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

JAHNAVI PRIYA ADDANKI
(Reg. No. 39110021)

NIKHILA MANDALAPU
(Reg. no. 391100597)

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 | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI, CHENNAI - 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

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 **JAHNAVI PRIYA.A(39110021)** and **NIKHILA.M (Reg No:39110597)** who carried out the Project Phase- 2 entitled **"HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES USING MACHINE LEARNING PROCESS"** under my supervision from Jan 2023 to April 2023 .

[illegible]

Internal Guide
Ms. K. DHANALAKSHMI M.E

[illegible]

Head of the Department
Dr.L.LAKSHMANAN M.E.,PhD

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY ★ DEPARTMENT OF CSE

Submitted for Viva-voce Examination held on 20.04.2023

[illegible]

[illegible]

Internal Examiner **External Examiner**

DECLARATION

I, **JAHNAVI PRIYA.A (39110021),** hereby declare that the Project Phase-2 Report entitled **"HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES USING MACHINE LEARNING PROCESS"** done by me under the guidance of **Ms.K. DHANALKSHMI M.E** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering.**

[illegible]

Date : 20.04.2023

PLACE: CHENNAI **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., PhD.**, **Dean**, School of Computing **Dr. L. Lakshmanan M.E., PhD.,** Head 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 **Ms.K.DHANALKSHMI M.E** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project phase-2 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 classify data into activity like Walking, walking up stairs, walking down stairs, sitting, standing, laying are recognized. Sensor data generated using its accelerometer and gyroscope, the sensor signals (accelerometer and gyroscope) were per-processed by applying noise filters.

The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butter worth low-pass filter into body acceleration and gravity. The analysis of data set by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multivariate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset.

To propose a machine learning-based method to accurately predict the stock price Index value by prediction results in the form of stock price increase or stable state best accuracy from comparing supervise classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given transport traffic department dateset with evaluation. dateset 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

APPENDIX

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

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 analytic efforts at Linked In and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data Scientist: Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

Required Skills for a Data Scientist:

- **Programming**: Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning**: Natural Language Processing, Classification, Clustering, ...
- **Data Visualization**: Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms**: MongoDB, Oracle, Microsoft Azure, Cloud era.

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 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.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence.These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes: This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes: This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes: This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural Language Processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural language user interface and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle".

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

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.

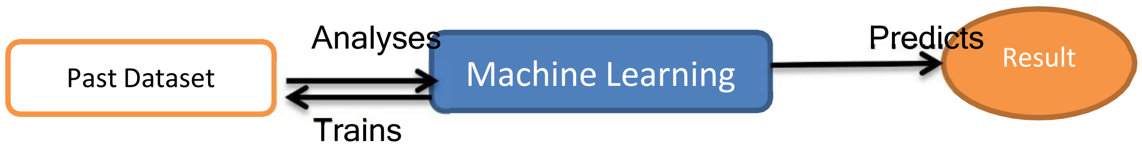


Fig-1.1: Process of Machine learning

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”.

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.

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 Butter worth 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.

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

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

- ➢ To overcome this method to implement machine learning approach by user interface of GUI application
- ➢ Multiple data sets 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.

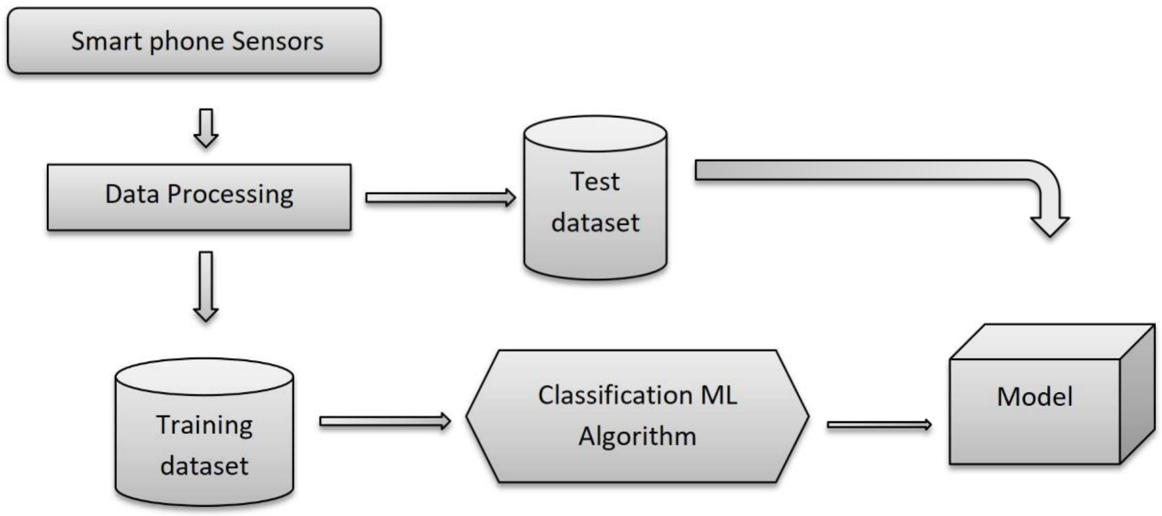


Fig 1.2: Architecture of Proposed model

ADVANTAGES

- These reports are to the investigation of applicability of machine learning techniques for human Activity Recognition in operational conditions.

- Finally, it highlights some observations on future research issues, challenges, and needs

CHAPTER-2
LITERATURE SURVEY

Several studies on human activity recognition have been conducted in the past few years. This section summarizes all of the previous work on facial expression recognition.

Usharani J et al. [1] came up with an idea for a human activity recognition system based on the Android platform. They created an application using the accelerometer data for classification, which supported on-line training and classification. They used the clustered k-NN approach to enhance the performance, accuracy, and execution time of the k-NN classifier with limited resources on the Android platform. They also concluded that the classification times were also dependent on the device models and capabilities.

In Davide Anguita et al.'s [2] paper, they introduced the improvised Support Vector Machine algorithm, which works with fixed point arithmetic to produce an energy-efficient model for the classification of human activities using a smartphone. They aimed to use the presented novel technology for various intelligence applications and smart environments for faster processing with the least possible use of system resources to save the consumption of energy along with maintaining comparable results with other generally used classification techniques.

- To understand people's behavior in different places such as homes, clinics, etc., Md Zia Uddin et al. [3] proposed a body-sensor-based activity recognition system using deep Neural Stretchered Learning based on Long Short-Term Memory (LSTM). For better clustering of features from all the activities, Kernel-based Discriminant Analysis (KDA) was applied, which will maximize inter-class scattering and minimize intra-class scattering of the samples. The proposed model successfully achieved a recall of 99%, which was further compared to the existing deep learning models such as the RNN, Convolutional Neural Network (CNN), and Deep Belief Network (DBN).

Meysam, Vakili et al. [4] proposed a real-time HAR model for online prediction of human physical movements based on the smartphone inertial sensors. A total of 20 different activities were selected, and six incremental learning algorithms were used to check the performance of the system, then all of them were also compared with the state-of-the-art HAR algorithms such as Decision Trees (DTs), AdaBoost, etc. Incremental k-NN and Incremental Naive Bayesian have given the best accuracy of 95%.

In Jirapond Muangprathub et al.'s [5] paper, they introduced a novel elderly person tracking system using a machine learning algorithm. In this work, they used the k-NN model with a k value of 5, which was able to achieve the best accuracy of 96.40% in detecting the real-time activity of elderly people. Furthermore, they created a system that displays information in a spatial format for an elderly person, and in case of an emergency, they can use a messaging device to request any help.

