

HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES USING MACHINE LEARNING PROCESS

Submitted in partial fulfillment of the requirements for the award
of
Bachelor of Engineering degree in Computer Science and Engineering

By

Muddana Dinesh Babu (Reg. No. 38110131)
Gadupudi Yashwanth (Reg. No. 38110148)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MUDDANA DINESH BABU** (38110131) and **GADUPUDI YASHWANTH** (38110148) who carried out the project entitled "**HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES USING MACHINE LEARNING PROCESS**" under my supervision from **October 2021** to **March 2022**.

Internal Guide

Mrs. Ramya G Franklin, M.E.,

Head of the Department

DR. VIGNESHWARI S, M.E., Ph. D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **MUDDANA DINESH BABU** (38110131) and **GADUPUDI YASHWANTH** (38110148) hereby declare that the Project Report entitled “**HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES USING MACHINE LEARNING PROCESS**” done by me under the guidance of Mrs. RAMYA G FRANKLIN (Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in **Computer Science and Engineering**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

I convey my thanks to **Dr.T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr.S.Vigneshwari M.E., Ph.D.**, and **Dr.L.Lakshmanan M.E., Ph.D.**, **Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs.RAMYA G FRANKLIN M.E.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Human activity recognition 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 classifies data into activities 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 supervised classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given transport traffic department dataset with evaluation. Dataset with 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.

TABLE OF CONTENTS

Chapter No.	Titile	Page no.
	Declaration	iii
	Acknowledgement	iv
	Abstract	v
	Table of Contents	vi
1.	INTROUCTION	1
	1.1 Outline	1
	1.2 Data Science	1
	1.3 Artificial Intelligence	2
	1.4 Machine Learning	4
	1.5 Objectives	12
	1.6 Project Goals	12
	1.7 Scope	13
	1.8 Problem Statement	14
2.	LITERATURE SURVEY	14
3.	METHODOLOGY	20
	3.1 PYTHON	20
	3.2 Existing System	33
	3.3 Proposed System	34
	3.4 Project Requirements	36
	3.4.1.Functional Requirements	36
	3.4.2.Non- Functional Requirements	37
	3.4.3.Enviromental Requirements	37
	3.5 Preparing The Data Set	38
	3.6 Software Description	39
	3.6.1.Anaconda Navigator	40
	3.6.2.Jupyter Notebook	44
	3.7 System Architecture	47
	3.8 Work Flow Diagram	48
	3.9 Use Case Diagram	49
	3.10 Class Diagram	50
	3.11 Activity Diagram	51
	3.12 Sequence Diagram	52

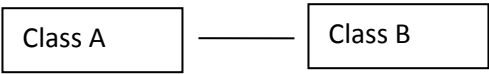
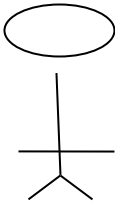
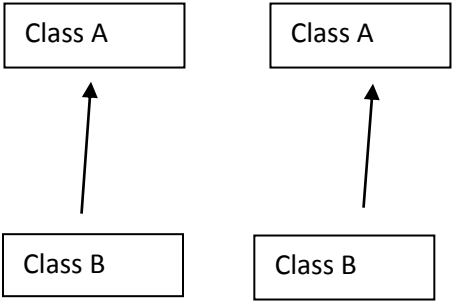
	3.13 Entity Relationship Diagram	53
4.	MODULES AND ALGORITMS	55
	4.1 List Of Modules	55
	4.2 Module Description	55
	4.3 Algorithm and Techniques	67
5.	RESULTS AND DISCUSSION	94
	5.1 Deploy Flask	94
	5.2 HTML Introduction	103
6.	CONCLUSION AND FUTURE WORK	117
	6.1 Conclusion	117
	6.2 Future Work	117
	REFERENCES	118
	APPENDICES	120
	A. Coding	120
	B. Screenshots	153
	C. Publication with Plagiarism Report	156

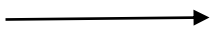

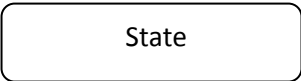
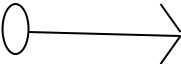
LIST OF FIGURES


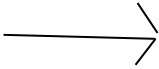
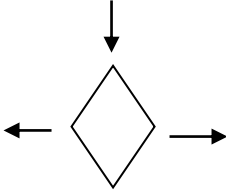
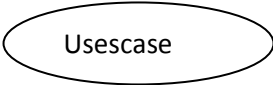
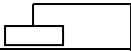
SL.NO	TITLE	PAGE.NO
01	SYSTEM ARCHITECTURE	47
02	WORKFLOW DIAGRAM	48
03	USECASE DIAGRAM	49
04	CLASS DIAGRAM	50
05	ACTIVITY DIAGRAM	51
06	SEQUENCE DIAGRAM	52
07	ER – DIAGRAM	53
08	MODULE DIAGRAM	92

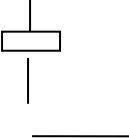
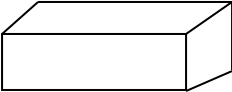
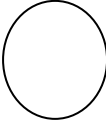

LIST OF SYSMBOLS


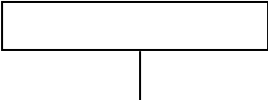
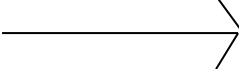
S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	<div><div><div>+ public</div><div>-private</div></div><div><div>Class Name</div><div>-attribute</div><div>-attribute</div></div></div>	Represents a collection of similar entities grouped together.
		<div>NAME</div> <div><div>Class A</div><div>Class B</div></div>	

2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the system and external environment

5.	<i>Relation</i> (uses)	<i>uses</i>	Used for additional process communication.
6.	Relation (extends)	EXTENDS 	Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the processs.
9.	Initial State		Initial state of the object

10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Usecase		Interaction between the system and external environment.
			

14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard, sensors, etc.
			Represents

18.	Transition		communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message	<p>Message</p> 	Represents the message exchanged.

CHAPTER 1

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 up stairs. 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.

Domain overview

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

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.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly

successful, helping to solve many challenging problems throughout industry and academia.

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 feed 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 in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It 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 modeling 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 data set 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.

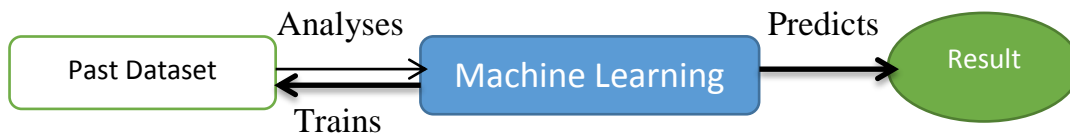


Fig: Process of Machine learning

Supervised Machine Learning **is the** majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output **is $y = f(X)$** . 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 labeled 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 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”.

Movements are often normal indoor activities such as standing, walking, laying sitting. 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

Human activities have been commonly used to define human behavioral 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
 - Loading the given dataset
 - Import required libraries packages
 - Analyze the general properties

- Find duplicate and missing values
- Checking unique and count values
- Uni-variate data analysis
 - Rename, add data and drop the data
 - To specify data type
- Exploration data analysis of bi-variate and multi-variate
 - Plot diagram of pairplot, heatmap, bar chart and Histogram
- Method of Outlier detection with feature engineering
 - Pre-processing the given dataset
 - Splitting the test and training dataset
 - Comparing the Decision tree and Logistic regression model and random forest etc
- Comparing algorithm to predict the result
 - 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 behavior of there 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 DESCRIPTION/ PROBLEM STATEMENTS

Now a days maximum of peoples are using smart phones, 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 in active like walking, running, sitting .Some people are not active in real life, those peoples are have obesity and some other health issues .We can use this also some security purpose and Transportation.

CHAPTER - 2

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 of them. Loan default trends have been long studied from a socio-economic stand point. Most economics surveys believe in empirical modeling 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 it is observing now. Some of the survey's to understand the past and present perspective of loan approval or not.

Review of Literature Survey

Title : A review of human activity Recognition methods

Author Michalis Vrigkas¹, Christophoros Nikou¹ and Ioannis A. Kakadiaris

Year : 16- November- 2016

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, view point, lighting, and appearance. Many applications, including video surveillance systems, human-computer interaction, and robotics for human behavior 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, we divide human activity classification methods in to two large categories according to whether they use data from different modalities or not. Then each of these categories is further analyzed into sub-categories, which reflect how they model human activities and what type of activities they are interested in. 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.

Title : A Survey on Activity Recognition and Behavior Understanding in Video Surveillance

Author : Sarvesh Vishwakarma· ,Anupam Agrawal

Year : · October 2012

This paper provides a comprehensive survey for activity recognition in video surveillance. 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 preprocessing (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.

Title : Trends over 5 Decades in U.S. Occupation-Related Physical Activity and Their Associations with Obesity

Author: P Timothy S. Church, Diana M. Thomas, Catrine Tudor-Locke , Peter T. Katzmarzyk, Conrad P. Earnest , Ruben Q. Rodarte , Corby K. Martin , Steven N. Blair , Claude Bouchard

Year : May 2011

The true causes of the obesity epidemic are not well understood and there are few longitudinal population based data published examining this issue. The objective of this analysis was to examine trends in occupational physical activity during the past 5 decades and explore how these trends relate to concurrent changes in body weight in the U.S. : Analysis of energy expenditure for occupations in U.S. 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. required at least moderate intensity physical activity whereas now less than 20% demand this level of energy expenditure. 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. Energy balance model predicted weights based on change in occupation-related daily energy expenditure 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. The results were similar for women. Over the last 50 years in the U.S. 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.

Title : Human detection in surveillance videos and its applications - a review

Author: Sarvesh Vishwakarma-Anupam Agrawal

Year : May 2011

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. 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 multi-view approach and adopting an improved model based on localized parts of the image.

Title : A Study of Vision based Human Motion Recognition and Analysis.

Author: Geetanjali Vinayak Kale, Varsha Hemant Patil

Year : 2 - July-December 2016

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 behavior understanding in surveillance system. 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. The paper emphasizes on human motion representation and the recognition methods along with their advantages and disadvantages. This study also discusses the 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 have shown success in speech and text recognition are successfully applied for action recognition. Advances in fields like Artificial Intelligence and Machine Learning needs to be applied for human motion recognition. Now, with efforts of research community in human motion recognition, products with intelligent camera systems for social and commercial applications should be made available in market.

CHAPTER -3

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.

History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

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 meta-programming 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 wide 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*

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.

Indentation:

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.

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.

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 equal 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=%s eggs=%d" % ("blah",2) evaluates to "spam=blah eggs=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:

- 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".
- 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.
- Raw string varieties, denoted by prefixing the string literal with an `r`. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.
- 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.

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 class to support native functions such as `print`, `length`, `comparison`, support for arithmetic operations, type conversion, and many more.

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, CPython. An experimental optional static type checker named `mypy` supports compile-time type checking.

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:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. 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 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.

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 approaches 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.

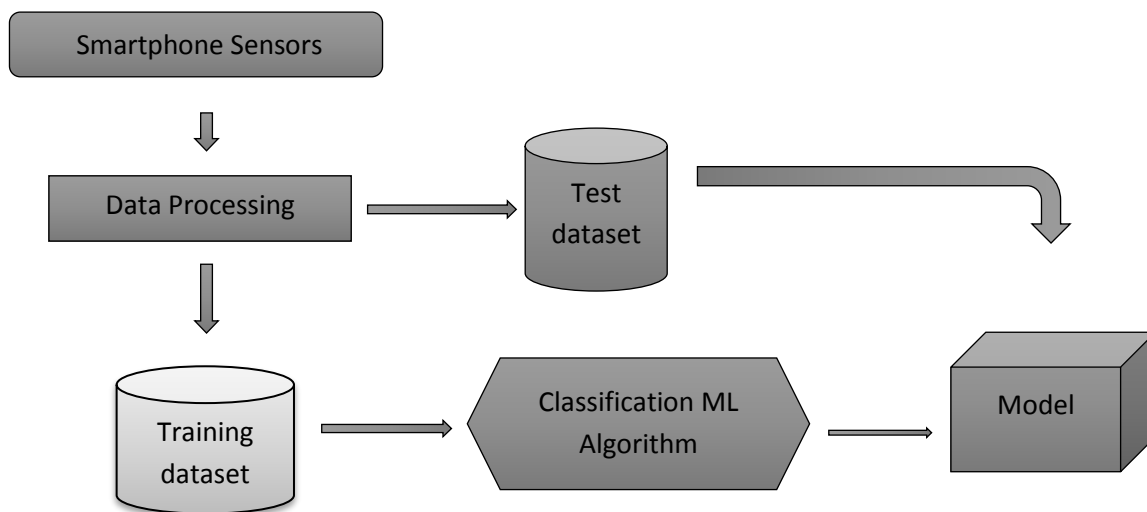


Fig: Architecture of Proposed model

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

General:

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. Prediction the result

3.4.3 Environmental Requirements:

1. Software Requirements:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

3.5 Preparing the Dataset

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 video-recorded 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 body acceleration and gravity. The 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.

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
Smart Phone	Samsung (Galaxy S II)
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

3.6 SOFTWARE DESCRIPTION

Anaconda is a free and open-source 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.

Anaconda. 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.

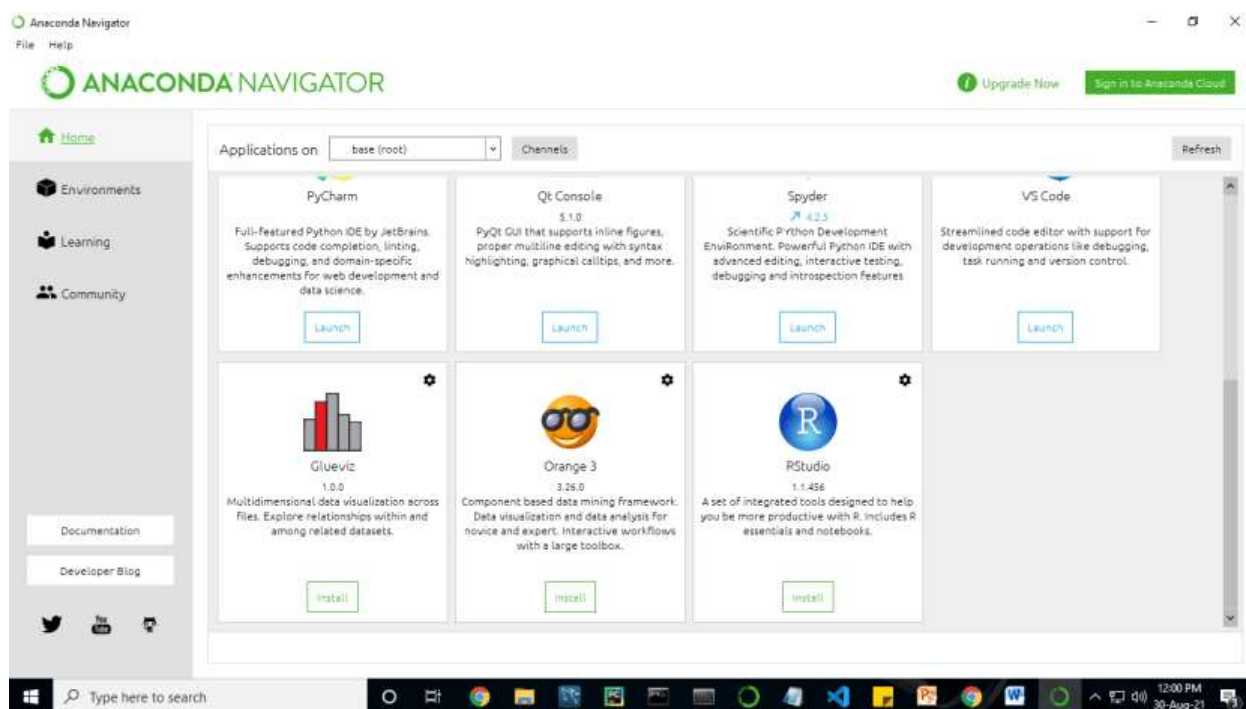
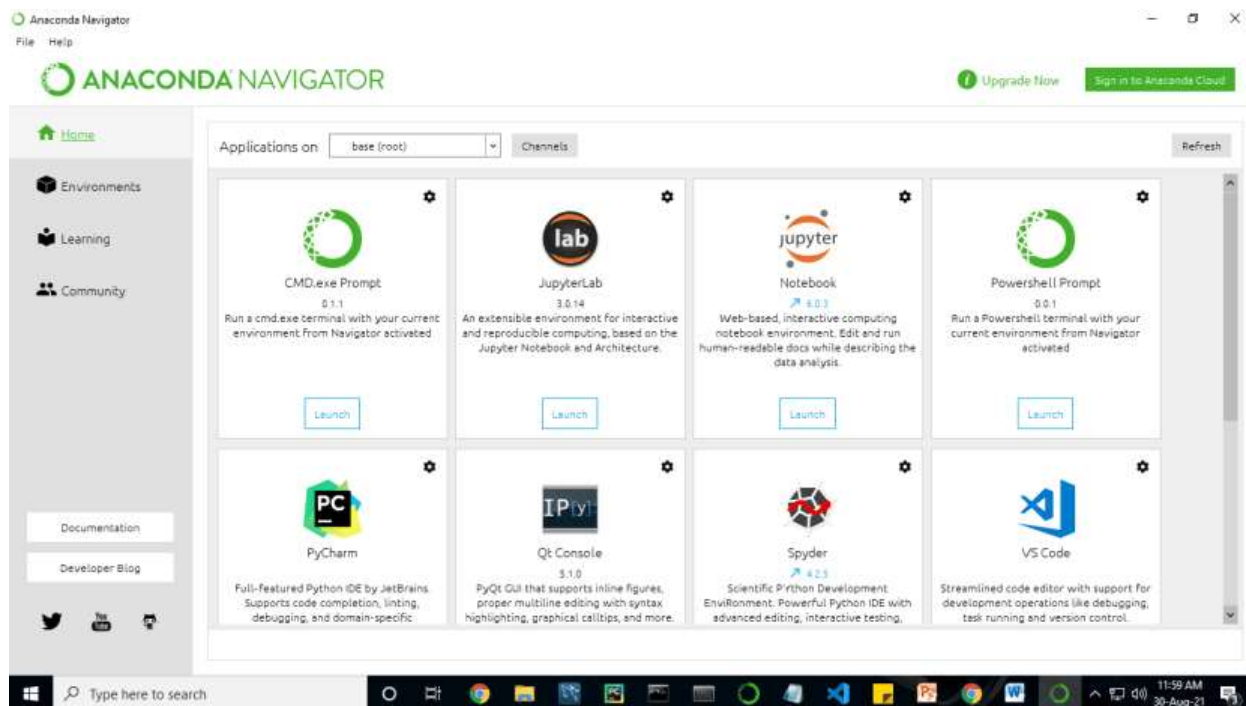
In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

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:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)



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 do not 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 softwares 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.

