$
- $C_{sub}(j)$  = the impurity value of node  $j$

See `method computeFeatureImportanceInTreeModels.scala`

To calculate the final feature importance at the Random Forest level, first the feature importance for each tree is normalized in relation to the tree:

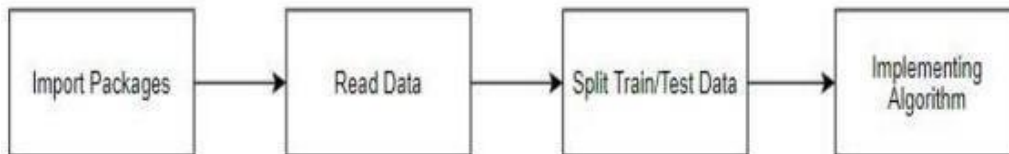
$$normfi_i = \frac{fi_i}{\sum_{j \in \text{all features}} fi_j}$$

- normfisub(i)=thenormalizedimportanceoffeaturei
- fisub(i)=theimportanceoffeaturei

Thenfeatureimportancevaluesfromeach treeare summednormalized:

$$RFfi_i = \frac{\sum_j normfi_{ij}}{\sum_{j \in \text{all features}, k \in \text{all trees}} normfi_{jk}}$$

- RFfi sub(i)= theimportance of featurei calculated fromall trees in the Random Forest model
- normfisub(ij)=thenormalizedfeatureimportanceforiintreej



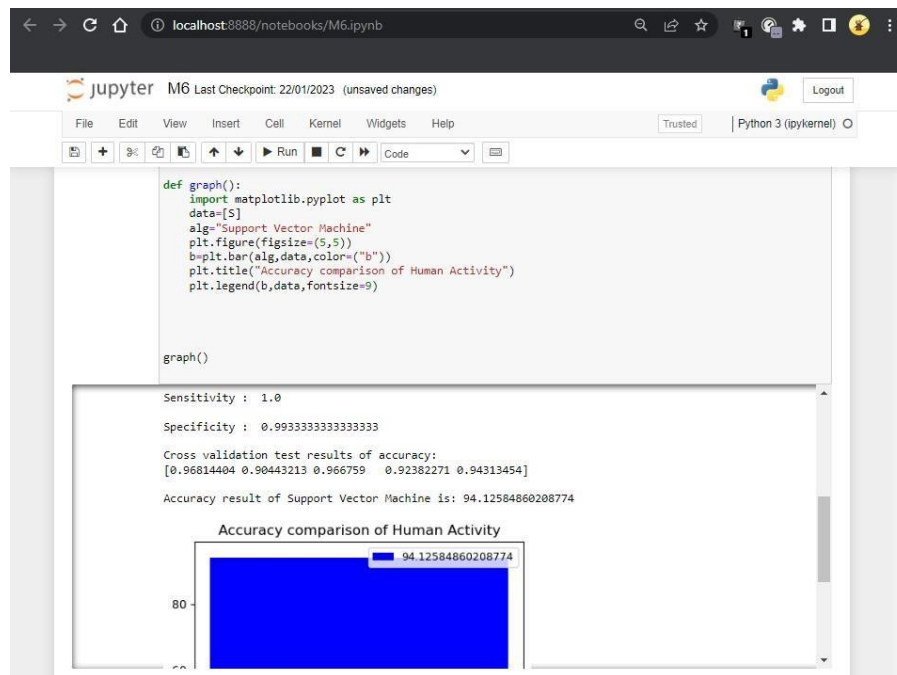
**Fig4.3.12.3:WorkflowofRandomForest Algorithm**

#### 4.3.12.4SupportVector Machines:

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. The objective of applying SVMs is to find the best line in two dimensions or the best hyperplane in more than two dimensions in order to help us separate our space into classes.

#### 4.3.12.4 Support Vector, Hyperplane, and Margin:

The vector points closest to the hyperplane are known as the **support vector points** because only these two points are contributing to the result of the algorithm, and other points are not. If a data point is not a support vector, removing it has no effect on the model. On the other hand, deleting the support vectors will then change the position of the hyperplane.