Baoding Zhou et al. [6] proposed a CNN for indoor human activity recognition. A total of nine different activities were recognized based on accelerometer, magnetometers, gyroscopes, and barometers collected by smart phones. The proposed method was able to achieve an excellent accuracy of 98%.

Abdulmajid Murad et al. [7] proposed a deep LSTM network for recognizing six different activities based on smartphone data. The network was able to achieve an accuracy of 96.70% on the UCI-HAD dataset.

2.1 INFERENCES FROM LITREATURE SURVEY

After going through the previous works we inferred some points. In Human activity recognition system based on the Android platform, they used the clustered approach to enhance the performance accuracy, and execution time but the classification times are dependent on the device models and capabilities. Real-time HAR model based on the smartphone inertial sensors used six different machine learning algorithms but they used inertial sensors. Novel elderly person tracking system using a machine learning algorithm. In this work, they used the k-NN model with a k value of 5, which was able to achieve the best accuracy of 96.40% in detecting the real-time activity of elderly people. Davide Anguita et al.'s paper, they introduced the improvised Support Vector Machine algorithm, which works with fixed point arithmetic to produce an energy-efficient model for the classification of human activities using a smartphone. Proposed model successfully achieved a recall of 99%, which was further compared to the existing deep learning models such as the RNN, Convolutional Neural Network (CNN), and Deep Belief Network (DBN). A deep LSTM network for recognizing six different activities based on smartphone data. The network was able to achieve an accuracy of 96.70% on the UCI-HAD dataset

2.2 EXISTING SYSTEM

There is a need to attach sensors to humans for activity recognition. Limited accuracy and availability of sensors on humans. Before HAR the sensors need to be attached. Sensor cost is high and in case of damage to sensor it costs us even more so, it’s not efficient 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).

CHAPTER-3

REQUIREMENT ANALYSIS

AIM :

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.

OBJECTIVE:

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

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.

- 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 multivariate
  - Plot diagram of pair plot, heat map, bar chart and Histogram
- Method of Outline 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.

3.1 FEASIBILITY STUDY

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Pre-processing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be prepossessed so as to improve the efficiency of the algorithm. The outliner have to be removed and also variable conversion need to be done.

Building the classification model

The predicting the Human activity recognition, decision tree algorithm prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in Pre-processing outliner, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data Pre-processing, raw data can't be used direct.Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.

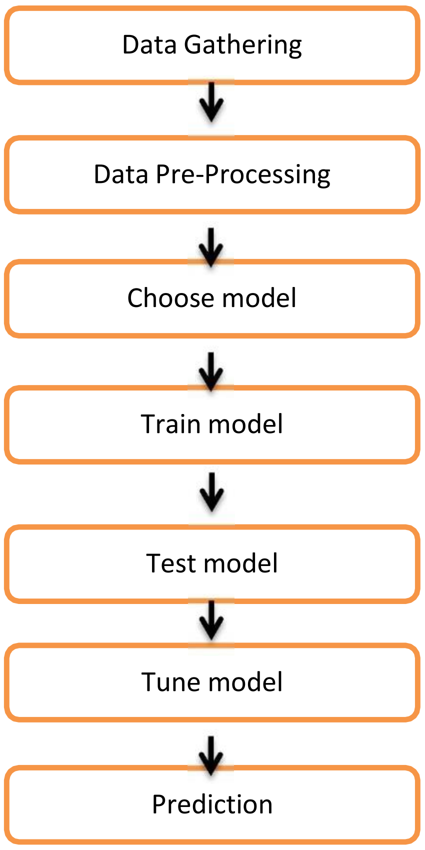


Fig 3.1 : Process of data flow diagram

3.2 HARDWARE AND SOFTWARE REQUIREMENTS:

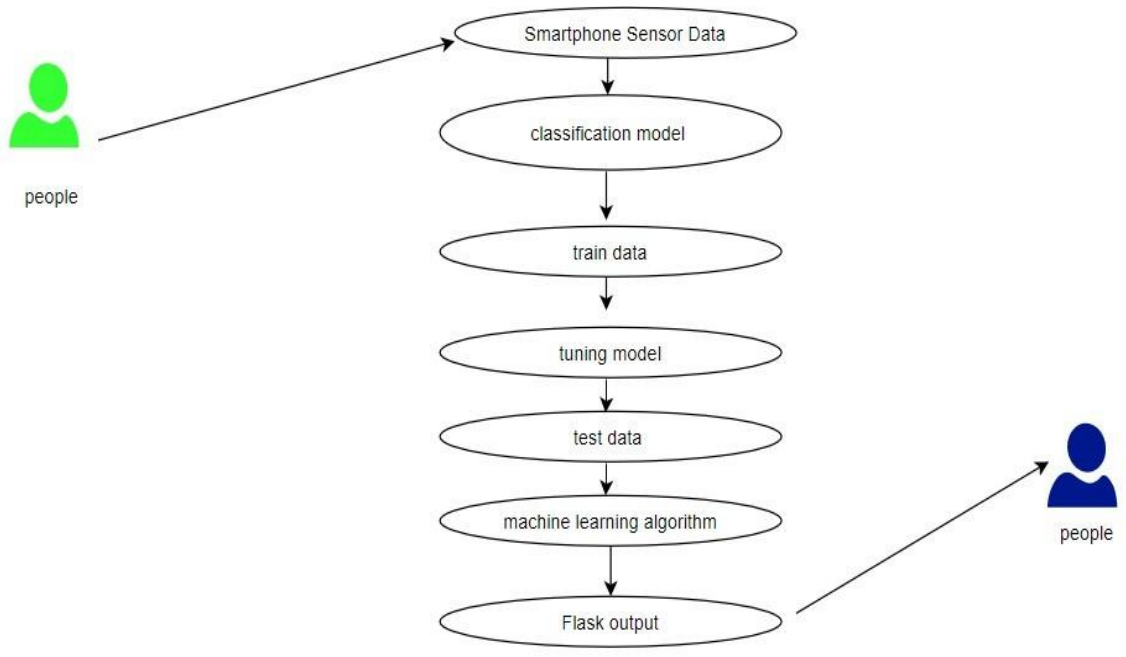
Software Requirements:

Operating System	: Windows
Tool	: Anaconda with Jupyter Notebook

Hardware requirements:

Processor	: Pentium IV/III
Hard disk	: minimum 80 GB
RAM	: minimum 2 GB

3.3 SYSTEM USE CASE



3.2 Use case diagram

CHAPTER-4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

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 Mac OS. 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.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

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.

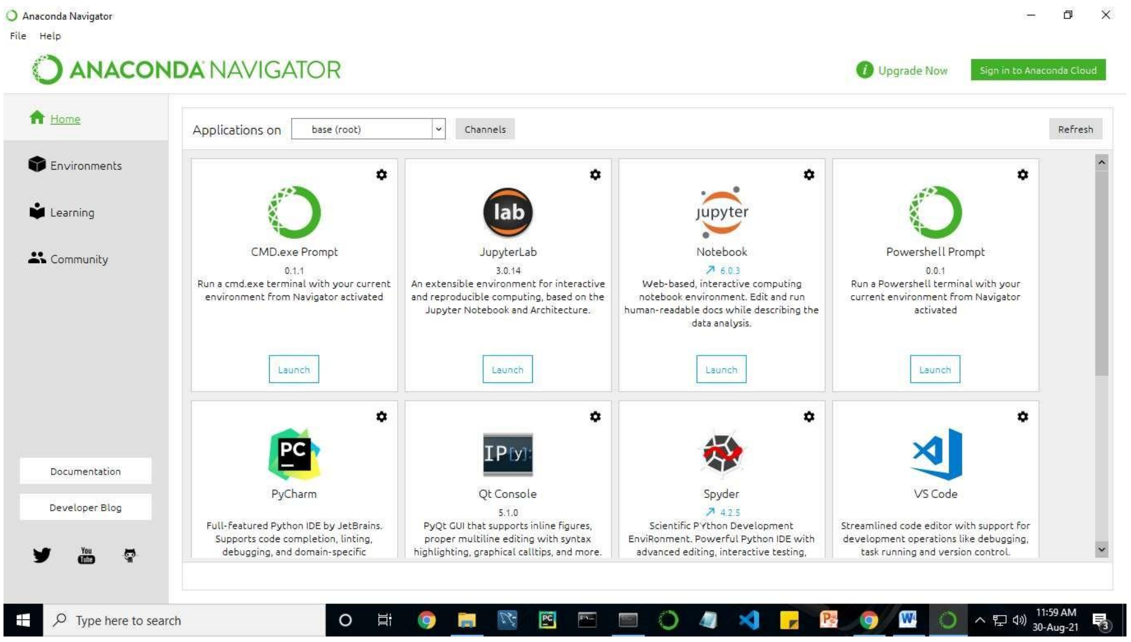
Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytic, and it is a Python distribution that comes pre-install with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytic, 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.