Installation: 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

Running the Jupyter Notebook

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.

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.

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.

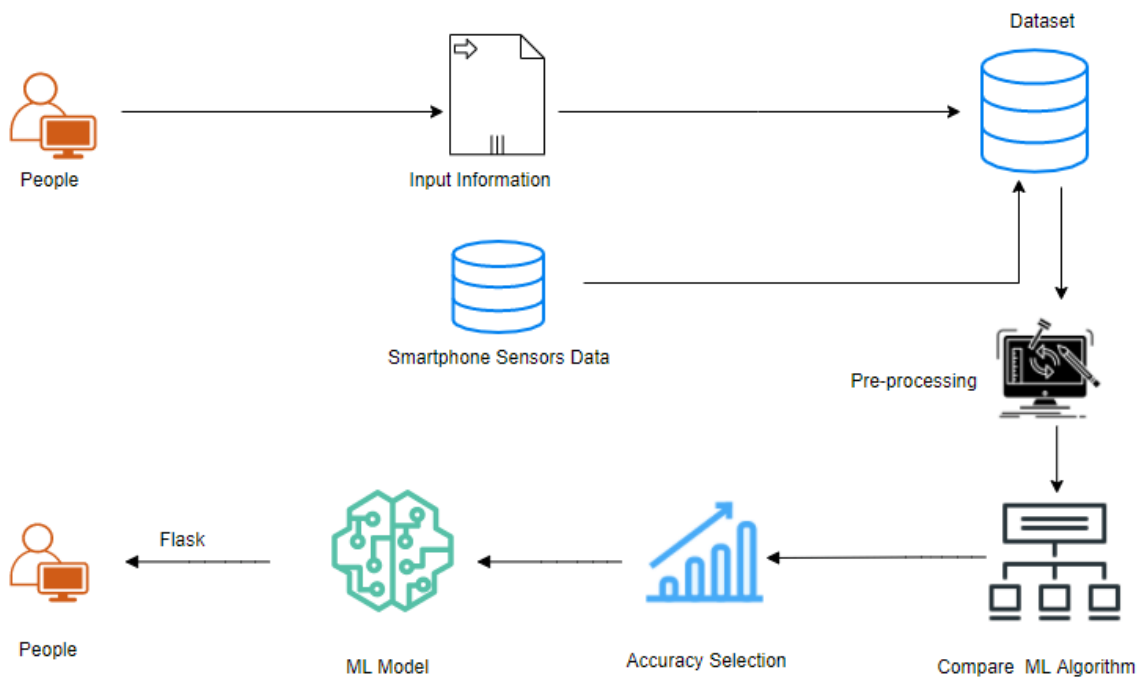
kernel: A notebook *kernel* is a “computational engine” that executes the code contained in a **Notebook document**. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

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

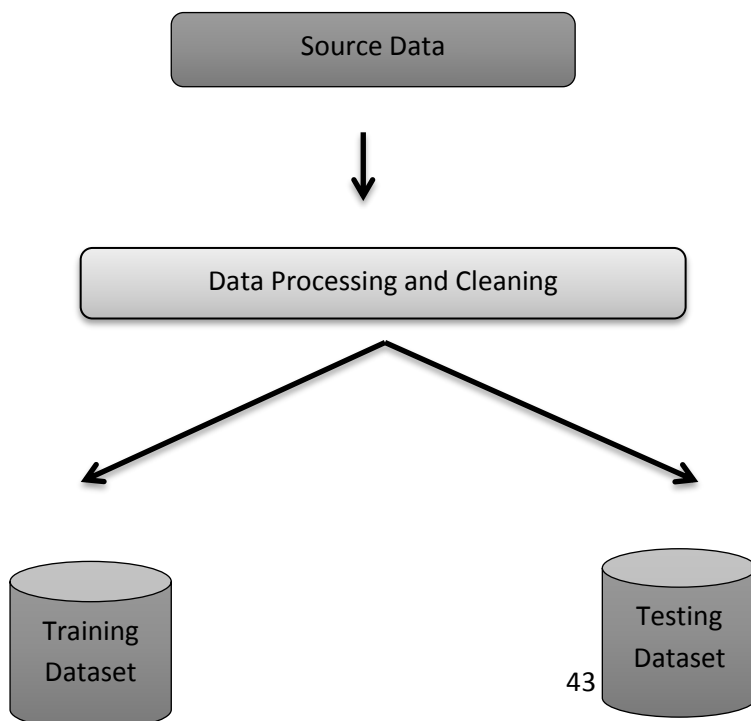
Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch **Jupyter Notebook App**. The *Notebook Dashboard* is mainly used to open **notebook documents**, and to manage the running **kernels** (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files.

3.7 SYSTEM ARCHITECTURE



3.8 WORK FLOW DIAGRAM



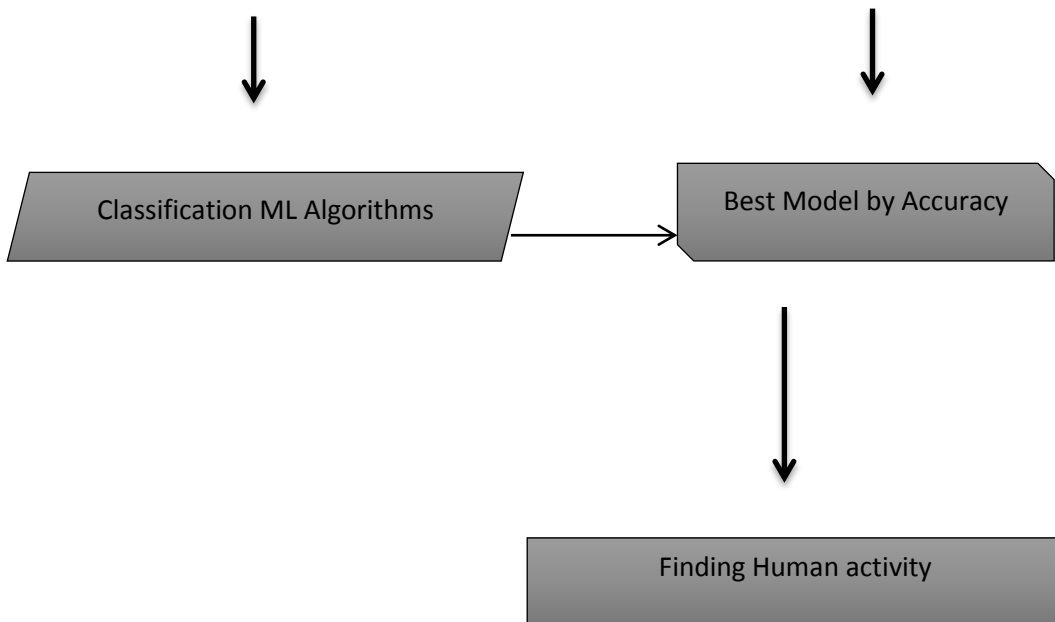
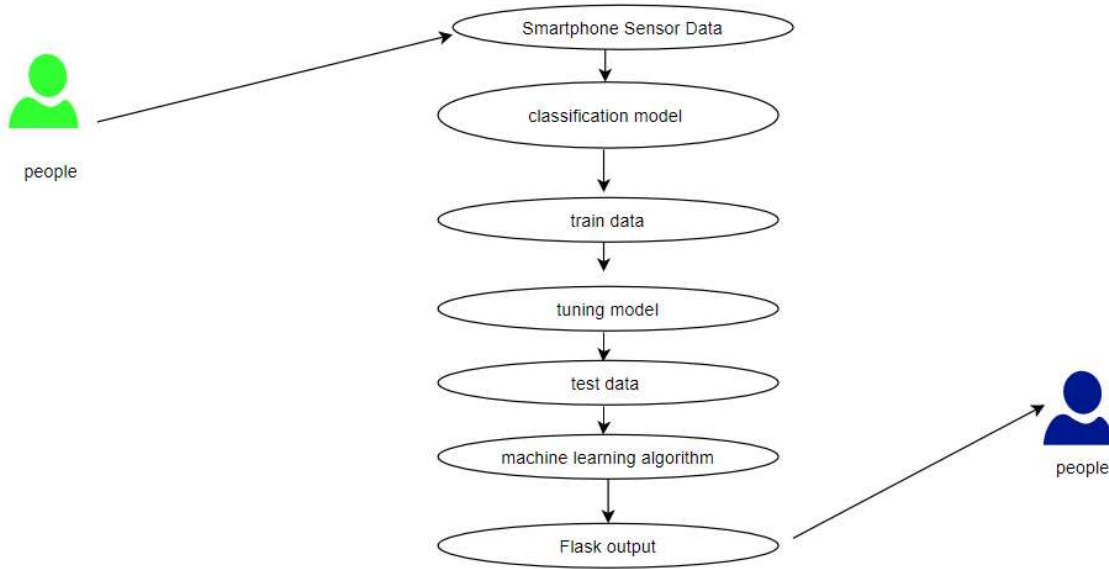


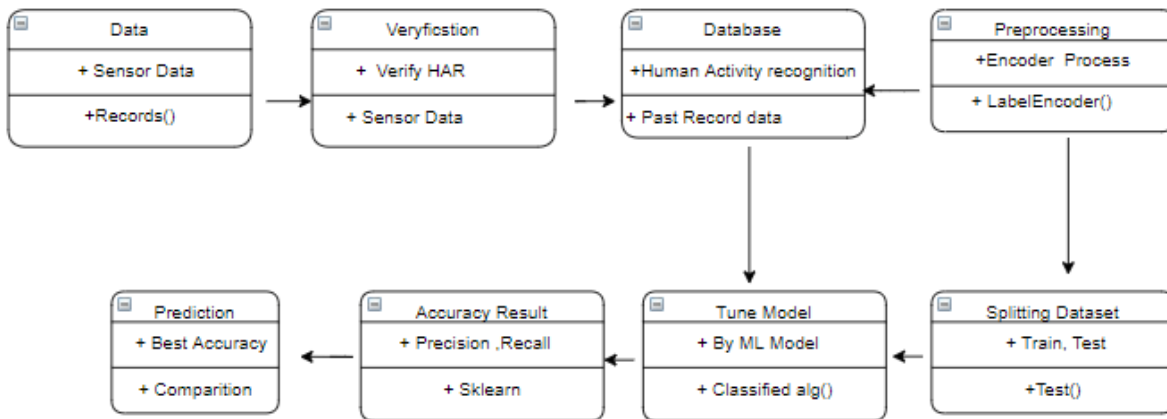
Fig: Workflow Diagram

3.9 USE CASE DIAGRAM



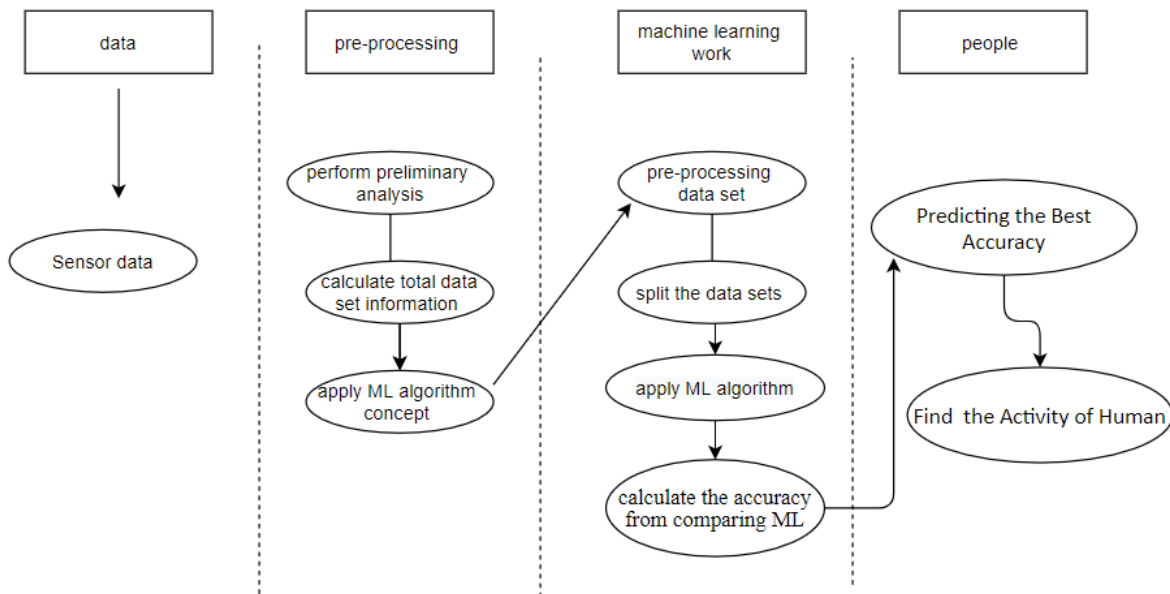
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

3.10 CLASS DIAGRAM:



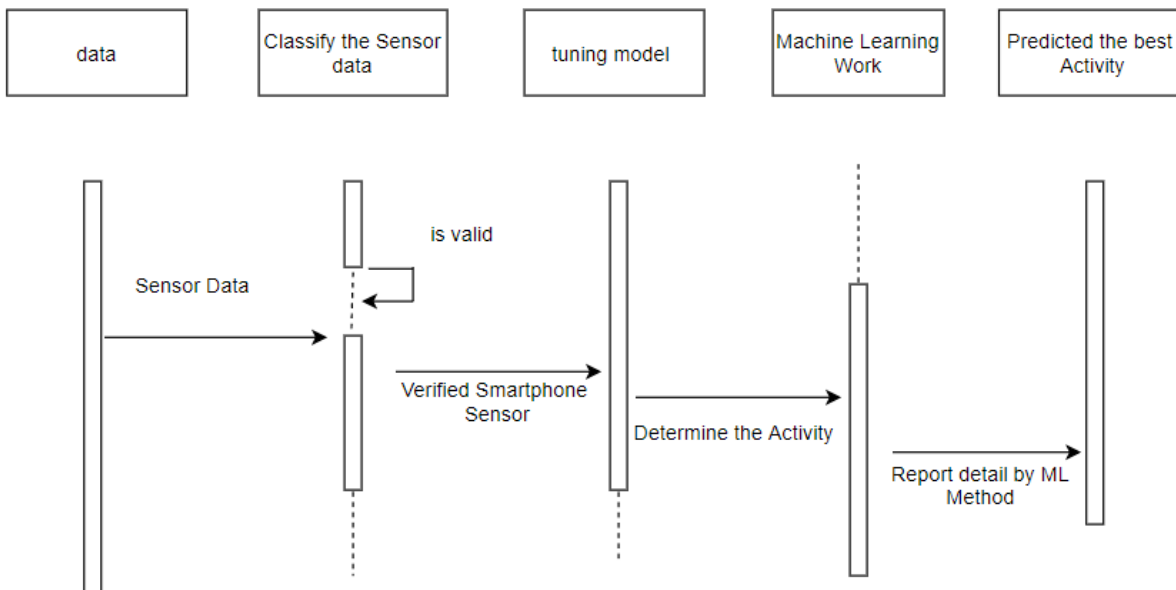
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element 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 ACTIVITY DIAGRAM:



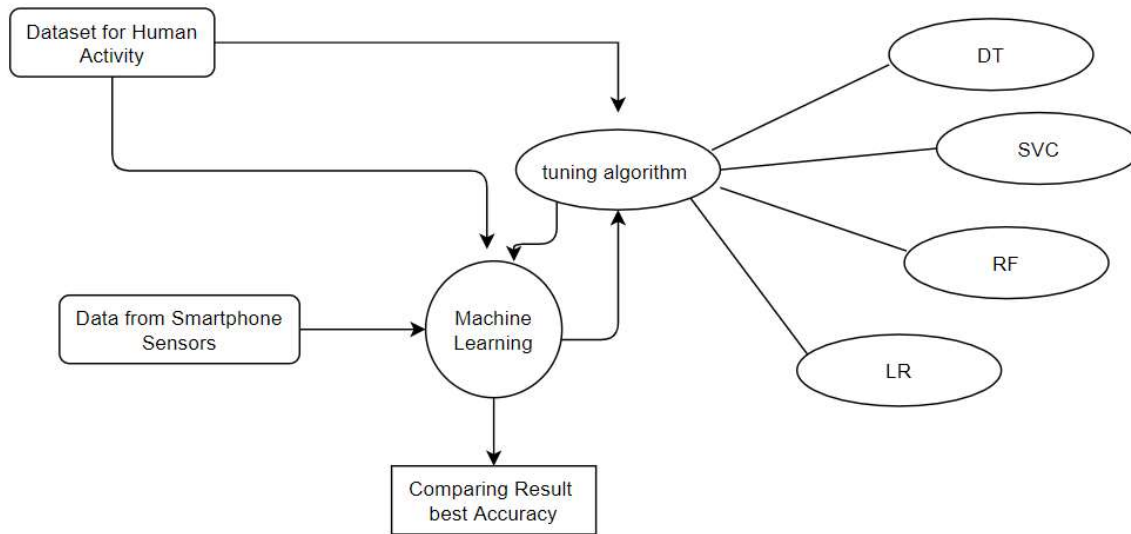
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. 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 looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

3.12 SEQUENCE DIAGRAM:



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. Sequence diagrams are the most popular 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 models are in my opinion the most important design-level models for modern business application development.