**Fig4.3.12.4:Support Vector Machines**

## CHAPTER – V

### RESULTS AND DISCUSSION

#### 5.1 DEPLOY:

##### **5.1.1 Flask(webframework):**

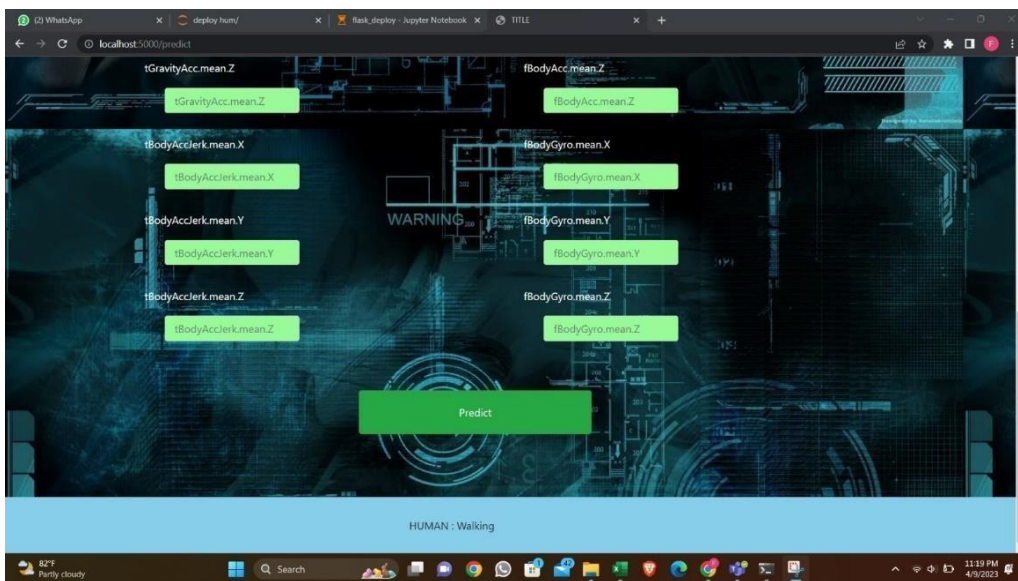
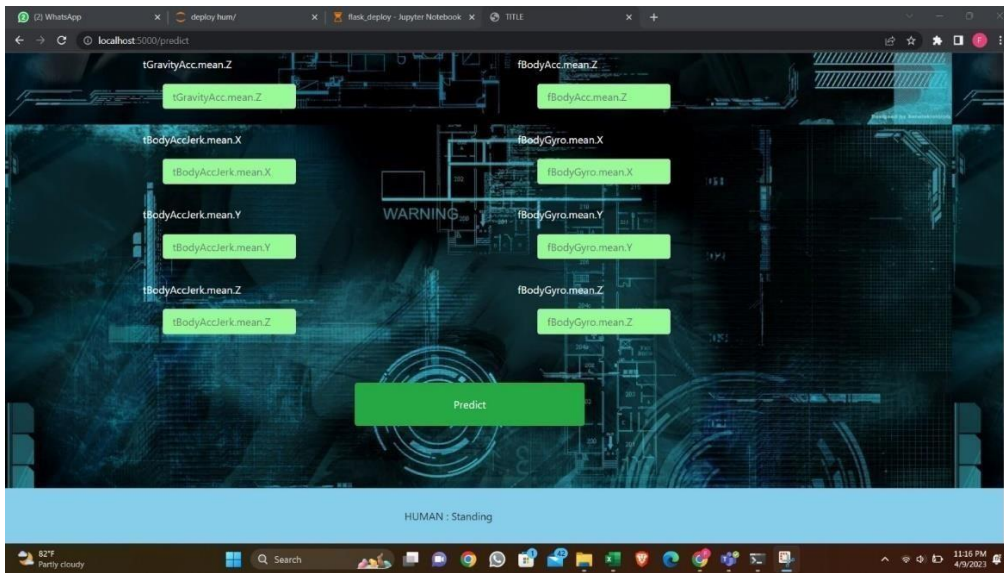
Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework. When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed. In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018. The micro-framework Flask is part of the Pallets Projects, and based on several others of them. Flask is based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python

webdevelopers. Let's take a closer look into Flask, so-called "micro" framework for Python



***Fig5.1:Results***

## CHAPTER – VI

### CONCLUSION AND FUTURE WORK

#### 6.1 CONCLUSION:

The applications of activity recognition with smartphones are numerous and include healthcare, fitness, sports, and security. For example, activity recognition can be used to monitor and analyze patients' movements to provide better healthcare, to track fitness levels and provide personalized workout plans, and to detect suspicious behavior and potential threats in security settings.