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
3.0.14
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

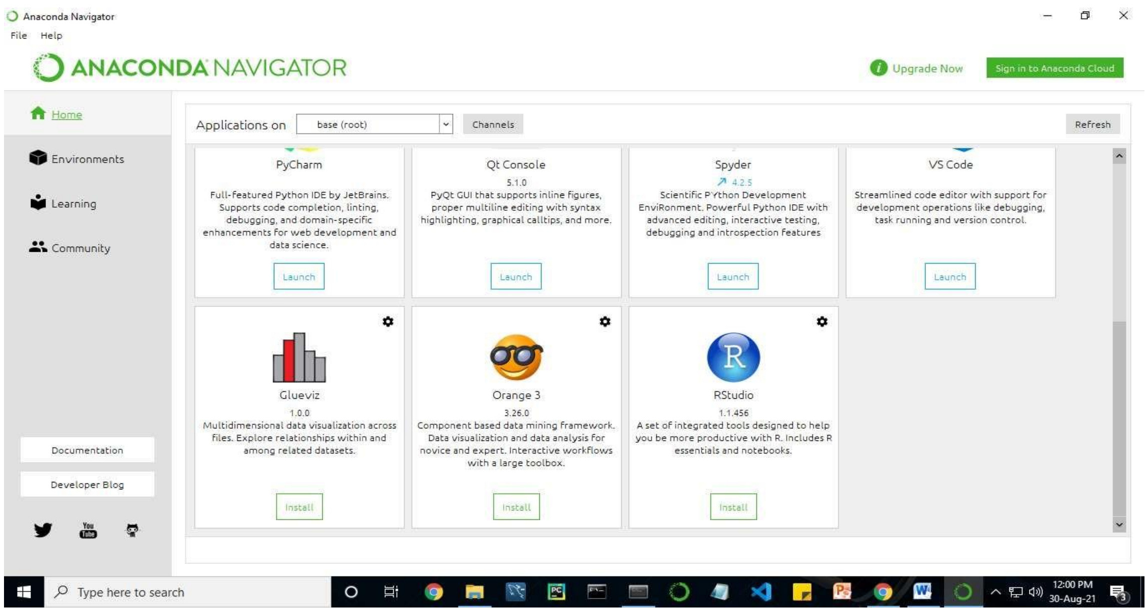
Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

PyCharm
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific

Qt Console
5.1.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
4.2.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing,

VS Code
Streamlined code editor with support for development operations like debugging, task running and version control.



PyCharm
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.

Qt Console
5.1.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
4.2.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz
1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

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.

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 I Python in 2014 by Fernando Perez.

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.

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.

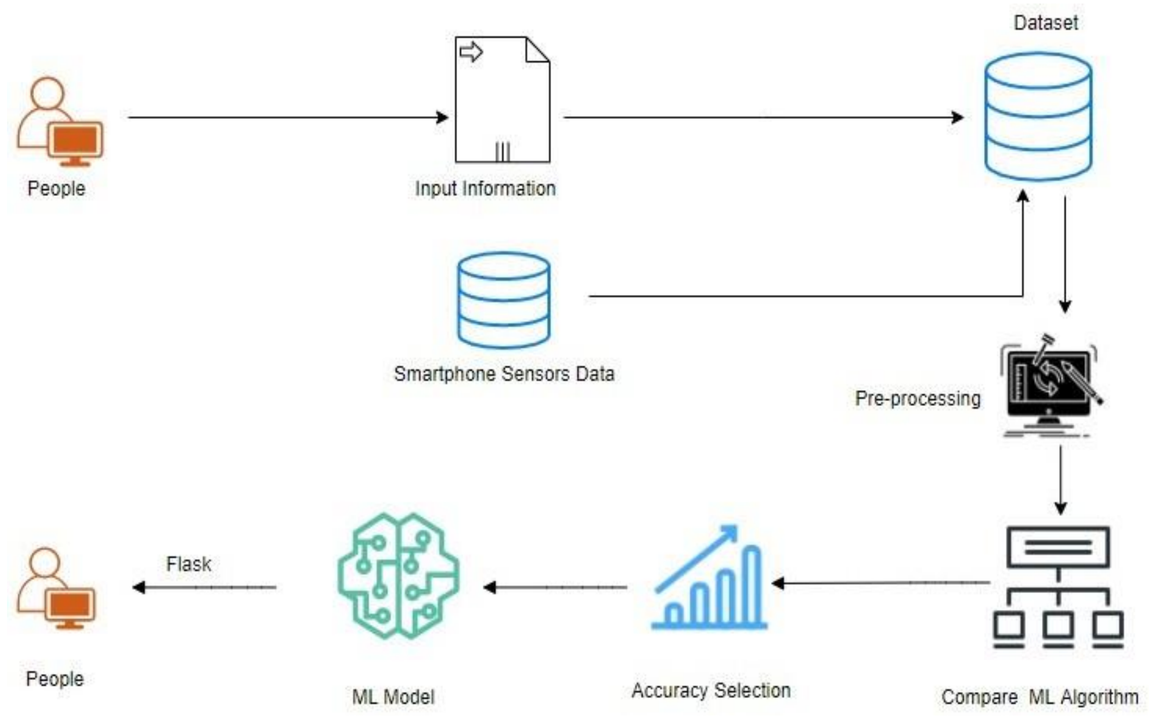
In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.

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

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. 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 sub folders. 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.

System Architecture



CHAPTER-5

IMPLEMENTATION DETAILS

DEVELOPMENT AND DEPLOYMENT SETUP

PYTHON:

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 module 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