3.13 ENTITY RELATIONSHIP DIAGRAM (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 modeling 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 – 4

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:

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 running on a local host. The notebook is titled 'M1' and has a last checkpoint from 01/17/2022. The code in the first cell is `data`, which outputs a table of data. The table has 8 columns: `activity`, `tBodyAcc.mean.X`, `tBodyAcc.mean.Y`, `tBodyAcc.mean.Z`, `tGravityAcc.mean.X`, `tGravityAcc.mean.Y`, `tGravityAcc.mean.Z`, and `tBodyAcc.peak.X`. The data is organized into rows, with some rows grouped by activity. For example, the first group shows 'STANDING' activity with 5 rows of data. The second group shows 'WALKING_UPSTAIRS' activity with 5 rows of data. The table is scrollable, and the bottom of the notebook shows the command `data.head()` and the output of the first few rows.

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.

```

In [26]: data.describe()

Out[26]:
   |BodyAcc.mean.X |BodyAcc.mean.Y |BodyAcc.mean.Z |GravityAcc.mean.X |GravityAcc.mean.Y |GravityAcc.mean.Z |BodyAcc.jerk.mean.X |BodyAcc
|-----|-----|-----|-----|-----|-----|-----|-----|
|coord|3009.999999|3009.999999|3009.999999|3009.999999|3009.999999|3009.999999|3009.999999|
|mean|0.274544|-0.817415|-0.109195|0.678336|0.06752|0.887683|0.675124|
|std|0.963598|0.942588|0.956218|0.513473|0.38357|0.329519|0.179744|
|min|-0.521900|-1.000000|-0.928000|-0.847000|-1.000000|-1.000000|-1.000000|
|25%|0.252000|-0.825200|-0.122000|0.812000|-0.24400|-0.119000|0.060593|
|50%|0.277000|-0.817200|-0.109000|0.821000|-0.14500|0.238000|0.676000|
|75%|0.287000|-0.811000|-0.989000|0.959000|0.14500|0.264000|0.089996|
|max|0.883000|1.000000|1.990000|0.884000|1.06600|0.989000|1.000000|

In [27]: pd.Categorical(data['label']).describe()

Out[27]:
      counts  freq
categories
LAYING      687  0.188885
SITTING     623  0.172624
STANDING    688  0.185083

```

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.

In [25]: `data.corr()`

Out[25]:

	tBodyAcc.mean.X	tBodyAcc.mean.Y	tBodyAcc.mean.Z	tGravityAcc.mean.X	tGravityAcc.mean.Y	tGravityAcc.mean.Z	tBodyAcc.jerk.X	tBodyAcc.jerk.Y	tBodyAcc.jerk.Z	tBodyGyro.mean.X	tBodyGyro.mean.Y	tBodyGyro.mean.Z
tBodyAcc.mean.X	1.00000	0.216788	-0.360876	0.210888	-0.334890	-0.031793	-0.10					
tBodyAcc.mean.Y	0.216788	1.00000	-0.104570	-0.080098	-0.902186	0.020518	0.06					
tBodyAcc.mean.Z	-0.360876	-0.104570	1.00000	-0.415145	0.023714	0.033781	0.08					
tGravityAcc.mean.X	0.210888	-0.080098	-0.415145	1.00000	-0.753296	-0.635952	-0.01					
tGravityAcc.mean.Y	-0.334890	-0.902186	0.023714	-0.753296	1.00000	0.621930	0.01					
tGravityAcc.mean.Z	-0.031793	0.020518	0.033781	-0.635952	0.621930	1.00000	0.02					
tBodyAcc.jerk.X	-0.10	0.06	0.08	-0.01	0.01	0.02	1.00					
tBodyAcc.jerk.Y		-0.05211	-0.054652	-0.088905	-0.087768	-0.016387	-0.00426	1.00				
tBodyAcc.jerk.Z		-0.042641	0.004386	-0.127636	0.003071	0.000629	-0.004152	-0.19	1.00			
tBodyGyro.mean.X	-0.019215	0.041435	0.062522	-0.045221	0.011522	0.038901	0.00			1.00		
tBodyGyro.mean.Y	0.017324	0.025387	-0.028302	0.072417	-0.040654	-0.094261	-0.03				1.00	
tBodyGyro.mean.Z	-0.025678	0.016835	-0.034832	-0.122329	0.136888	0.198763	0.03					1.00
tBodyAcc.jerk.X	0.025423	-0.056623	-0.041486	0.388966	-0.475285	-0.412233	-0.01					
tBodyAcc.jerk.Y	0.012181	-0.051518	-0.035741	0.484707	-0.518887	-0.428033	-0.03					
tBodyAcc.jerk.Z	0.005962	-0.049471	-0.033065	0.370308	-0.468843	-0.499408	-0.02					
tBodyGyro.mean.X	0.019215	0.041435	0.062522	-0.045221	0.011522	0.038901	0.00					

- 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 data frame
 - show columns
 - shape of the data frame
 - To describe the data frame
 - Checking data type and information about dataset
 - Checking for duplicate data
 - Checking Missing values of data frame
 - Checking unique values of data frame
 - Checking count values of data frame
 - Rename and drop the given data frame
 - To specify the type of values

- To create extra columns

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

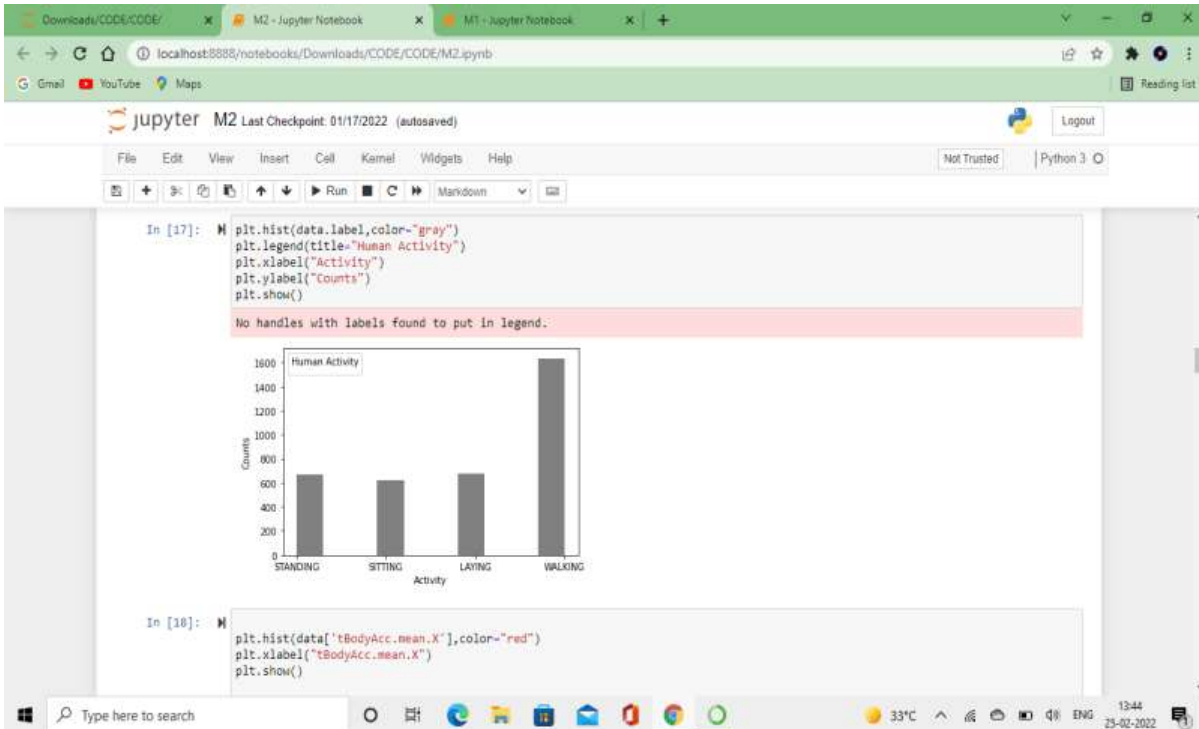
input : data

output : removing noisy data

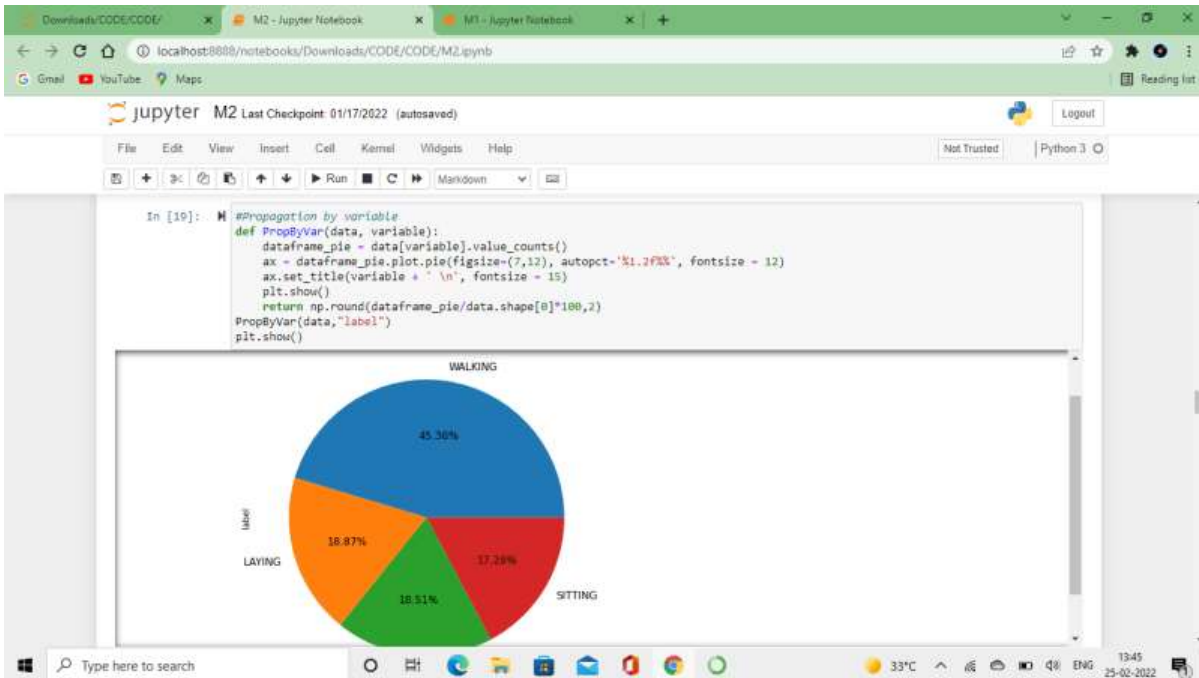
Data Validation/ Cleaning/Preparing Process

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 back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when 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 primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

Exploration data analysis of 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 stakeholders 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.

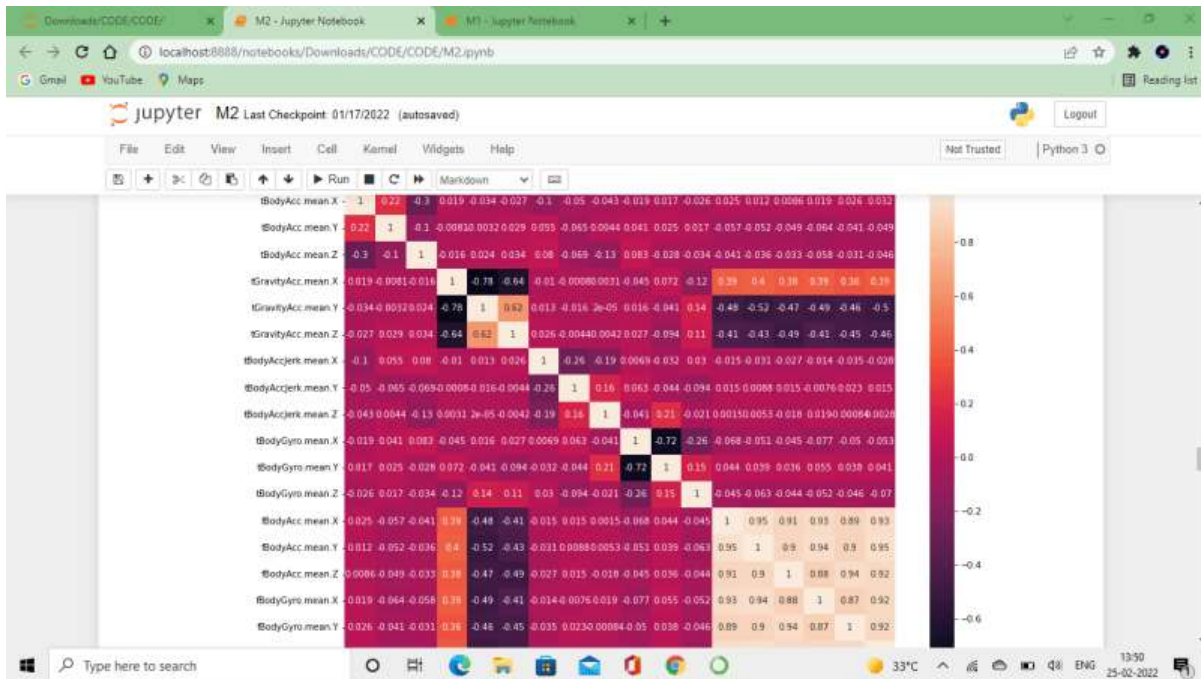


Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will 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 raw format which is not feasible for the analysis. To achieving 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.



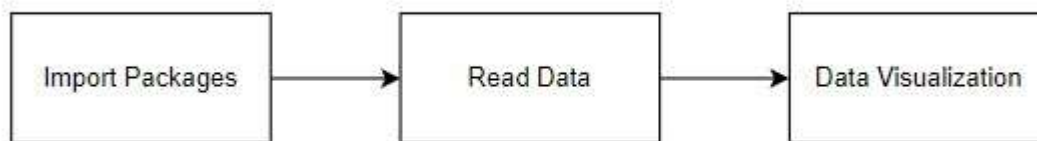
False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class 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.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

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

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:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector 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 splits 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 done preprocessing, linear model

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.

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.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

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 observation 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.