Overall, the recognition of human activities with smartphones using machine learning process is a promising area of research with potential applications in a wide range of industries. As technology continues to advance, it is likely that we will see more sophisticated and accurate methods of activity recognition that will further enhance its usefulness and applications.

#### 6.2 FUTURE WORK:

Human Activity Recognition connects with AI model. To automate this process by showing the prediction result in web application or desktop application. To optimize the work to implement in Artificial Intelligence environment. Improving the accuracy of activity recognition is one of the main challenges in activity recognition is achieving high accuracy, particularly for complex activities that involve multiple movements.

Using multiple sensors while smartphones are equipped with several sensors, including accelerometers, gyroscopes, and magnetometers, future research can explore the use of multiple sensors to improve the accuracy of activity recognition. Improving the accuracy of activity recognition, using multiple sensors, developing real-time activity recognition systems, addressing privacy and security concerns, and exploring new applications in different domains.

## REFERENCES:

- [1]. <https://doi.org/10.3389/frobt.2015.00028>. "A Review of Human Activity Recognition Methods" Michalis Vrigkas<sup>1</sup>, Christophoros Nikou<sup>1\*</sup> and Ioannis A. Kakadiaris<sup>2</sup>, Front. Robot. Reenacted knowledge, 16 November 2015.
- [2]. Imen Jegham, Anouar Ben Khalifa, Ihsan Alouani, Mohamed Ali Mahjoub, "Vision based human movement affirmation: A diagram and certifiable challenges", Forensic Science International: Digital Investigation, Volume 32, March 2020, 200901.
- [3]. Deep Learning Models for Human Activity Recognition by Jason Brownlee on September 26, 2018, in Deep Learning for Time Series, Last Updated on August 5, 2019.
- [4]. Zehua Sun, Qihong Ke, Hossein Rahmani, Mohammed Bennamoun, Gang Wang, Jun Liu, "Human Action Recognition from Various Data Modalities: A Review", Submitted on 22 Dec 2020 (v1), last refreshed 23 Jul 2021 (this version, v4).
- [5]. Muhammad Attique Khan, Kashif Javed, Sajid Ali Khan, Tanzila Saba, Usman Habib, Junaid Ali Khan, and Aaqif Afzaal Abbasi, "Human activity acknowledgment utilizing a combination of multiview and profound elements: an application to video observation", Published: 14 March 2020.
- [6]. Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen, "A Comprehensive Survey of Vision-Based Human Action Recognition Methods", Received: 2 February 2019; Accepted: 25 February 2019; Published: 27 February 2019.
- [7]. Muhammad Attique Khan<sup>1</sup>, Majed Alhaisoni<sup>2</sup>, Ammar Armghan<sup>3</sup>, Fayadh Alenezi<sup>3</sup>, Usman Tariq<sup>4</sup>, Yunyoung Nam<sup>5,\*</sup>, Tallha Akram<sup>6</sup>, "Video Analytics Framework for Human Action Recognition", Issue distributed 06 May 2021.
- [8]. <https://doi.org/10.18280/ts.370105> "Human Action Recognition in Video Sequences Using Deep Belief Networks" by Mehrez Abdellaoui, Ali Douik. Distributed: 29 February 2020

[9].[https://www.geintrauah.org/framework/records/paperactionbank\\_iwbbio\\_final.pdf](https://www.geintrauah.org/framework/records/paperactionbank_iwbbio_final.pdf),GEI  
NTRA Group, University of Alcala.

[10]. Human Action Recognition using Detectron2 and LSTM, Bibin Sebastian, JULY  
26, 2021.