WORK FLOW DIAGRAM

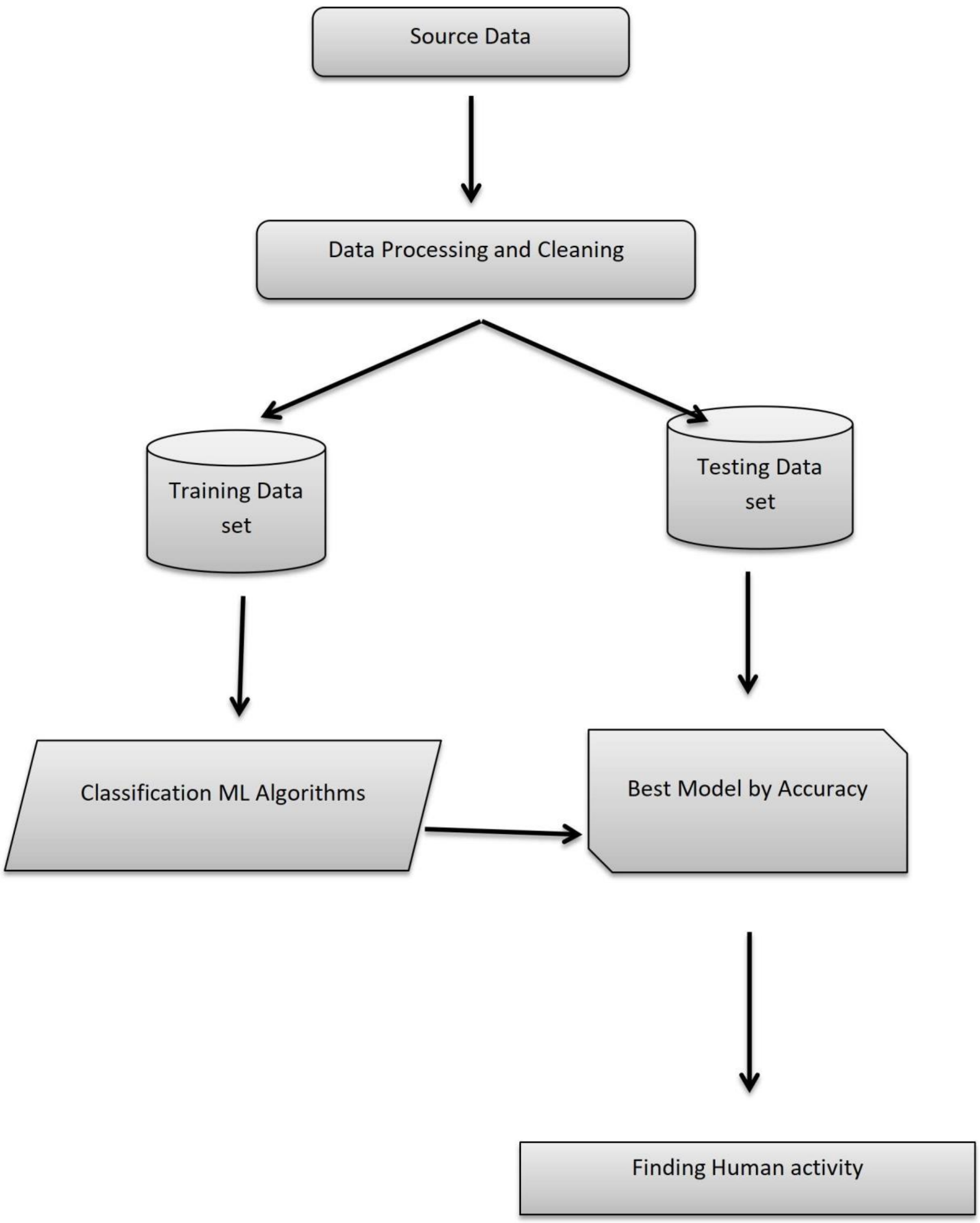
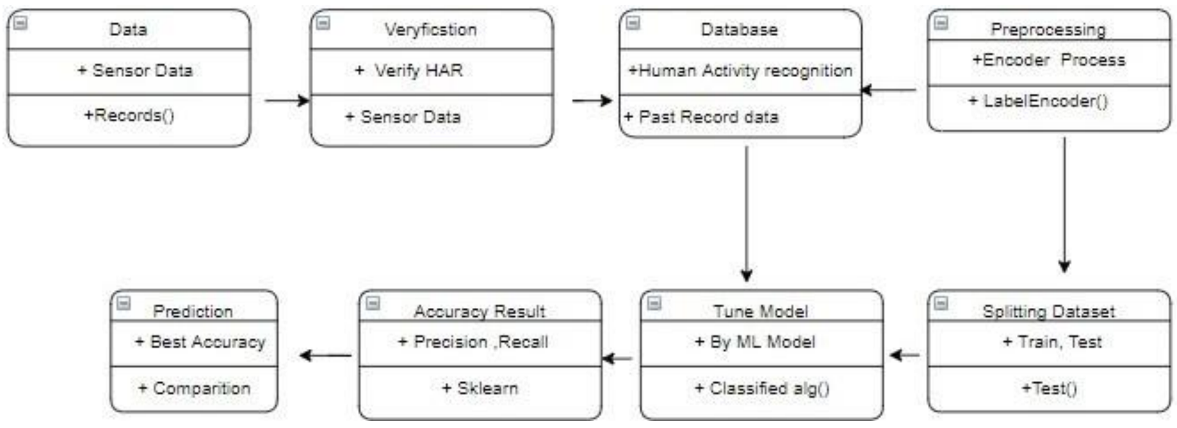


Fig 5.1: Workflow Diagram

CLASS DIAGRAM:

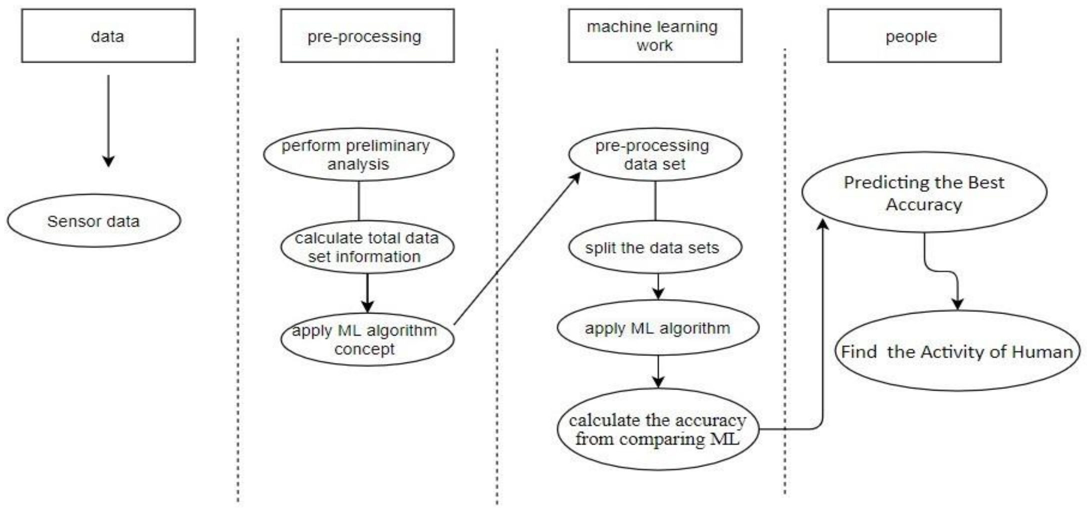


5.2 Class diagram

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.

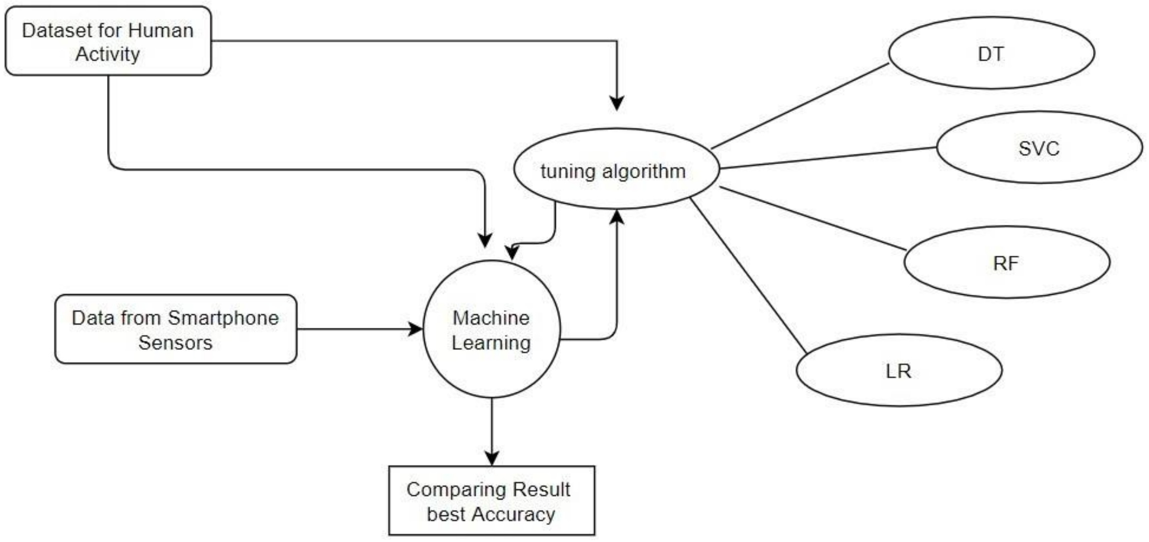
ACTIVITY DIAGRAM:


data
pre-processing
machine learning work
people
Sensor data
perform preliminary analysis
calculate total data set information
apply ML algorithm concept
pre-processing data set
split the data sets
apply ML algorithm
calculate the accuracy from comparing ML
Predicting the Best Accuracy
Find the Activity of Human

5.3 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.

ENTITY RELATIONSHIP DIAGRAM (ERD):


Dataset for Human Activity
DT
SVC
tuning algorithm
RF
Data from Smartphone Sensors
Machine Learning
LR
Comparing Result best Accuracy

5.4 Entity realtionship diagram

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.

5.2 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, Decision Tree Classifier 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:

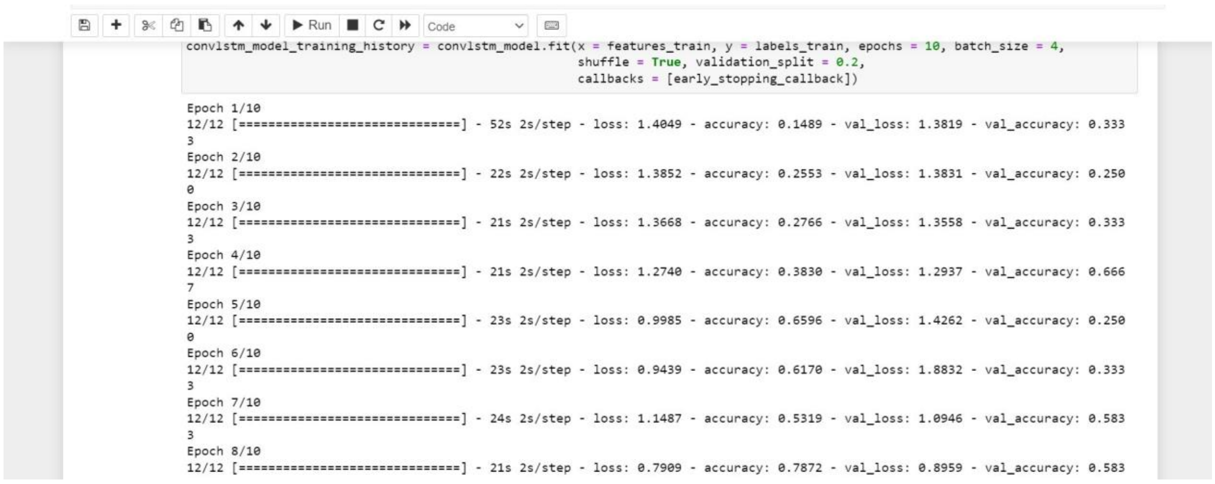
- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.

Logistic Regression

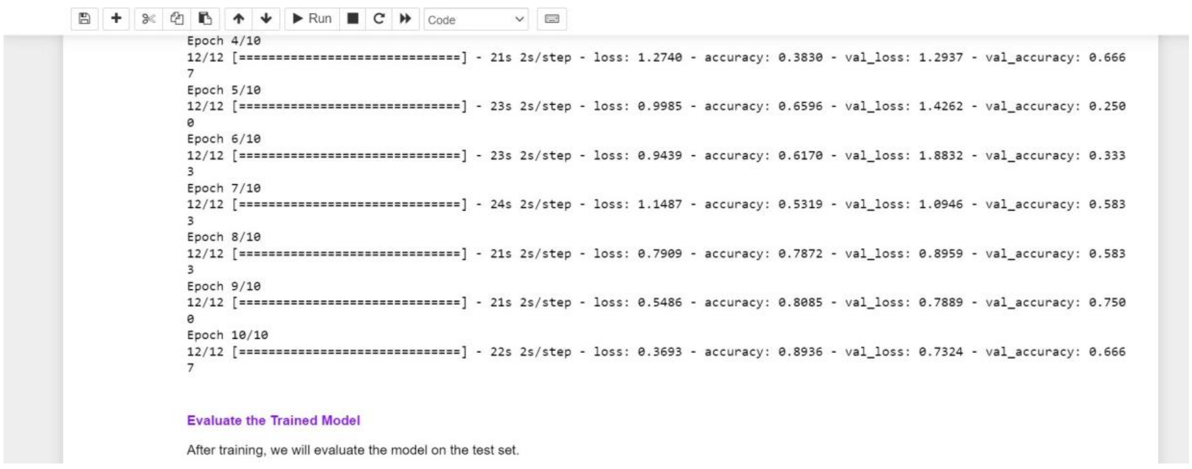
It is a statistical method for analyzing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to 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 (no, failure, etc.).



```
convlstm_model_training_history = convlstm_model.fit(x = features_train, y = labels_train, epochs = 10, batch_size = 4,
                                                     shuffle = True, validation_split = 0.2,
                                                     callbacks = [early_stopping_callback])

Epoch 1/10
12/12 [==============================] - 52s 2s/step - loss: 1.4049 - accuracy: 0.1489 - val_loss: 1.3819 - val_accuracy: 0.333
3
Epoch 2/10
12/12 [==============================] - 22s 2s/step - loss: 1.3852 - accuracy: 0.2553 - val_loss: 1.3831 - val_accuracy: 0.250
0
Epoch 3/10
12/12 [==============================] - 21s 2s/step - loss: 1.3668 - accuracy: 0.2766 - val_loss: 1.3558 - val_accuracy: 0.333
3
Epoch 4/10
12/12 [==============================] - 21s 2s/step - loss: 1.2740 - accuracy: 0.3830 - val_loss: 1.2937 - val_accuracy: 0.666
7
Epoch 5/10
12/12 [==============================] - 23s 2s/step - loss: 0.9985 - accuracy: 0.6596 - val_loss: 1.4262 - val_accuracy: 0.250
0
Epoch 6/10
12/12 [==============================] - 23s 2s/step - loss: 0.9439 - accuracy: 0.6170 - val_loss: 1.8832 - val_accuracy: 0.333
3
Epoch 7/10
12/12 [==============================] - 24s 2s/step - loss: 1.1487 - accuracy: 0.5319 - val_loss: 1.0946 - val_accuracy: 0.583
3
Epoch 8/10
12/12 [==============================] - 21s 2s/step - loss: 0.7909 - accuracy: 0.7872 - val_loss: 0.8959 - val_accuracy: 0.583
```

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.