General Formula:

$$\text{F- Measure} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

F1-Score Formula:

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

4.3 ALGORITHM AND TECHNIQUES

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 data set 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.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include 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 maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

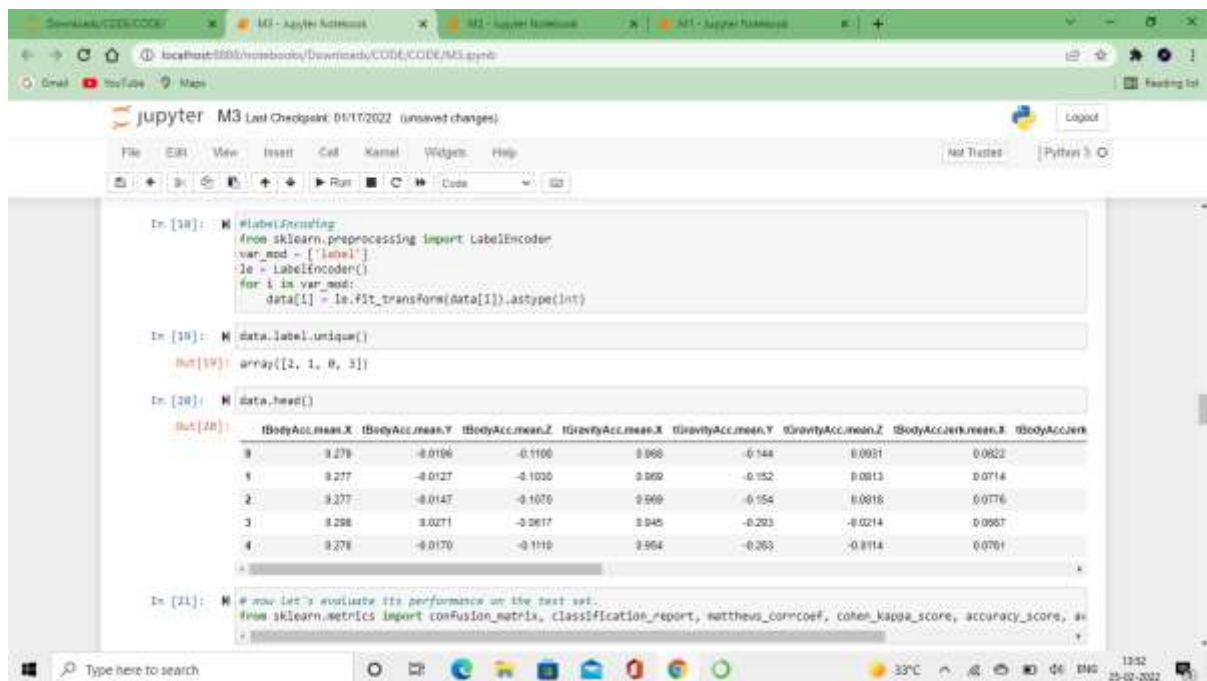
- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

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

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



```
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]:
```

	BodyAcc.mean.X	BodyAcc.mean.Y	BodyAcc.mean.Z	GravityAcc.mean.X	GravityAcc.mean.Y	GravityAcc.mean.Z	BodyAcc.mk.mean.X	BodyAcc.mk
0	0.278	-0.0106	-0.1100	0.960	-0.144	0.0021	0.0022	
1	0.277	-0.0127	-0.1030	0.960	-0.152	0.0013	0.0714	
2	0.277	-0.0147	-0.1070	0.960	-0.154	0.0016	0.0776	
3	0.298	0.0271	-0.0617	0.945	-0.203	-0.0214	0.0907	
4	0.278	-0.0170	-0.1110	0.964	-0.203	-0.0114	0.0701	

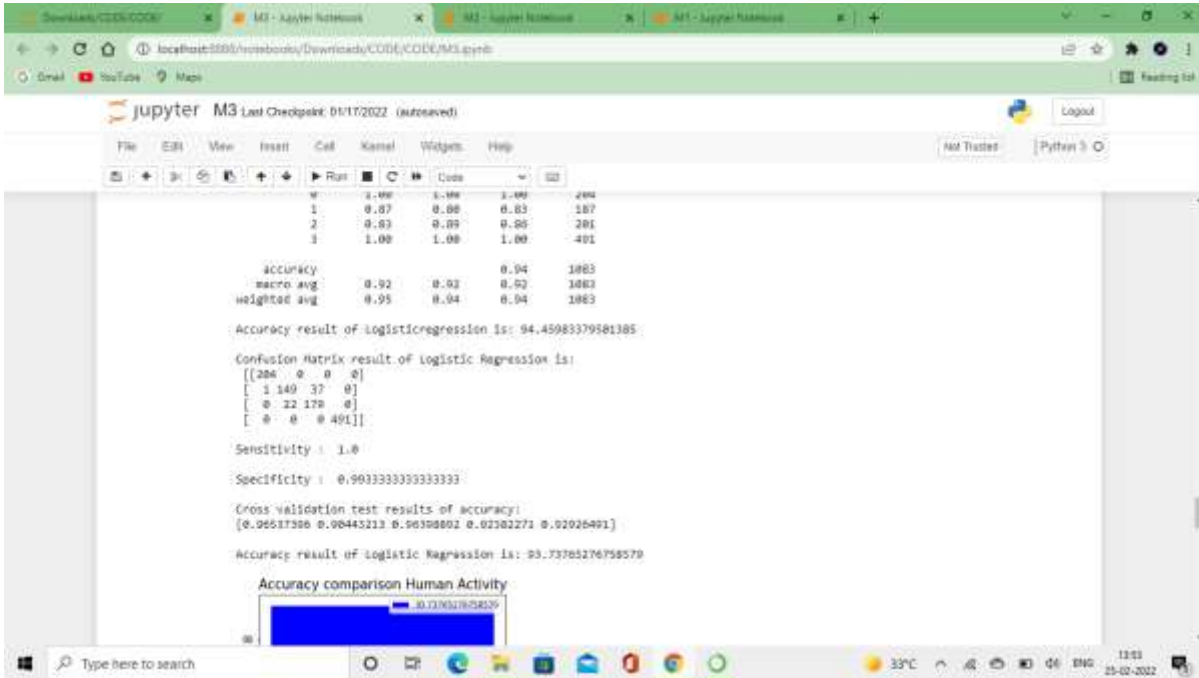
```
In [21]: # now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef, cohen_kappa_score, accuracy_score, au
```

failure etc).

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

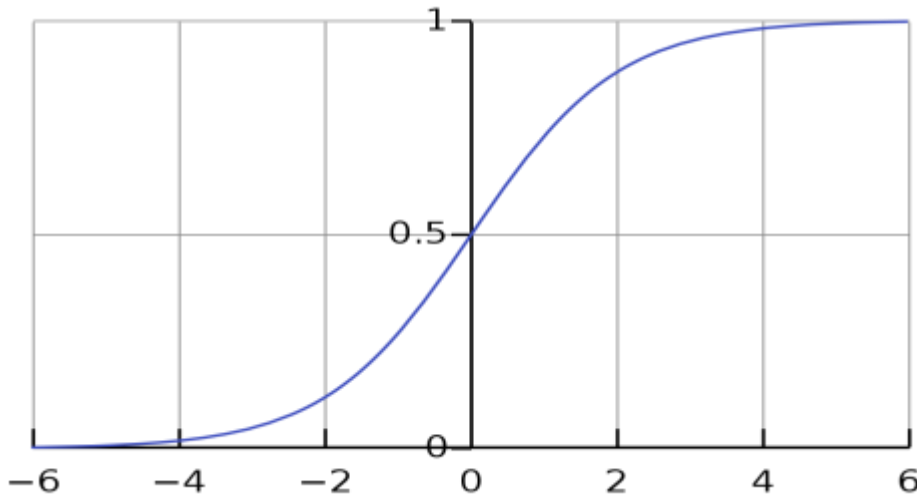
Logistic regression Assumptions:

- 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:



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.



$$1 / (1 + e^{-\text{value}})$$

- 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 variable should not have multi-collinearity.

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 $-\text{[infinity]}$ to $+\text{[infinity]}$, 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.

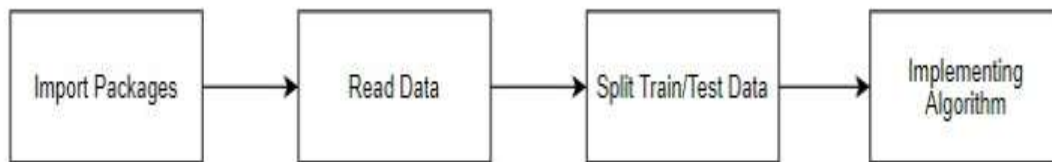
- **Multinomial**

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.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

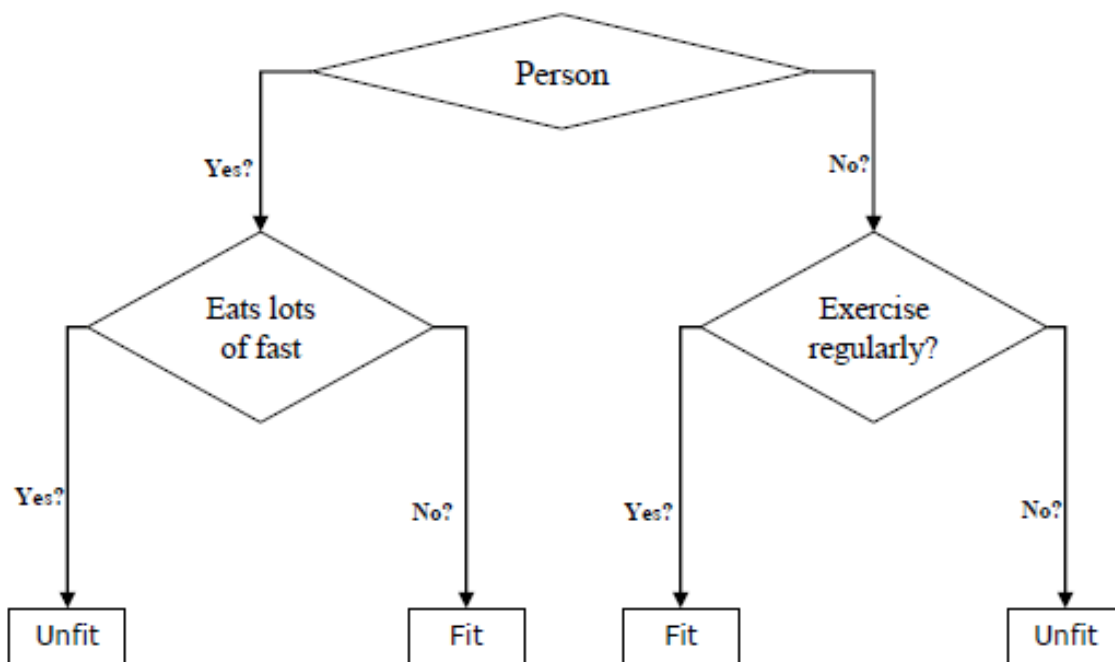
Decision Tree

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.

Decisions trees are the most powerful algorithms that falls 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 got 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 –



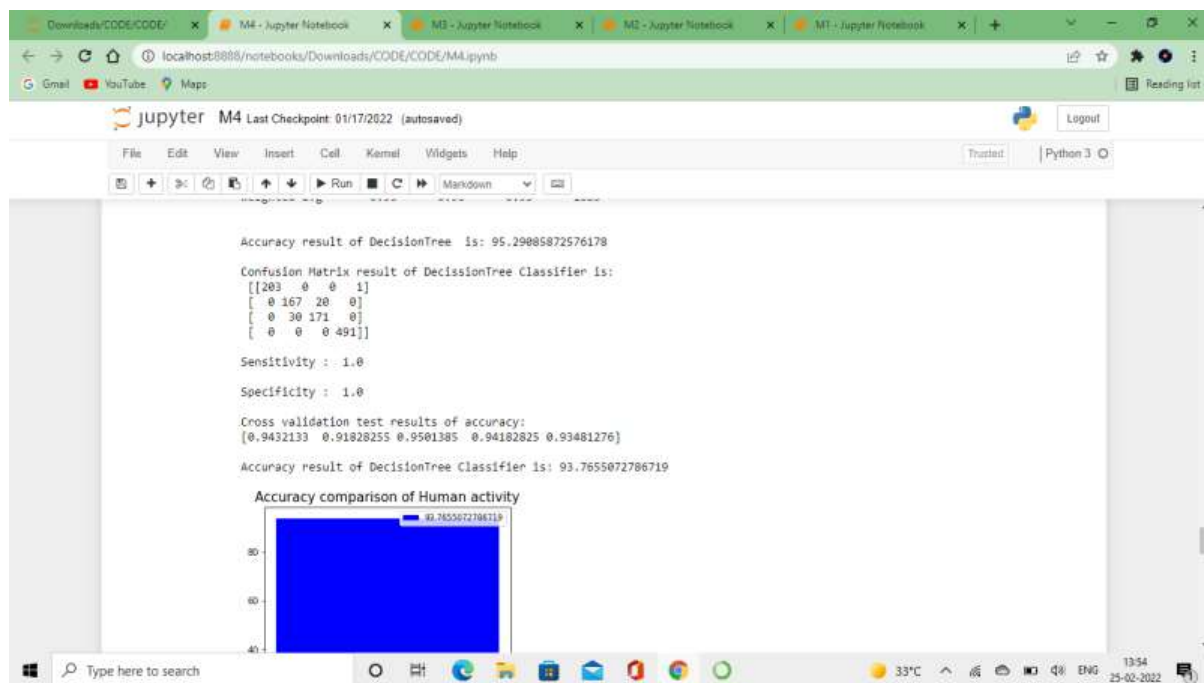
In the above decision tree, the question are decision nodes and final outcomes are leaves. We have the following two types of decision trees.

- **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

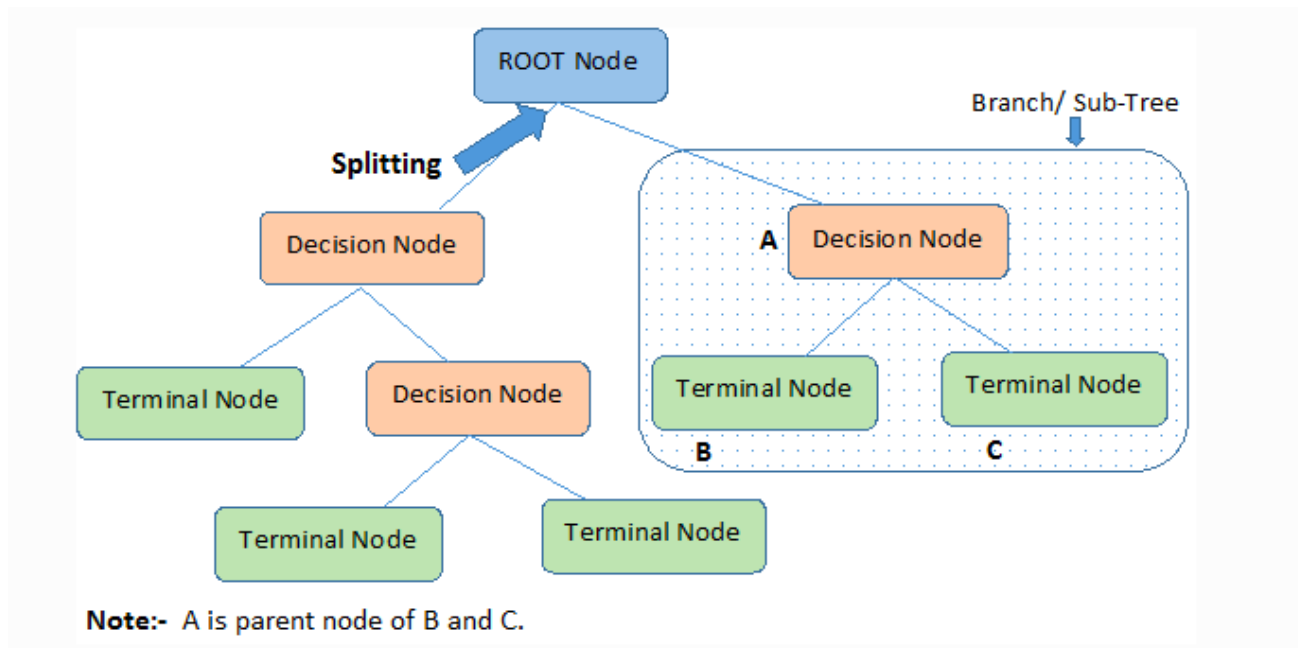
Gini Index

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 do not split is 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.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **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.



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.

Assumptions while creating Decision Tree

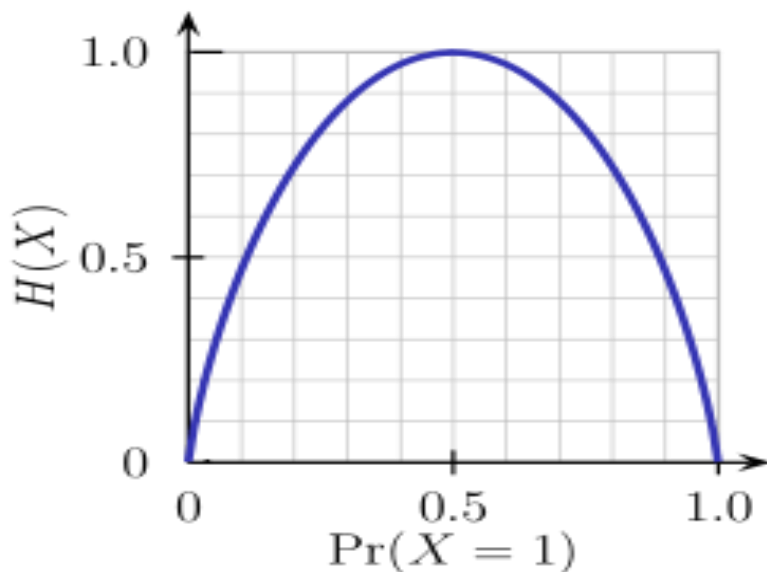
Below are some of the assumptions we make while using Decision tree:

- 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.

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.



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.