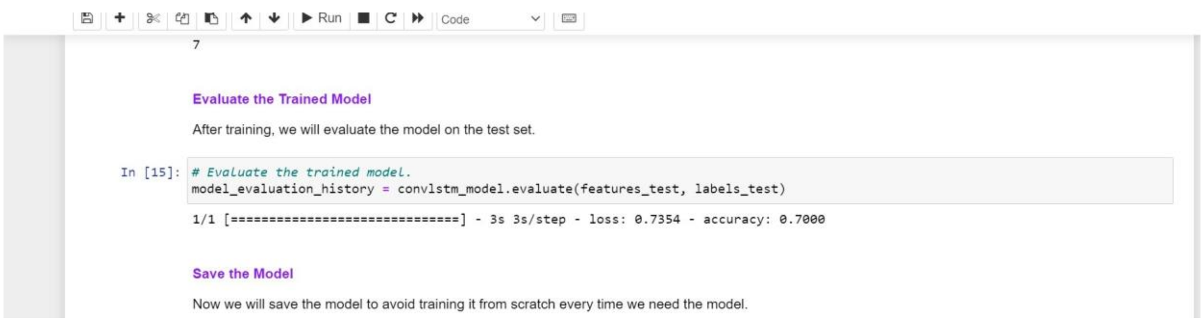


```
Epoch 4/10
12/12 [==============================] - 21s 2s/step - loss: 1.2740 - accuracy: 0.3830 - val_loss: 1.2937 - val_accuracy: 0.666
7
Epoch 5/10
12/12 [==============================] - 23s 2s/step - loss: 0.9985 - accuracy: 0.6596 - val_loss: 1.4262 - val_accuracy: 0.250
0
Epoch 6/10
12/12 [==============================] - 23s 2s/step - loss: 0.9439 - accuracy: 0.6170 - val_loss: 1.8832 - val_accuracy: 0.333
3
Epoch 7/10
12/12 [==============================] - 24s 2s/step - loss: 1.1487 - accuracy: 0.5319 - val_loss: 1.0946 - val_accuracy: 0.583
3
Epoch 8/10
12/12 [==============================] - 21s 2s/step - loss: 0.7909 - accuracy: 0.7872 - val_loss: 0.8959 - val_accuracy: 0.583
3
Epoch 9/10
12/12 [==============================] - 21s 2s/step - loss: 0.5486 - accuracy: 0.8085 - val_loss: 0.7889 - val_accuracy: 0.750
0
Epoch 10/10
12/12 [==============================] - 22s 2s/step - loss: 0.3693 - accuracy: 0.8936 - val_loss: 0.7324 - val_accuracy: 0.666
7
```

Evaluate the Trained Model

After training, we will evaluate the model on the test set.

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.



7

Evaluate the Trained Model

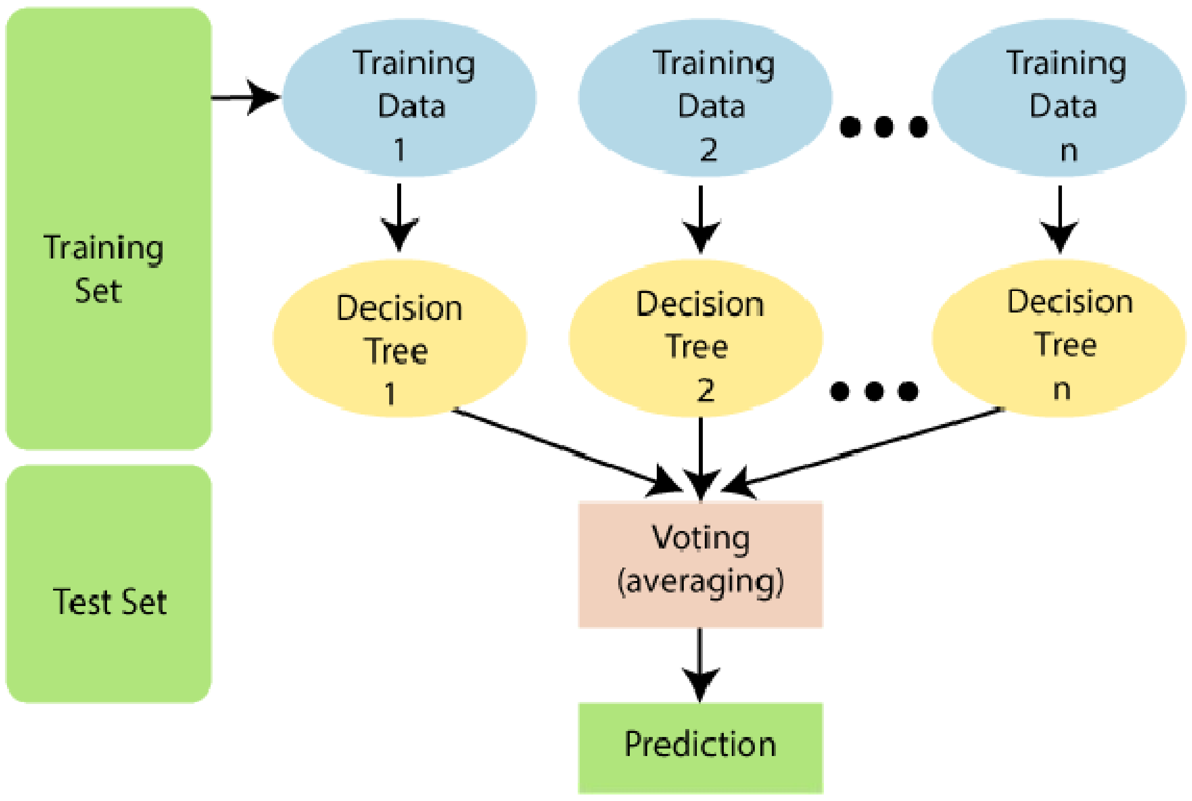
After training, we will evaluate the model on the test set.

```
In [15]: # Evaluate the trained model.
         model_evaluation_history = convlstm_model.evaluate(features_test, labels_test)

1/1 [==============================] - 3s 3s/step - loss: 0.7354 - accuracy: 0.7000
```

Save the Model

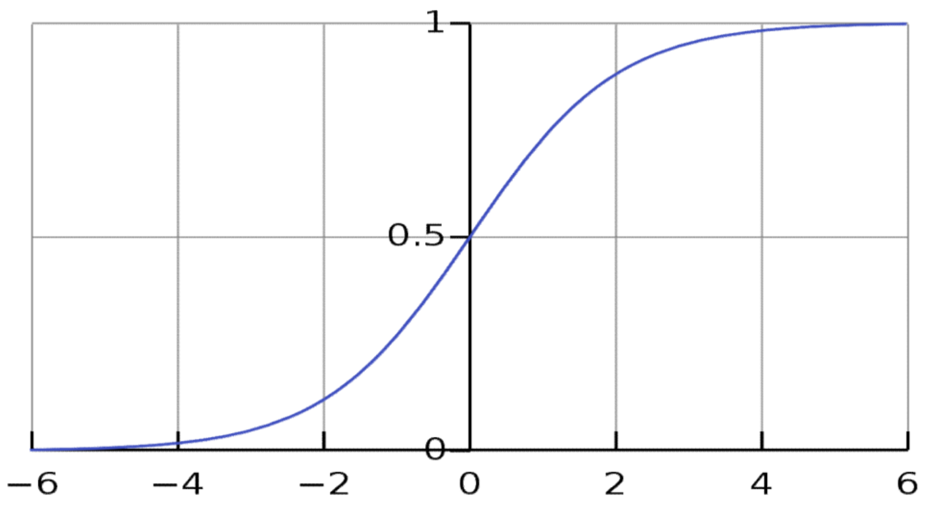
Now we will save the model to avoid training it from scratch every time we need the model.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.The below diagram explains the working of the Random Forest algorithm:



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^-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:

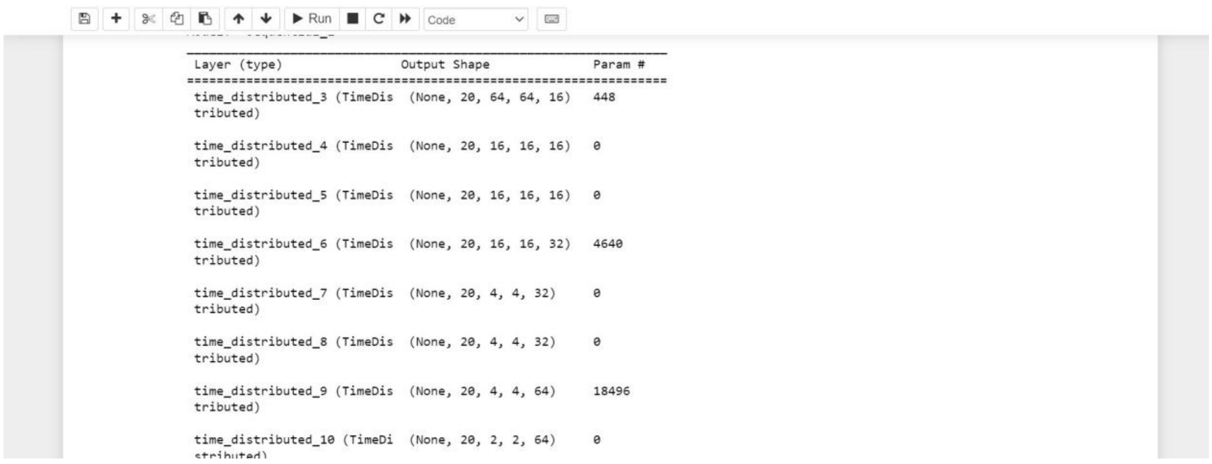
$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

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

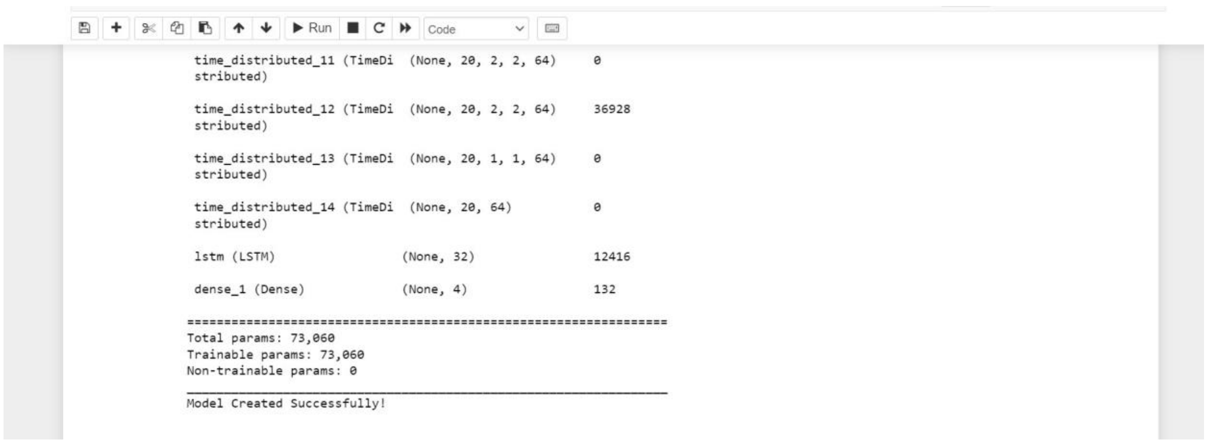
$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

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

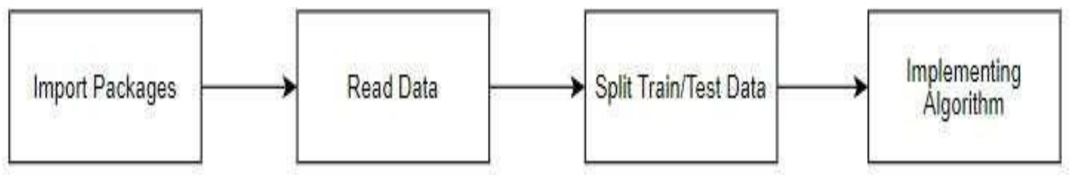


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed_3 (TimeDis  (None, 20, 64, 64, 16)   448
 tributed)

 time_distributed_4 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_5 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_6 (TimeDis  (None, 20, 16, 16, 32)   4640
 tributed)

 time_distributed_7 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_8 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_9 (TimeDis  (None, 20, 4, 4, 64)     18496
 tributed)

 time_distributed_10 (TimeDi  (None, 20, 2, 2, 64)     0
 stributed)
```



```
 time_distributed_11 (TimeDi  (None, 20, 2, 2, 64)     0
 stributed)

 time_distributed_12 (TimeDi  (None, 20, 2, 2, 64)     36928
 stributed)

 time_distributed_13 (TimeDi  (None, 20, 1, 1, 64)     0
 stributed)

 time_distributed_14 (TimeDi  (None, 20, 64)           0
 stributed)