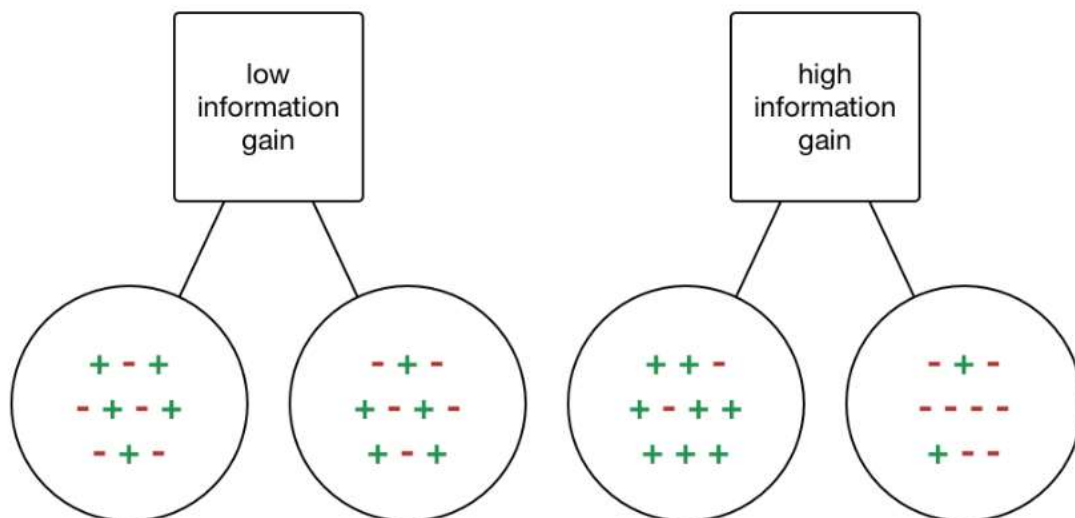
Mathematically Entropy for 1 attribute is represented as:

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

Where **S** → **Current state**, and **P_i** → **Probability of an event *i* of state S or Percentage of class *i* in a node of state S.**

Information Gain

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.



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

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:

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

In a much simpler way, we can conclude that:

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

Information Gain

Where “before” is the dataset before the split, K is the number of subsets generated by the split, and (j, after) is subset j after the split.

Split Creation

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

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 the 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.
- **Minimum Node Records** – 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.

Terminal node is used to make a final prediction.

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.

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.

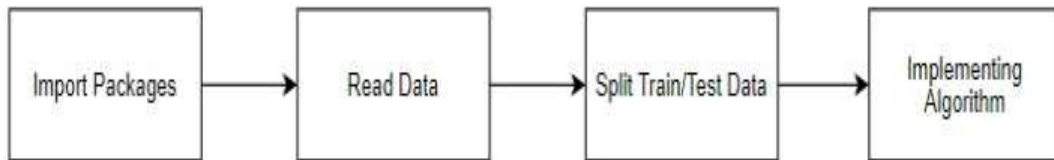
Assumptions

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

- While preparing decision trees, the training set is as 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.

•

MODULE DIAGRAM

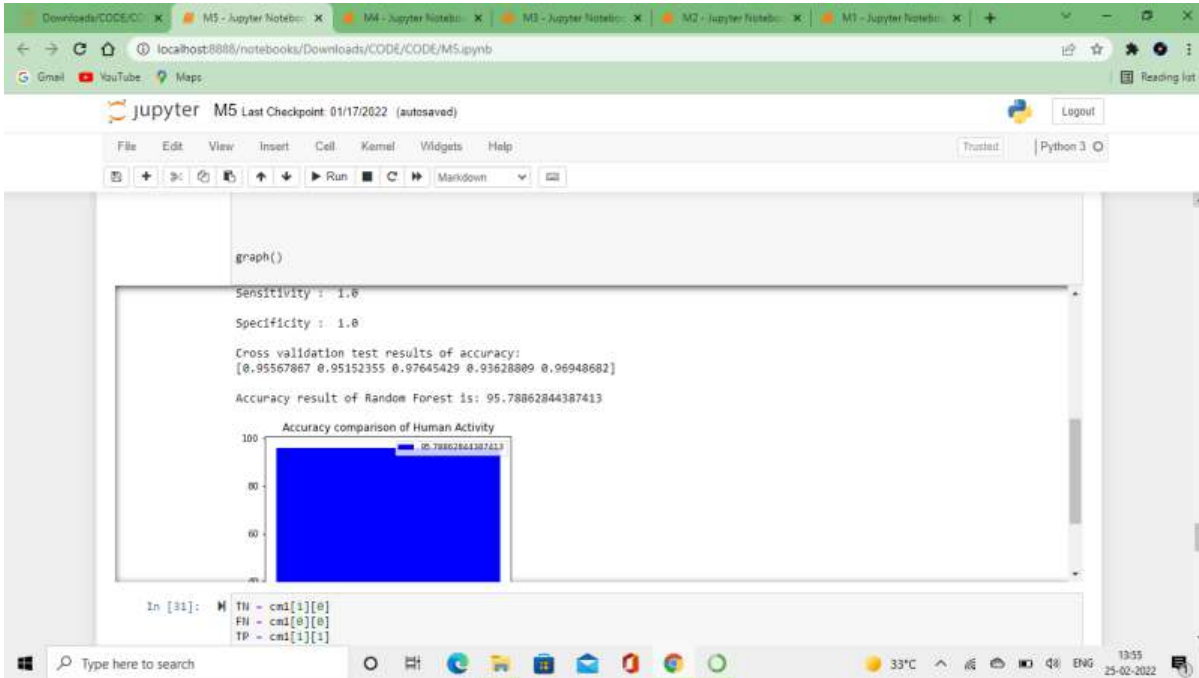


GIVEN INPUT EXPECTED OUTPUT

- input : data
- output : getting accuracy

Random forest Algorithm

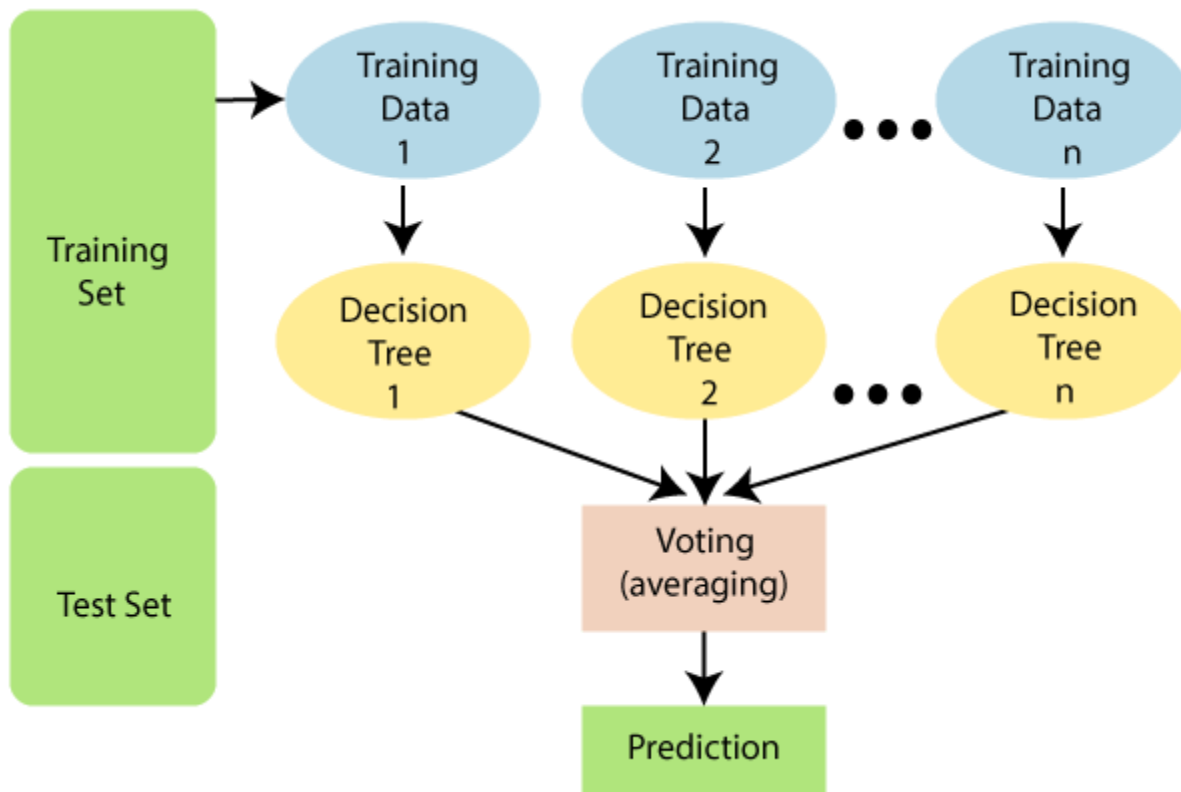
Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning**, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.



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:



Note: To better understand the Random Forest Algorithm, you should have knowledge of the Decision Tree Algorithm.

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.

- 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-phase first is to create the random forest by combining N decision tree, 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:

Implementation in Scikit-learn

For each decision tree, Scikit-learn calculates a nodes 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 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_i = the importance of feature i
- ni_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:

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

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:

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

- $RF_{fi\ sub(i)}$ = the importance of feature i calculated from all trees in the Random Forest model
- $norm_{fi\ sub(ij)}$ = 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 [StackExchange 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:

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

- $fi\ sub(i)$ = 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 `computeFeatureImportance` in [treeModels.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:

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

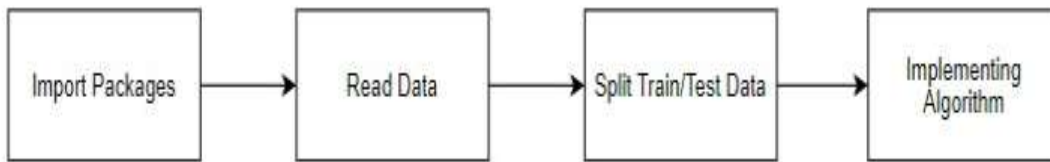
- $normfi_{sub(i)}$ = the normalized importance of feature i
- $fi_{sub(i)}$ = the importance of feature i

Then feature importance values from each tree are summed normalized:

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

- $RFfi_{sub(i)}$ = the importance of feature i calculated from all trees in the Random Forest model
- $normfi_{sub(ij)}$ = the normalized feature importance for i in tree j

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

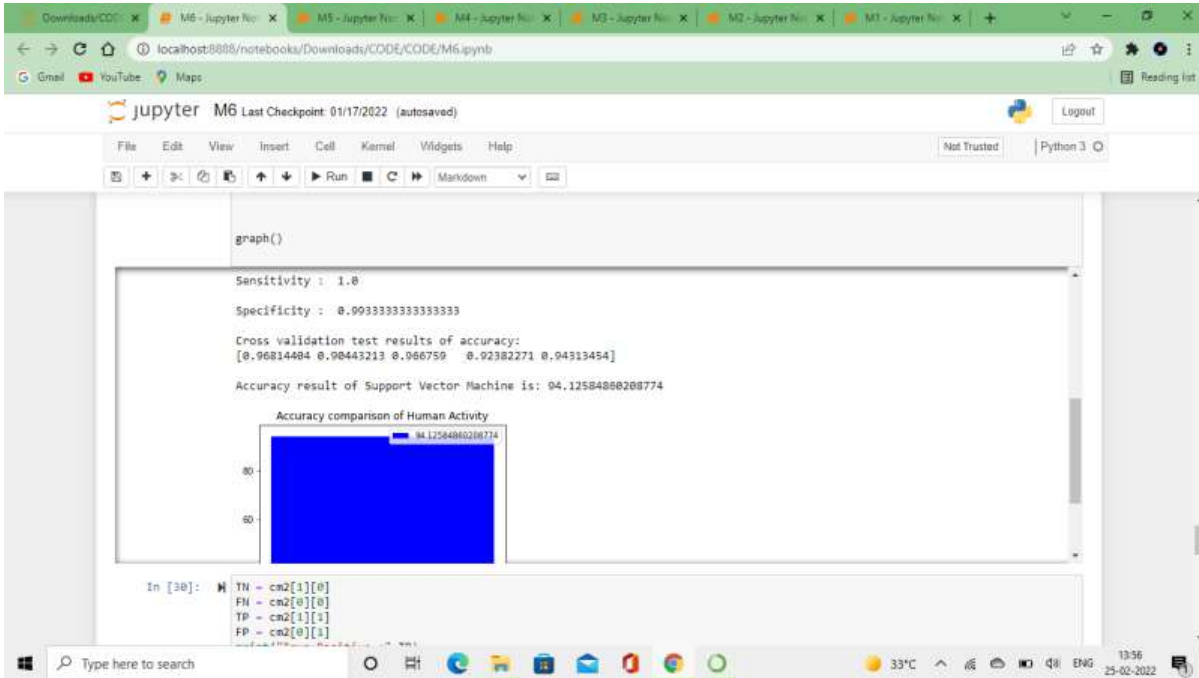
output : getting accuracy

Support Vector 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. The hyperplane (line) is found through the **maximum margin**, i.e., the maximum distance between data points of both classes.

Don't you think the definition and idea of SVM look a bit abstract? No worries, let me explain in details.



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.

The dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

The distance of the vectors from the hyperplane is called the **margin**, which is a separation of a line to the closest class points. We would like to choose a hyperplane

that maximises the margin between classes. The graph below shows what good margin and bad margin are.

Hard Margin

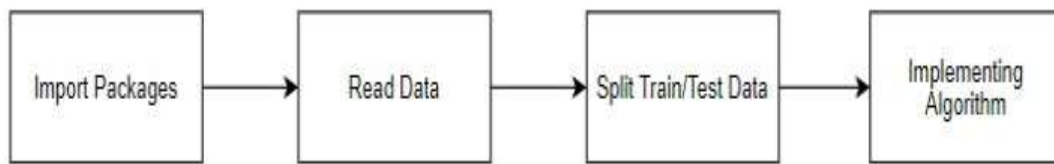
If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

Soft Margin

As most of the real-world data are not fully linearly separable, we will allow some margin violation to occur, which is called soft margin classification. It is better to have a large margin, even though some constraints are violated. Margin violation means choosing a hyperplane, which can allow some data points to stay in either the incorrect side of the hyperplane and between the margin and the correct side of the hyperplane.

In order to find the **maximal margin**, we need to maximize the margin between the data points and the hyperplane. In the following session, I will share the mathematical concepts behind this algorithm.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

CHAPTER - 5

RESULTS AND DISCUSSION

5.1 DEPLOY

Flask (web framework)

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 Pocco, 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 Pocco projects Werkzeug and Jinja were developed.

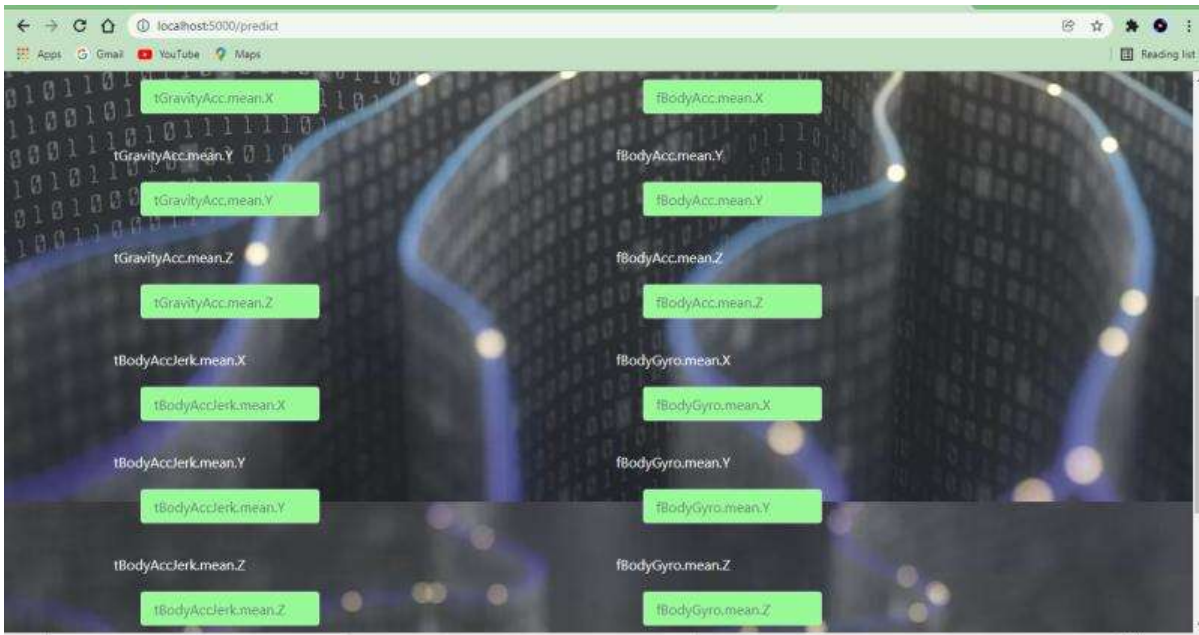
In April 2016, the Pycoc team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.

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.

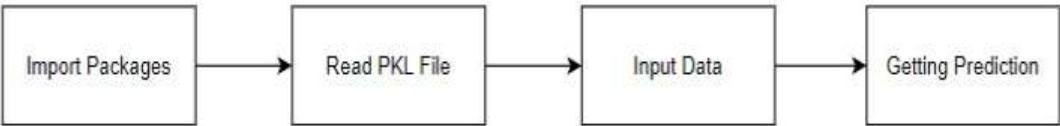
The micro-framework Flask is part of the Pallets Projects, and based on several others of them.

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





MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

FEATURES:

Flask was designed to be **easy to use and extend**. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you

are free to **plug in any extensions** you think you need. Also you are free to build your own modules. Flask is great for all kinds of projects. It's especially good for prototyping. Flask depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Still the question remains why use Flask as your web application framework if we have immensely powerful Django, Pyramid, and don't forget web mega-framework Turbo-gears? Those are supreme Python web frameworks BUT out-of-the-box Flask is pretty impressive too with its:

- Built-In Development server and Fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 Templating
- support for secure cookies (Client Side Sessions)
- Unicode based
- Extensive Documentation
- Google App Engine Compatibility
- Extensions available to enhance features desired

Plus Flask gives you so much more **CONTROL** on the development stage of **your project**. It follows the principles of minimalism and let you decide how you will build your application.

- Flask has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
- ORM-agnostic: you can plug in your favourite ORM e.g. SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.
- Flask documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Flask.
- It is super easy to deploy Flask in production (Flask is 100% WSGI 1.0 compliant”)
- HTTP request handling functionality
- High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Flask is one of the most polished and feature-rich micro frameworks available. Still young, Flask has a thriving community, first-class extensions, and an **elegant API**. Flask comes with all the benefits of fast templates, strong WSGI features, **thorough unit testability** at the web application and library level, **extensive documentation**. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Flask

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

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.

Overview of Python Flask Framework Web apps are developed to generate content based on retrieved data that changes based on a user's interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Flask is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features.

Flask is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Flask is a good start if you are getting into web development. Flask is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Advantages of Flask:

- Higher compatibility with latest technologies.
- Technical experimentation.

- Easier to use for simple cases.
- Codebase size is relatively smaller.
- High scalability for simple applications.
- Easy to build a quick prototype.
- Routing URL is easy.
- Easy to develop and maintain applications.

Framework Flask is a web framework from Python language. Flask provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework flask still doesn't use the Model View Controller (MVC) method.

Flask-RESTful is an extension for Flask that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Flask-Restful is a lightweight abstraction that works with the existing ORM/libraries. Flask-RESTful encourages best practices with minimal setup.

Flask Restful is an extension for Flask that adds support for building REST APIs in Python using Flask as the back-end. It encourages best practices and is very easy to set up. Flask restful is very easy to pick up if you're already familiar with flask.

Flask is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also we can add to this application to create our API.

Start Using an API

1. Most APIs require an API key. ...
2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.

The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an `__init__.py` file inside) or a standard module (just a `.py` file).

For more information about resource loading, see `open_resource()`.

Usually you create a Flask instance in your main module or in the `__init__.py` file of your package.

Parameters

- **rule** (*str*) – The URL rule string.
- **endpoint** (*Optional[str]*) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to `view_func.__name__`.

- **view_func** (*Optional[Callable]*) – The view function to associate with the endpoint name.
 - **provide_automatic_options** (*Optional[bool]*) – Add the OPTIONS method and respond to OPTIONS requests automatically.
 - **options** (*Any*) – Extra options passed to the Rule object.
- Return type -- None

After_Request(f)

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining `after_request` functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use `teardown_request()` for that.

Parameters:

f (*Callable[[Response], Response]*)

Return type

`Callable[[Response], Response]`

`after_request_funcs: t.Dict[AppOrBlueprintKey,`

`t.List[AfterRequestCallable]]`

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the after_request() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

app_context()

Create an AppContext. Use as a with block to push the context, which will make current_app point at this application.

An application context is automatically pushed by RequestContext.push() when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With app.app_context():

Init_db()

5.2 HTML Introduction

HTML stands for HyperText Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-

readable. The language uses tags to define what manipulation has to be done on the text.

Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.



`<!DOCTYPE html>` — This tag **specifies the language** you will write on the page. In this case, the language is HTML 5.

`<html>` — This tag signals that from here on we are going to write in HTML code.

`<head>` — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

`<body>` — This is where the **content of the page** goes.

Further Tags

Inside the `<head>` tag, there is one tag that is always included: `<title>`, but there are others that are just as important:

`<title>`

This is where we **insert the page name** as it will appear at the top of the browser window or tab.

`<meta>`

This is where information *about* the document is stored: character encoding, name (page context), description.

Head Tag

```
<head>
```

```
<title>My First Webpage</title>
```

```
<meta charset="UTF-8">
```

```
<meta name="description" content="This field contains information about your page. It  
is usually around two sentences long.">
```

```
<meta name="author" content="Conor Sheils">
```

```
</head>
```

Adding Content

Next, we will make `<body>` tag.

The HTML `<body>` is where we add the content which is designed for viewing by human eyes.

This includes **text**, **images**, **tables**, **forms** and everything else that we see on the internet each day.

Add HTML Headings To Web Page

In HTML, **headings** are written in the following elements:

- `<h1>`
- `<h2>`
- `<h3>`
- `<h4>`
- `<h5>`
- `<h6>`

As you might have guessed `<h1>` and `<h2>` should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

Search engine bots use this order when deciphering which information is most important on a page.

Creating Your Heading

Let's try it out. On a new line in the HTML editor, type:

```
<h1>Welcome to My Page</h1>
```

And hit save. We will save this file as "index.html" in a new folder called "my webpage."

The Moment of Truth: Click the newly saved file and your first ever web page should open in your default browser. It may not be pretty it's yours... all yours. *Evil laugh*

Adding text to our HTML page is simple using an element		Bold	Highlight important information
		Strong	Similarly to bold, to highlight key text
	<i>	Italic	To denote text
		Emphasised Text	Usually used as image captions
	<mark>	Marked Text	Highlight the background of the text
	<small>	Small Text	To shrink the text
	<strike>	Striked Out Text	To place a horizontal line across the text
	<u>	Underlined Text	Used for links or text highlights
	<ins>	Inserted Text	Displayed with an underline to show an insert
	<sup>	Superscript Text	Another typographical presentation style

nt opened with the tag `<p>` which **creates a new paragraph**. We place all of our regular text inside the element `<p>`.

When we write text in HTML, we also have a number of other elements we can use to **control the text or make it appear in a certain way**.

Add Links In HTML

As you may have noticed, the internet is made up of lots of [links](#).

Almost everything you click on while surfing the web is a link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the `<a>` tag. This element is the first that we've met which uses an attribute and so it **looks different to previously mentioned tags**.

```
<a href="http://www.google.com">Google</a>
```

Image Tag

In today's modern digital world, images are everything. The `` tag has everything you need to display images on your site. Much like the `<a>` anchor element, `` also contains an attribute.

The attribute features information for your computer regarding the **source, height, width** and **alt text** of the image

```

```

25. CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users. CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

CSS Syntax

```
Selector {
    Property 1 : value;
    Property 2 : value;
    Property 3 : value;
}
```

For example:

```
h1
{
    Color: red;
    Text-align: center;
}

#unique
{
    color: green;
}
```

- Selector: selects the element you want to target
- Always remains the same whether we apply internal or external styling
- There are few basic selectors like tags, id's, and classes
- All forms this key-value pair

- Keys: properties(attributes) like color, font-size, background, width, height,etc
- Value: values associated with these properties

CSS Comment

- Comments don't render on the browser
- Helps to understand our code better and makes it readable.
- Helps to debug our code
- Two ways to comment:
 - Single line

CSS How-To

- There are 3 ways to write CSS in our HTML file.
 - Inline CSS
 - Internal CSS
 - External CSS
- Priority order
 - Inline > Internal > External

Inline CSS

- Before CSS this was the only way to apply styles
- Not an efficient way to write as it has a lot of redundancy
- Self-contained
- Uniquely applied on each element
- The idea of separation of concerns was lost

- Example:

`<h3 style=" color:red"> Have a great day </h3>`

`<p style =" color: green"> I did this , I did that </p>`

Internal CSS

- With the help of style tag, we can apply styles within the HTML file
- Redundancy is removed
- But the idea of separation of concerns still lost
- Uniquely applied on a single document
- Example:

`< style>`

```
h1{  
    color:red;  
}
```

`</style>`

`<h3> Have a great day </h3>`

External CSS

- With the help of `<link>` tag in the head tag, we can apply styles
- Reference is added
- File saved with .css extension
- Redundancy is removed
- The idea of separation of concerns is maintained
- Uniquely applied to each document

- Example:

```
<head>
```

```
<link rel="stylesheet" type="text/css" href="name of the Css file">
```

```
</head>
```

```
h1{  
    color:red;    //.css file  
}
```

CSS Selectors

- The selector is used to target elements and apply CSS
- Three simple selectors
 - Element Selector
 - Id Selector
 - Class Selector
- Priority of Selectors

CSS Colors

- There are different colouring schemes in CSS
- **RGB**-This starts with RGB and takes 3 parameter
- **HEX**-Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
- **RGBA**-This starts with RGB and takes 4 parameter

CSS Background

- There are different ways by which CSS can have an effect on HTML elements
- Few of them are as follows:
 - Color – used to set the color of the background
 - Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
 - Image – used to set an image as the background
 - Position – used to determine the position of the image
 - Attachment – It basically helps in controlling the mechanism of scrolling

CSS BoxModel

- Every element in CSS can be represented using the BOX model
- It allows us to add a border and define space between the content
- It helps the developer to develop and manipulate the elements
- It consists of 4 edges
 - Content edge – It comprises of the actual content
 - Padding edge – It lies in between content and border edge
 - Border edge – Padding is followed by the border edge
 - Margin edge – It is an outside border and controls the margin of the element

CHAPTER – 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score is will be find out. This application can help to find the Human Activity Based on the Smartphone sensor.

6.2 FUTURE WORK

- Human Activity Recognition connect with AI model.
- To automate this process by show the prediction result in web application or desktop application.
- To optimize the work to implement in Artificial Intelligence environment.

REFERENCES

- [1]. <https://doi.org/10.3389/frobt.2015.00028>. "A Review of Human Activity Recognition Methods" Michalis Vrigkas¹, Christophoros Nikou^{1*} and Ioannis A. Kakadiaris², 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]. Profound 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 Khan¹, Majed Alhaisoni², Ammar Armghan³, Fayadh Alenezi³, Usman Tariq⁴, Yunyoung Nam^{5,*}, Tallha Akram⁶, "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, GEINTRA Group, University of Alcala.

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

APPENDICES

A. Coding

Module 1: Data validation and pre-processing technique

```
#import library packages
import pandas as pd
import numpy as np

#load given dataset
data=pd.read_csv("human.csv")

data

data.head()

data.tail()

#shape
data.shape

# size
data.size

#Checking datatype and information about dataset
data.info()

#To describe the dataframe
data.describe()

#show columns
data.columns
```

```

data.head()

data.isnull()

#null values
data.isnull().sum()

data.duplicated()

data.duplicated().sum()

# drop duplivcate values
data.dropna()

data.duplicated().sum()

#correlation
data.corr()

data.activity.unique()

data["label"]=data.activity.map({"STANDING":"STANDING","SITTING":"SITTING",
"LAYING":"LAYING","WALKING":"WALKING",'WALKING_DOWNSTAIRS':"WALKING",'WALKI
NG_UPSTAIRS':"WALKING"})

data.label.unique()

del data['activity']

data.describe()

pd.Categorical(data['label']).describe()

len(data['fBodyGyro.mean.X'].unique())

```

```
len(data['fBodyGyro.mean.Y'].unique())
```

```
len(data['fBodyGyro.mean.Z'].unique())
```

```
data["label"].value_counts()
```

```
data['tBodyAcc.mean.X'].unique()
```

```
data['tBodyAcc.mean.Y'].unique()
```

```
data['tBodyAcc.mean.Z'].unique()
```

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['tBodyAcc.mean.X', 'tBodyAcc.mean.Y', 'tBodyAcc.mean.Z',
           'tGravityAcc.mean.X', 'tGravityAcc.mean.Y', 'tGravityAcc.mean.Z',
           'tBodyAccJerk.mean.X', 'tBodyAccJerk.mean.Y', 'tBodyAccJerk.mean.Z',
           'tBodyGyro.mean.X', 'tBodyGyro.mean.Y', 'tBodyGyro.mean.Z',
           'fBodyAcc.mean.X', 'fBodyAcc.mean.Y', 'fBodyAcc.mean.Z',
           'fBodyGyro.mean.X', 'fBodyGyro.mean.Y', 'fBodyGyro.mean.Z', 'label']
le = LabelEncoder()
for i in var_mod:
    data[i] = le.fit_transform(data[i]).astype(int)
```

In []:

```
data.head()
```

In []:

```
data["label"].unique()
```

Module 2: Exploration data analysis of visualization and training a model by given attributes

```
#import library packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
#loading dataset
data=pd.read_csv("human.csv")
```

```
data.head(10)
```

```
data.shape
```

```
data.size
```

```
data.isnull().sum()
```

```
data.duplicated()
```

```
data.dropna()
```

```
data.shape
```

```
data.columns
```

```
data.activity.unique()
```

```
data["label"]=data.activity.map({"STANDING":"STANDING","SITTING":"SITTING",  
"LAYING":"LAYING","WALKING":"WALKING",'WALKING_DOWNSTAIRS':"WALKING",  
'WALKING_UPSTAIRS':"WALKING"})
```

```
data.columns
```

```
data.label.unique()
```

```
del data['activity']
```

```
plt.hist(data.label,color="gray")  
plt.legend(title="Human Activity")  
plt.xlabel("Activity")  
plt.ylabel("Counts")  
plt.show()
```

```
plt.hist(data['tBodyAcc.mean.X'],color="red")  
plt.xlabel("tBodyAcc.mean.X")  
plt.show()
```

```
plt.hist(data['tBodyAcc.mean.Y'],color="blue")  
plt.xlabel("tBodyAcc.mean.Y")  
plt.show()
```

```
plt.hist(data['tBodyAcc.mean.Z'],color="green")  
plt.xlabel("tBodyAcc.mean.Z")  
plt.show()
```

```
plt.show()
```

```
#Propagation by variable  
def PropByVar(data, variable):  
    dataframe_pie = data[variable].value_counts()
```

```

    ax = dataframe_pie.plot.pie(figsize=(7,12), autopct='%1.2f%%',
fontsize = 12)
    ax.set_title(variable + ' \n', fontsize = 15)
    plt.show()
    return np.round(dataframe_pie/data.shape[0]*100,2)
PropByVar(data,"label")
plt.show()

```

```

data.columns

```

```

plt.figure(figsize=(10,6))
plt.plot(data['tBodyGyro.mean.X'])
plt.show()

```

```

# Heatmap plot diagram
fig, ax = plt.subplots(figsize=(14,10))
sns.heatmap(data.corr(), ax=ax, annot=True)

```

```

boxplot = data.boxplot(column=['tBodyAcc.mean.Y'],
by="label",figsize=(8,6),  fontsize=15)
plt.show()

```

```

plt.figure(figsize=(12,8))
plt.scatter(data["tBodyGyro.mean.Y"],data["label"],color="red")

```

```

data1=data.iloc[:, :9]
data2=data.iloc[:, 9:-1]

```

```

a=pd.plotting.scatter_matrix(data1, alpha=0.05, figsize=(18,10),
diagonal='hist')
axis = 'off'

```

```

plt.show()

```

```

a=pd.plotting.scatter_matrix(data2, alpha=0.05, figsize=(18,10),
diagonal='hist')
axis = 'off'

```

```
plt.show()
```

```
plt.figure(figsize=(12,7))  
sns.countplot(x='label',data=data)
```

```
data.columns
```

```
from sklearn.preprocessing import LabelEncoder  
var_mod = ['tBodyAcc.mean.X', 'tBodyAcc.mean.Y', 'tBodyAcc.mean.Z',  
           'tGravityAcc.mean.X', 'tGravityAcc.mean.Y',  
           'tGravityAcc.mean.Z',  
           'tBodyAccJerk.mean.X', 'tBodyAccJerk.mean.Y',  
           'tBodyAccJerk.mean.Z',  
           'tBodyGyro.mean.X', 'tBodyGyro.mean.Y', 'tBodyGyro.mean.Z',  
           'fBodyAcc.mean.X', 'fBodyAcc.mean.Y', 'fBodyAcc.mean.Z',  
           'fBodyGyro.mean.X', 'fBodyGyro.mean.Y', 'fBodyGyro.mean.Z',  
           'label']  
le = LabelEncoder()  
for i in var_mod:  
    data[i] = le.fit_transform(data[i]).astype(int)
```

```
data.head()
```

Module 3 : Performance measurements of Logistic regression algorithms

```
#import library packages  
import pandas as p  
import matplotlib.pyplot as plt  
import seaborn as s
```

```

import numpy as n

#Load given dataset
data = p.read_csv("human.csv")

import warnings
warnings.filterwarnings('ignore')

data.head()

data.shape

data.size
len(data.columns)
data.isnull().sum()
data.dropna()
data.corr()
data.columns
data.activity.unique()
#mapping
data["label"]=data.activity.map({"STANDING":"STANDING","SITTING":"SITTING",
"LAYING":"LAYING","WALKING":"WALKING","WALKING_DOWNSTAIRS":"WALKING",
"WALKING_UPSTAIRS":"WALKING"})
data.label.unique()
data.head()
del data["activity"]
data.columns

```



```

#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)
data.label.unique()
data.head()
# now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
X = data.drop(labels='label', axis=1)
#Response variable
y = data.loc[:, 'label']
#We'll use a test size of 30%. We also stratify the split on the
response variable, which is very important to do because there are so
few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1, stratify=y)

```

Logistic Regression :

```

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

logR= LogisticRegression()

```

```

logR.fit(X_train,y_train)

predictLR = logR.predict(X_test)

print("")
print('Classification report of Logistic Regression Results:')
print("")
print(classification_report(y_test,predictLR))
x = (accuracy_score(y_test,predictLR)*100)

print('Accuracy result of Logisticregression is:', x)

print("")

cm2=confusion_matrix(y_test,predictLR)
print('Confusion Matrix result of Logistic Regression is:\n',cm2)
print("")
sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity2 )
print("")
specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity2)
print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")

```

```
print("Accuracy result of Logistic Regression is:",accuracy.mean() *  
100)  
LR=accuracy.mean() * 100
```

```
def graph():  
    import matplotlib.pyplot as plt  
    data=[LR]  
    alg="Logistic Regression"  
    plt.figure(figsize=(5,5))  
    b=plt.bar(alg,data,color="b")  
    plt.title("Accuracy comparison Human Activity",fontsize=15)  
    plt.legend(b,data,fontsize=9)
```

```
graph()
```

```
TN = cm2[1][0]  
FN = cm2[0][0]  
TP = cm2[1][1]  
FP = cm2[0][1]  
print("True Positive :",TP)  
print("True Negative :",TN)  
print("False Positive :",FP)  
print("False Negative :",FN)
```

```

print("")
TPR = TP / (TP+FN)
TNR = TN / (TN+FP)
FPR = FP / (FP+TN)
FNR = FN / (TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP / (TP+FP)
NPV = TN / (TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion matrix-LR',
cmap=plt.cm.Blues):
    target_names=['']
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = n.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm2=confusion_matrix(y_test, predictLR)

```

```
print('Confusion matrix-LR:')  
print(cm2)  
plot_confusion_matrix(cm2)
```

Module 4 : Performance measurements of DecisionTree:

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
#Warning  
import warnings  
warnings.filterwarnings('ignore')  
data=pd.read_csv("human.csv")  
data.head()  
data.shape  
data.shape  
data.duplicated().sum()  
data.dropna()  
data.shape  
data.isnull()  
data.isnull().sum()  
data.columns
```

```

data.activity.unique()
data["label"]=data.activity.map({'STANDING':'STANDING',
'SITTING':'SITTING', 'LAYING':'LAYING', 'WALKING':'WALKING',
'WALKING_DOWNSTAIRS':'WALKING','WALKING_UPSTAIRS':'WALKING'})
data.label.unique()
data.head()
del data["activity"]
data.describe()
data.corr()
#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)
data
data.label.unique()

# let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
X = data.drop(labels='label', axis=1)
#Response variable
y = data.loc[:, 'label']
#We'll use a test size of 30%. We also stratify the split on the
response variable, which is very important to do because there are so
few fraudulent transactions.

```

```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1, stratify=y)
#We'll use a test size of 30%. We also stratify the split on the
response variable, which is very important to do because there are so
few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1, stratify=y)

```

DecisionTree:

```

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

DT=DecisionTreeClassifier()

DT.fit(X_train,y_train)

predictDT = DT.predict(X_test)

print("")
print('Classification report DecisionTree classifier Results:')
print("")
print(classification_report(y_test,predictDT))

print("")
x = (accuracy_score(y_test,predictDT)*100)

```

```

print('Accuracy result of DecisionTree is:', x)
print("")

cm2=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of DecissionTree Classifier
is:\n',cm2)
print("")
sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity2 )
print("")
specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity2)
print("")

accuracy = cross_val_score(DT, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of DecisionTree Classifier
is:",accuracy.mean() * 100)
dt=accuracy.mean() * 100

def graph():
    import matplotlib.pyplot as plt
    data=[dt]

```



```

alg="Decision Tree"
plt.figure(figsize=(5,5))
b=plt.bar(alg,data,color="b")
plt.title("Accuracy comparison of Human activity",fontsize=15)
plt.legend(b,data,fontsize=9)

```

```

graph()
TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)

```

```

NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion matrix-DT',
cmap=plt.cm.Blues):
    target_names=['']
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm2=confusion_matrix(y_test, predictDT)
print('Confusion matrix-DT:')
print(cm2)
plot_confusion_matrix(cm2)

```

Module 5 : Performance measurements of Random Forest algorithms

```
import pandas as pd
```

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
data = pd.read_csv("human.csv")
data.head()
data.shapeIn [ ]:
data.size
data.columns
data.isnull()
data.isnull().sum()
data.duplicated()
data.dropna()
data.shape
data.corr()
data.columns
data.activity.unique()
#mapping
data["label"]=data.activity.map({"STANDING":"STANDING","SITTING":"SITTING",
"LAYING":"LAYING","WALKING":"WALKING",'WALKING_DOWNSTAIRS':"WALKING",
'WALKING_UPSTAIRS':"WALKING"})
data.head()
data.label.unique()
data.head()
del data["activity"]
data.columns
#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)
data.head()

# model, so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
X = data.drop(labels='label', axis=1)
#Response variable
y = data.loc[:, 'label']
#We'll use a test size of 30%. We also stratify the split on the
response variable, which is very important to do because there are so
few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1, stratify=y)

```

Random Forest:

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

rfc = RandomForestClassifier()

rfc.fit(X_train, y_train)

predictR = rfc.predict(X_test)

```

```

print("")
print('Classification report of Random Forest Results:')
print("")

print(classification_report(y_test,predictR))
x = (accuracy_score(y_test,predictR)*100)

print('Accuracy result of Random Forest is:', x)
print("")
cm1=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest is:\n',cm1)
print("")
sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Random Forest is:",accuracy.mean() * 100)
RFC=accuracy.mean() * 100

```

```

def graph():
    import matplotlib.pyplot as plt
    data=[RFC]
    alg="Random orest"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color="b")
    plt.title("Accuracy comparison of Human Activity")
    plt.legend(b,data,fontsize=9)

```

```

graph()

```

```

TN = cm1[1][0]
FN = cm1[0][0]
TP = cm1[1][1]
FP = cm1[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)

```

```

FNR = FN / (TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP / (TP+FP)
NPV = TN / (TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion matrix-RF',
cmap=plt.cm.Blues):
    target_names=['']
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names, rotation=45)
    plt.yticks(tick_marks, target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm2=confusion_matrix(y_test, predictR)
print('Confusion matrix-RF:')
print(cm2)
plot_confusion_matrix(cm2)

```

model building

```
import joblib
joblib.dump(rfc, 'RFC.pkl')
```

Module 6 : Performance measurements of Support Vector Machines algorithms

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("human.csv")
data.head()
data.shape
data.size
data.columns
data.isnull()
data.isnull().sum()
data.duplicated()
data.dropna()
data.shape
data.corr()
data.columns
data.activity.unique()
#mapping
```



```

data["label"]=data.activity.map({"STANDING":"STANDING","SITTING":"SITTING",
"LAYING":"LAYING","WALKING":"WALKING",'WALKING_DOWNSTAIRS':"WALKING",
'WALKING_UPSTAIRS':"WALKING"})
data.head()
data.label.unique()
data.head()
del data["activity"]
data.columns
#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)
data.head()
# now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
X = data.drop(labels='label', axis=1)
#Response variable
y = data.loc[:, 'label']
#We'll use a test size of 30%. We also stratify the split on the
response variable, which is very important to do because there are so
few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1, stratify=y)

```

Support Vector Machines:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

s = SVC()

s.fit(X_train,y_train)

predicts = s.predict(X_test)

print("")
print('Classification report of Support Vector Machines Results:')
print("")

print(classification_report(y_test,predicts))
x = (accuracy_score(y_test,predicts)*100)

print('Accuracy result of Support Vector Machines is:', x)
print("")
cm2=confusion_matrix(y_test,predicts)
print('Confusion Matrix result of Support Vector Machines is:\n',cm2)
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
```

```

print("")

accuracy = cross_val_score(s, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of Support Vector Machine is:",accuracy.mean()
* 100)
S=accuracy.mean() * 100

```

```

def graph():
    import matplotlib.pyplot as plt
    data=[S]
    alg="Support Vector Machine"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color="b")
    plt.title("Accuracy comparison of Human Activity")
    plt.legend(b,data,fontsize=9)

```

```

graph()

```

```

TN = cm2[1][0]
FN = cm2[0][0]
TP = cm2[1][1]
FP = cm2[0][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion matrix-SVM',
cmap=plt.cm.Blues):
    target_names=['']
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

```

```

tick_marks = np.arange(len(target_names))
plt.xticks(tick_marks, target_names, rotation=45)
plt.yticks(tick_marks, target_names)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

cm2=confusion_matrix(y_test, predicts)
print('Confusion matrix-SVM:')
print(cm2)
plot_confusion_matrix(cm2)

```

Flask deploy

```

import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
import joblib

app = Flask(__name__)
model = joblib.load('RFC.pkl')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''

```

```

For rendering results on HTML GUI
'''
int_features = [(x) for x in request.form.values()]
final_features = [np.array(int_features)]
print(final_features)
prediction = model.predict(final_features)

output = prediction[0]
print(output)
if output == 0:
    output = 'Laying'
elif output == 1:
    output = 'Sitting'
elif output == 2:
    output = 'Standing'
else:
    output = 'Walking'

    return render_template('index.html', prediction_text='HUMAN :
{}'.format(output))

if __name__ == "__main__":
    app.run(host="localhost", port=6002)

```

HTML & CSS

```

<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
    <meta charset="UTF-8">
    <title>TITLE</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">

```

```

<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<style>
.back{
  background-image: url("{ url_for('static', filename='image/img.jpg') }");
}
.white{
color:white;
}
.space{
margin:10px 30px;
padding:15px 10px;
background: palegreen;
width:auto;
}
.gap{
padding:10px 20px;
}

</style>

</head>

<body class="back">
  <div>

    <div class="jumbotron">
      <h1 style="text-align:center"> HUMAN ACTIVITY </h1>

    </div>

    <!-- Main Input For Receiving Query to our ML -->
    <form class="form-group" action="{ url_for('predict') }" method="post">

```

```

<div class="container">
  <div class="row">
    <div class="gap col-md-6">

      <label class="white" for=""> tBodyAcc.mean.X </label>
      <input type="number" class="space form-control" step="0.01" name="
tBodyAcc.mean.X " placeholder=" tBodyAcc.mean.X " required="required" /><br>
      <label class="white" for=""> tBodyAcc.mean.Y </label>
      <input type="number" class="space form-control" step="0.01" name="
tBodyAcc.mean.Y " placeholder=" tBodyAcc.mean.Y " required="required" /><br>
      <label class="white" for=""> tBodyAcc.mean.Z </label>
      <input type="number" class="space form-control" step="0.01" name="
tBodyAcc.mean.Z " placeholder=" tBodyAcc.mean.Z " required="required" /><br>

      <label class="white" for=""> tGravityAcc.mean.X </label>
      <input type="number" class="space form-control" step="0.01" name="
tGravityAcc.mean.X " placeholder=" tGravityAcc.mean.X " required="required" /><br>
      <label class="white" for=""> tGravityAcc.mean.Y </label>
      <input type="number" class="space form-control" step="0.01" name="
tGravityAcc.mean.Y " placeholder=" tGravityAcc.mean.Y " required="required" /><br>
      <label class="white" for=""> tGravityAcc.mean.Z </label>
      <input type="number" class="space form-control" step="0.01" name="
tGravityAcc.mean.Z " placeholder=" tGravityAcc.mean.Z " required="required" /><br>

      <label class="white" for=""> tBodyAccJerk.mean.X </label>
      <input type="number" class="space form-control" step="0.01" name="
tBodyAccJerk.mean.X " placeholder=" tBodyAccJerk.mean.X " required="required" /><br>
      <label class="white" for=""> tBodyAccJerk.mean.Y </label>
      <input type="number" class="space form-control" step="0.01" name="
tBodyAccJerk.mean.Y " placeholder=" tBodyAccJerk.mean.Y " required="required" /><br>
      <label class="white" for=""> tBodyAccJerk.mean.Z </label>
      <input type="number" class="space form-control" step="0.01" name="
tBodyAccJerk.mean.Z " placeholder=" tBodyAccJerk.mean.Z " required="required" /><br>
    </div>
    <div class="gap col-md-6">
      <label class="white" for=""> tBodyGyro.mean.X </label>

```



```

        <input type="number" class="space form-control" step="0.01" name="
tBodyGyro.mean.X " placeholder=" tBodyGyro.mean.X " required="required" /><br>
        <label class="white" for=""> tBodyGyro.mean.Y </label>
        <input type="number" class="space form-control" step="0.01" name="
tBodyGyro.mean.Y " placeholder=" tBodyGyro.mean.Y " required="required" /><br>
        <label class="white" for=""> tBodyGyro.mean.Z </label>
        <input type="number" class="space form-control" step="0.01" name="
tBodyGyro.mean.Z " placeholder=" tBodyGyro.mean.Z " required="required" /><br>

```

```

        <label class="white" for=""> fBodyAcc.mean.X</label>
        <input type="number" class="space form-control" step="0.01" name="
fBodyAcc.mean.X" placeholder=" fBodyAcc.mean.X" required="required" /><br>
        <label class="white" for=""> fBodyAcc.mean.Y</label>
        <input type="number" class="space form-control" step="0.01" name="
fBodyAcc.mean.Y" placeholder=" fBodyAcc.mean.Y" required="required" /><br>
        <label class="white" for=""> fBodyAcc.mean.Z</label>
        <input type="number" class="space form-control" step="0.01" name="
fBodyAcc.mean.Z" placeholder=" fBodyAcc.mean.Z" required="required" /><br>

```

```

        <label class="white" for=""> fBodyGyro.mean.X </label>
        <input type="number" class="space form-control" step="0.01" name="
fBodyGyro.mean.X " placeholder=" fBodyGyro.mean.X " required="required" /><br>
        <label class="white" for=""> fBodyGyro.mean.Y </label>
        <input type="number" class="space form-control" step="0.01" name="
fBodyGyro.mean.Y " placeholder=" fBodyGyro.mean.Y " required="required" /><br>
        <label class="white" for=""> fBodyGyro.mean.Z </label>
        <input type="number" class="space form-control" step="0.01" name="
fBodyGyro.mean.Z " placeholder=" fBodyGyro.mean.Z " required="required" /><br>

```

```

</div>

```

```

</div>

```

```

</div>

```

```

<div style="padding:2% 35%">

```

```

    <button type="submit" class="btn btn-success btn-block"
style="width:350px;padding:20px">Predict</button>

```

```
</div>

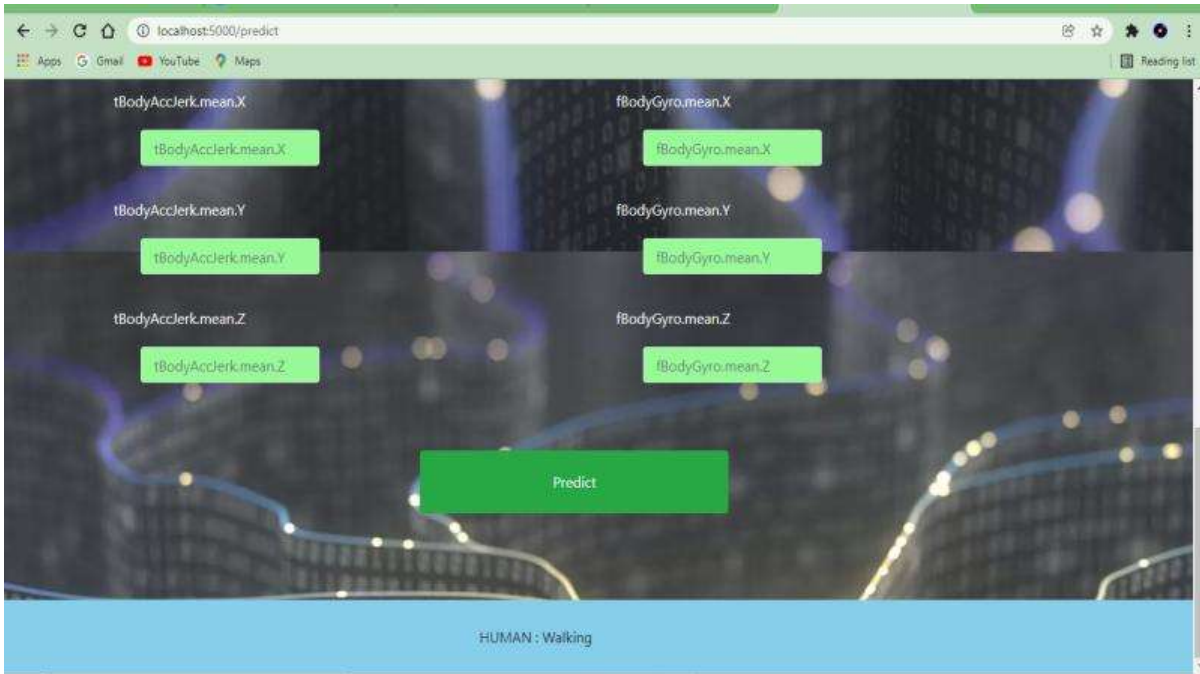
    </form>

    <br>
    <br>
<div style="background:skyblue;padding:2% 40%">
    {{ prediction_text }}
</div>
</div>

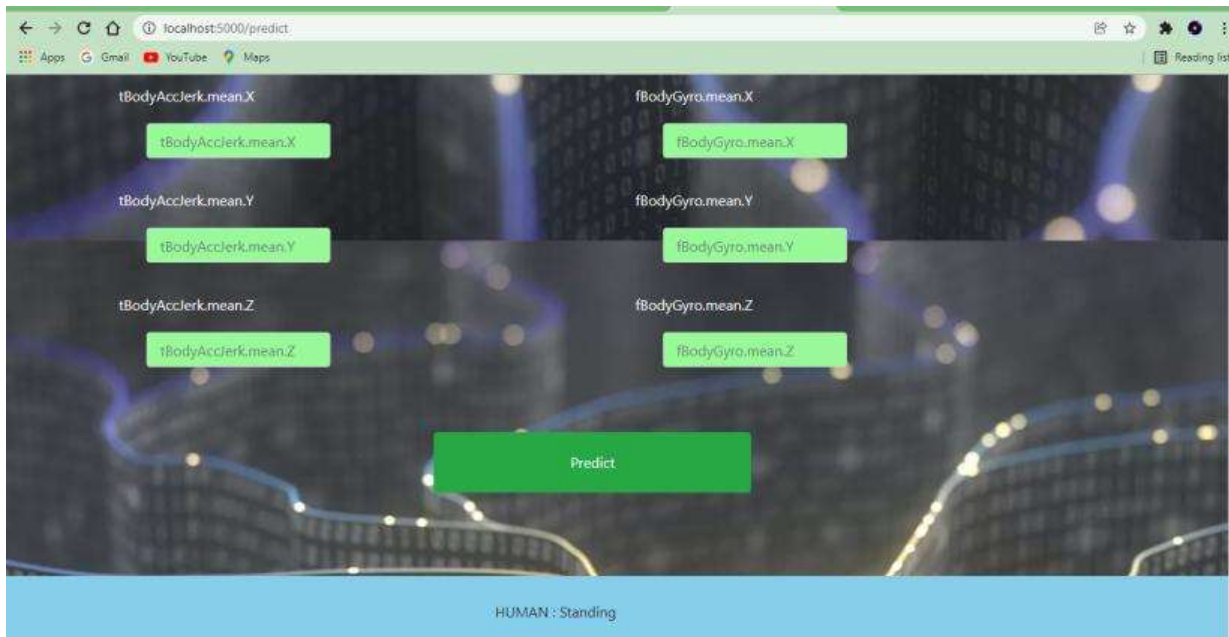
</body>
</html>
```

B. SCREENSHOTS

Walking:



Standing:



Laying:

localhost:5000/predict

tBodyAccJerk.mean.X
tBodyAccJerk.mean.X

tBodyAccJerk.mean.Y
tBodyAccJerk.mean.Y

tBodyAccJerk.mean.Z
tBodyAccJerk.mean.Z

fBodyGyro.mean.X
fBodyGyro.mean.X

fBodyGyro.mean.Y
fBodyGyro.mean.Y

fBodyGyro.mean.Z
fBodyGyro.mean.Z

Predict

HUMAN : Laying

Sitting:

localhost:5000/predict

tBodyAccJerk.mean.X
tBodyAccJerk.mean.X

tBodyAccJerk.mean.Y
tBodyAccJerk.mean.Y

tBodyAccJerk.mean.Z
tBodyAccJerk.mean.Z

fBodyGyro.mean.X
fBodyGyro.mean.X

fBodyGyro.mean.Y
fBodyGyro.mean.Y

fBodyGyro.mean.Z
fBodyGyro.mean.Z

Predict

HUMAN : Sitting

C. PUBLICATION WITH PLAGIARISM REPORT

HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES USING MACHINE LEARNING PROCESS

by Turnitin Official

Submission date: 04-Feb-2022 01:04AM (UTC+0900)

Submission ID: 1754126292

File name: ITPML16.docx (129.91K)

Word count: 2062

Character count: 11685

HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES USING MACHINE LEARNING PROCESS

ABSTRACT:

Human action acknowledgment requires foreseeing the activity of an individual dependent on sensor-produced information. It has drawn in significant interest in the beyond couple of years, because of the enormous number of utilizations empowered by present day pervasive figuring gadgets. It arrange information into action like Walking, strolling up steps, strolling down steps, sitting, standing, laying are perceived. Commotion channels were used to pre-handle sensor signals (accelerometer and spinner) to produce sensor information. A Butterworth low-pass channel was used to separate the sensor speed increase signal from the gravitational and body movement components. Low recurrence portions are predicted in gravitational power. Worked out factors from the time and recurrence space produced a vector of highlights. Predicting AI-based methods for Human Activity Recognition provides the most accurate results. All of the given dataset will be examined using a regulated AI technique (SMLT) in order to catch a few data similarities, variable recognisable proof, univariate investigation, bi-variate investigation, and multivariate investigation of missing worth medicines, information cleaning/getting ready, and information perception. To present an AI-based technique for accurately forecasting the stock cost index based on predicted outcomes as stock cost increment or stable state highest accuracy from a variety of AI computations. Besides that, the presentation of various AI computations from the provided vehicle traffic division dataset will be assessed. categorization of the data and a dataset with an evaluation report that illustrates the feasibility of the proposed AI calculation technique can measure up to best exactness with precision, it is possible to distinguish a chaotic lattice, Recall and F1 Score.

Keywords: Machine Learning, HAR(Human Activity Recognition) , Data Science

INTRODUCTION

MACHINE LEARNING

AI is the ability to predict the future based on previous data. (ML) is a kind of computerised reasoning (AI) that enables PCs to learn without being rewritten. Computer programmes that can adapt to new information are at the heart of AI, as are machine learning's nuts and bolts and the execution of a fundamental AI computation in python. The use of specific computations is an element of the interaction between planning and forecasting. Uses preparation data to make projections on other test data based on the calculation's use of this preparation data.

LITERATURE SURVEY

Title : A review of human activity Recognition methods

Author Michalis Vrigkas¹, Christophoros Nikou¹ and Ioannis A. Kakadiaris

Year : 16- November- 2016

Perceiving human movements from video or still images is a challenging task due of factors such as foundation mess, fractional obstruction, changes in size, view point, illumination, and visual appearance. A variety of action acknowledgment frameworks are required for a variety of applications, including

video observation frameworks, human-PC cooperation, and sophisticated mechanics for human conduct portrayal. Here, we provide a comprehensive list of the most recent and cutting-edge developments in human action arrangement research. We propose a classification of human movement approaches and examine their benefits and impediments. Specifically, we partition human movement characterization techniques in to two enormous classifications as indicated by whether or not they use information from various modalities. Then, at that point, every one of these classifications is additionally examined into sub-classifications, which reflect how they model human exercises and what kind of exercises they are keen on. Additionally, we give a thorough investigation of the current, openly accessible human action arrangement datasets and look at the prerequisites for an ideal human movement acknowledgment dataset. An extensive survey of existing human action characterization benchmarks was likewise introduced and we analyzed the provokes of information securing to the issue of understanding human action. At last, we gave the qualities of building an optimal human action acknowledgment framework.

Title : In video surveillance, a survey on activity recognition and behaviour understanding was conducted.

Author : Sarvesh Vishwakarma, Anupam Agrawal

Year : October 2012

This paper gives a thorough study to action acknowledgment in video observation. It begins with a portrayal of basic and complex human action, and different applications. The uses of movement acknowledgment are complex, going from visual observation through content based recovery to human PC connection. A broad range of topics related to human movement acknowledgement are included in the scope of this paper's connections. Then, at this point, it summarises and categorises the subsequent dispersed exploration advancements.

All the preprocessing (i.e., classification and identification) procedures have been summarised in this study. The connection we've provided shows a slew of restrictions and unresolved issues. Due to the presence of enlightenment and climatic fluctuations, shadows, self-impediment and full obstruction, it is difficult to recognise movement in unusual surroundings. Division methods still need quick and accurate techniques in order to affect the final decision phases. Portrayal

based methodologies are doing admirably to perceive undeniable level exercises whose sub occasions are coordinated simultaneously and happening in a consecutive way in contrast with the measurable or syntactic methodologies. The factual and syntactic methodologies can adequately deal with the action video dirtied with commotion.

Title : Occupation-Related Physical Activity and Obesity in the United States during the past five decades

Author: Diana M. Thomas, Catrine Tudor-Locke, Peter T. Katzmarzyk, Conrad P. Earnest, Ruben Q. Rodarte, Corby K. Martin and Claude Bouchard are the authors of this paper.

Year : May 2011

The genuine reasons for the heftiness scourge are not surely known and there are not many longitudinal populace based information distributed analyzing this issue. The purpose of this study was to look for changes in word-related active labour over the last fifty years and see whether these trends correlate with changes in body weight in the United States. Data from the US Department of Labor Statistics was used to analyse energy usage in private sector employment in the United States starting about 1960. Body mass index (BMI) was derived from

the results of U.S. Health and Nutrition Examination Surveys (NHANES). Nearly half of all jobs in private business in the United States required active work using just modest amounts of force in the mid-1960s; now, only about a quarter of all jobs need this level of energy consumption. Beginning around 1960 the assessed mean every day energy use because of business related actual work has dropped by in excess of 100 calories in all kinds of people. Around 1960, the energy balance model predicted loads based on changes in daily energy use associated to various occupations. Using gauge weights from 1960–2002, we calculated that a 142-calorie reduction would increase the average burden to 89.7 kg, which was in line with the 2003–06 NHANES average weight of 91.8 kg. For women, the results were the same. We estimate that during the last 50 years in the United States, daily occupation-related energy usage has decreased by over 100 calories, and this drop in energy use accounts for a significant portion of the rise in mean body loads for women and men in the United States.

EXISTING SYSTEM

This paper proposes and fosters a full profound neural organization (CDNN) to

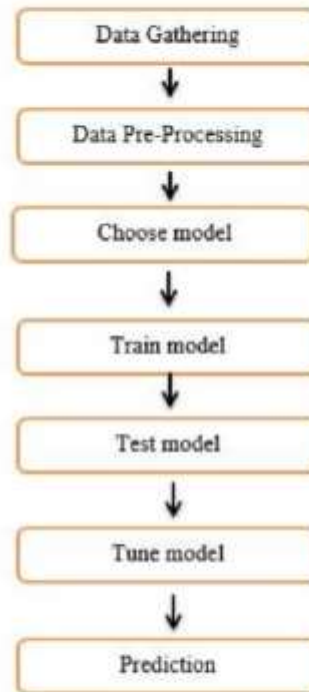
break down information, gathered utilizing the sensors of advanced cells, to precisely limit an item in an indoor climate. There are many existing investigations that have endeavored to distinguish the area of an occupant in a room through the examination of the radio transmission strength (RSS), with changing achievement. The strength of the RSS differs with distance and the presence of impediments inside the view. Accordingly, a mechanized framework utilizing RSS signal in one climate probably won't work in another. In this paper accordingly, we propose and foster an alternate restriction technique dependent on information gathered from various sensors installed in an advanced mobile phone. To dissect and anticipate the specific area inside an extremely brief distance (say a 1 to 1.5 m sweep), we foster an original CDNN. The indoor confinement of items has part of utilization inside workplaces, emergency clinics and public spots. The proposed CDNN experiences space and computational intricacies, uncommonly for preparing every one of the DNNs in the CDNN. We additionally plan to further develop the CDNN construction to such an extent that the quantity of DNNs can be decreased without influencing the restriction precision.

PROPOSED SYSTEM

A general-purpose pattern recognition system's method is very similar to that of recognising human actions, which includes everything from data gathering to activity categorization. The raw data from sensors is converted in a series of steps to provide useful models for classifying human actions. Machine learning methods (e.g., SVM, decision tree, Random forest) may be used as shallow algorithms in the HAR methodology for cellphones with inertial sensors. As a part of this process, we will conduct variable identification, univariate analysis, bivariate analysis, and multivariate analysis, as well as missing value treatment and data validation.

To properly analyse parameters, a photo graphic-based approach must be used, and the resulting data set is large in size. To get around this, we'll utilise a GUI programme with a machine learning methodology built in. Different machine learning techniques and datasets from various sources would be integrated to create a generalised dataset from which to extract patterns to get the most accurate results.

WORKFLOW DIAGRAM



SYSTEM ARCHITECTURE



Module description:

Data Pre-processing

Approval methods in AI are used to get the ML model's failure rate, which may be regarded as being close to the dataset's true error rate. In most cases, you don't need to go through the approval process if the data volume is large enough to represent the population. Even in real-world settings, it is necessary to deal with data that may or may not be a true indicator of the population in a specific dataset. Finding and resolving information type's value, whether it's a float variable or whole integer. When tweaking model hyperparameters, the example of information was used to offer a fair evaluation of model fit on the preparation dataset.

Exploration data analysis of visualization

Applied measurements and artificial intelligence (AI) need competence in information perception. In the real world, insights inevitably focus on quantitative representations and judgments of data. An important set of tools for reaching a subjective consensus is provided by information representation. If you're trying to figure out how to analyse a dataset, this may help you identify patterns, degenerate

information, and more. Key relationships in plots or outlines may be conveyed and shown by using information representations, which are more intuitive and understandable than percentages of affiliation or significance. The topic of information representation and exploratory information exploration is a whole in and of itself, and it suggests a deeper dive into some of the publications cited at the conclusion.

Algorithm explanation

Characterization is a method used in artificial intelligence and insights in which a computer programme learns from the data that is sent to it and then uses that knowledge to construct new perceptions. For example, if an individual's gender is determined, or if the mail is classified as either spam or non-spam, this information gathering might be either bi-class (such as determining if the mail is sent to a person of a certain gender) or multi-class. Discourse recognition, penmanship acknowledgment, biometric distinguishing evidence, archive grouping, and so on are examples of characterisation issues. Calculations in Supervised Learning benefit from marked data. After understanding the information, the calculation figures out which name ought to be given to new information dependent

on example and partners the examples to the unlabeled new information.

Conclusion

Data cleansing and processing, missing values, exploratory analysis, and eventually model construction and assessment were all steps in the analytical process. On a public test set, the best accuracy will be determined by the greater accuracy score. Using the phone's sensors, this app can tell you how busy you were.

Future Work

- Human Activity Recognition connect with AI model.
- To automate this procedure by displaying the prediction outcome in a web or desktop application.
- To make the job easier to implement in an Artificial Intelligence context.

REFERENCES:

[1]. Akram Bayat* , Marc Pomplun, Duc A. Tran, " A Study on Human Activity Recognition Using Accelerometer Data from Smartphones".

[2]. Jakaria Rabbi, Md. Tahmid Hasan Fuad, Md. Abdul Awal, "Human Activity Analysis and Recognition from Smartphones using Machine Learning Techniques".

[3]. K.Ishwaryaa , and A.Alice Nithyab, " Human Activity Recognition Methods: A Review".

[4]. Huaijun Wang,^{1,2} Jing Zhao,, " Wearable Sensor-Based Human Activity Recognition Using Hybrid Deep Learning Techniques".

[5]. Ong Chin Ann, Bee Theng Lau, " Human activity recognition: A review ".

HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES USING MACHINE LEARNING PROCESS

ORIGINALITY REPORT

10%	7%	7%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	R Anand, D Aneeshsa, Manisha Srinidhi M, R.K.V Thanuja. "Breast Cancer Prediction in Machine Learning Techniques using Blockchain", 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), 2021 Publication	2%
2	reference.jrank.org Internet Source	2%
3	journals.plos.org Internet Source	1%
4	www.tandfonline.com Internet Source	1%
5	Submitted to Gautam Buddha University Student Paper	1%
6	www.ncbi.nlm.nih.gov Internet Source	1%
7	www.coursehero.com Internet Source	<1%

8	www.researchgate.net Internet Source	<1 %
9	www.ijrte.org Internet Source	<1 %
10	Timothy S. Church, Diana M. Thomas, Catrine Tudor-Locke, Peter T. Katzmarzyk et al. "Trends over 5 Decades in U.S. Occupation-Related Physical Activity and Their Associations with Obesity", PLoS ONE, 2011 Publication	<1 %
11	www.hindawi.com Internet Source	<1 %
12	www.ijitee.org Internet Source	<1 %

Exclude quotes On
Exclude bibliography On

Exclude matches Off