 lstm (LSTM)                 (None, 32)                12416

 dense_1 (Dense)             (None, 4)                 132

=================================================================
Total params: 73,060
Trainable params: 73,060
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully!
```

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

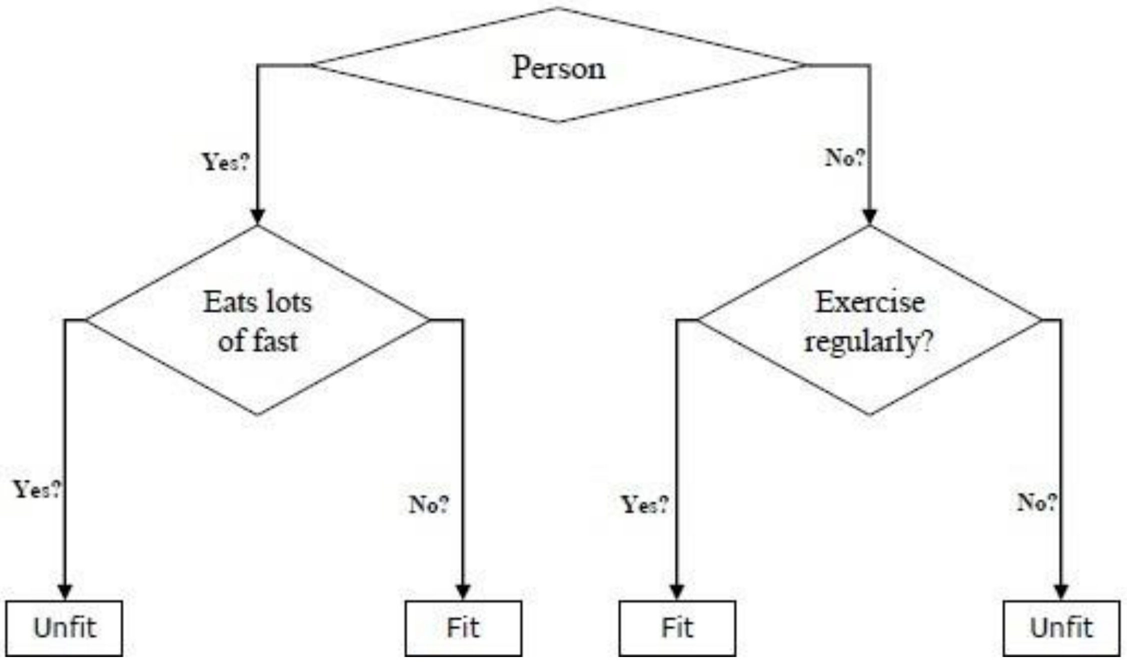
output : getting accuracy

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.

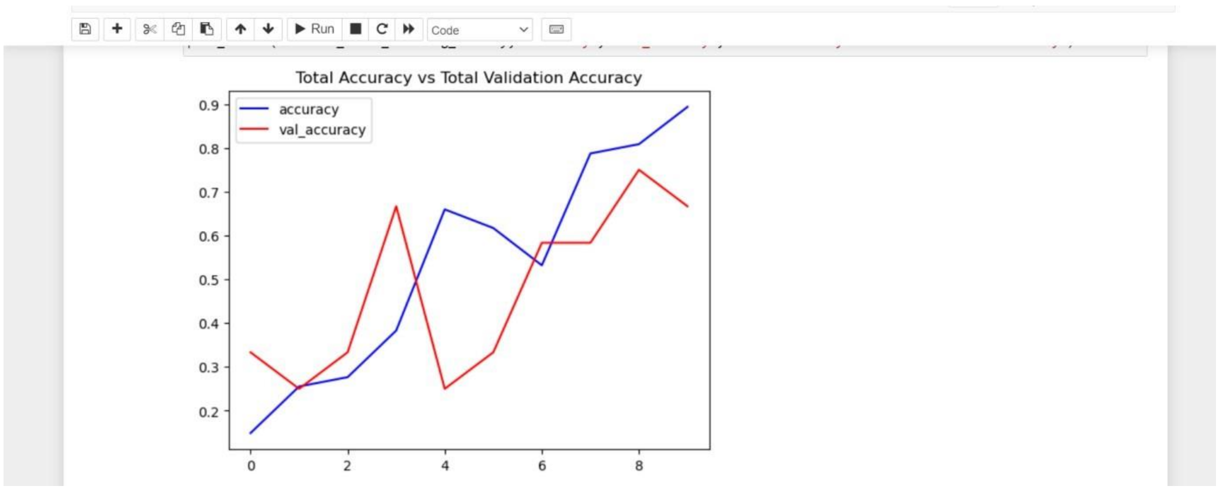
They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits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

Important Terminology related to Decision Tree



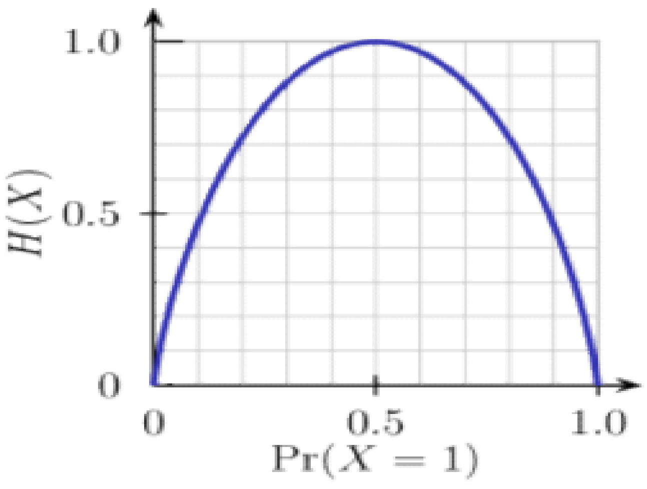
Assumptions while creating Decision Tree

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

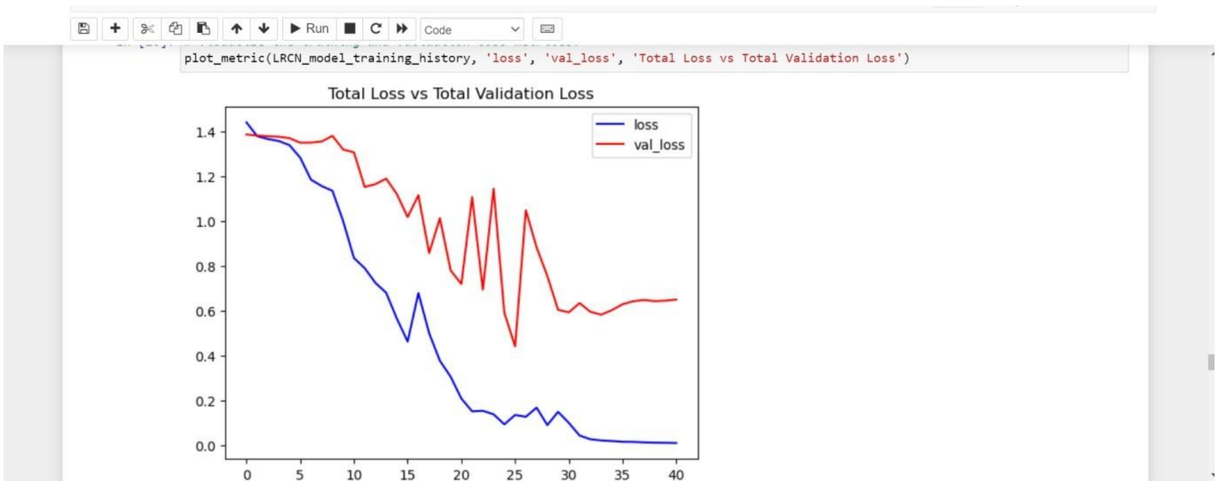
- In the beginning, the whole training set is considered as the **root.**
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- Records are **distributed recursively** on the basis of attribute values.
- Order to placing 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.



```
plot_metric(LRCN_model_training_history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```

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 Pi** → **Probability of an event i of state S or Percentage of class i in a node of state S.**

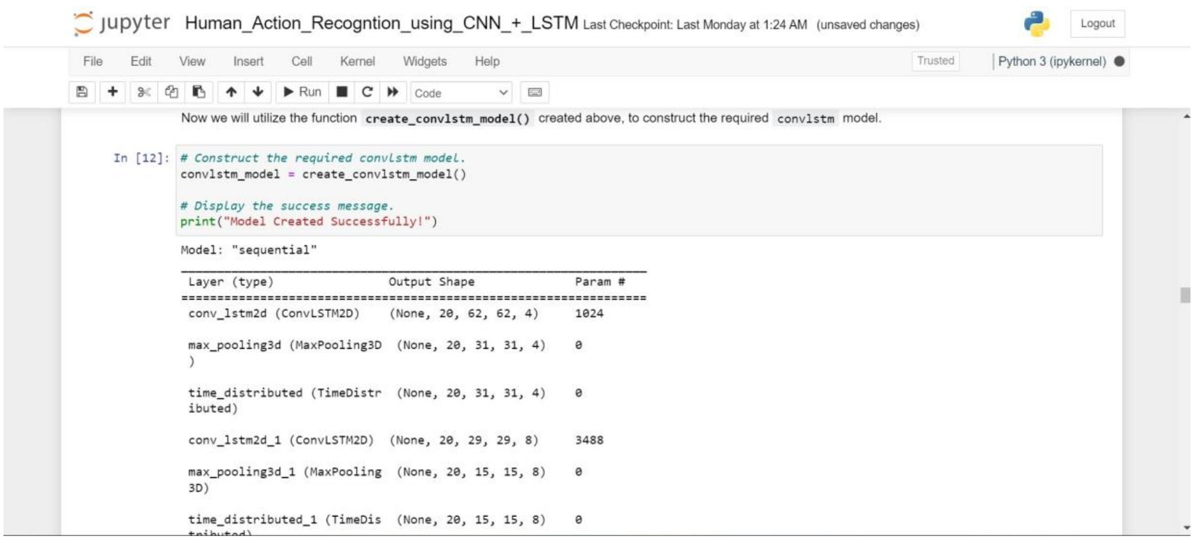
5.3 TESTING

List of Modules:

- Data Pre-processing Technique
- Data analysis of visualization
- Comparing Algorithm with prediction in the form of best accuracy result
- Deployment using Flask

DATA PRE-PROCESSING

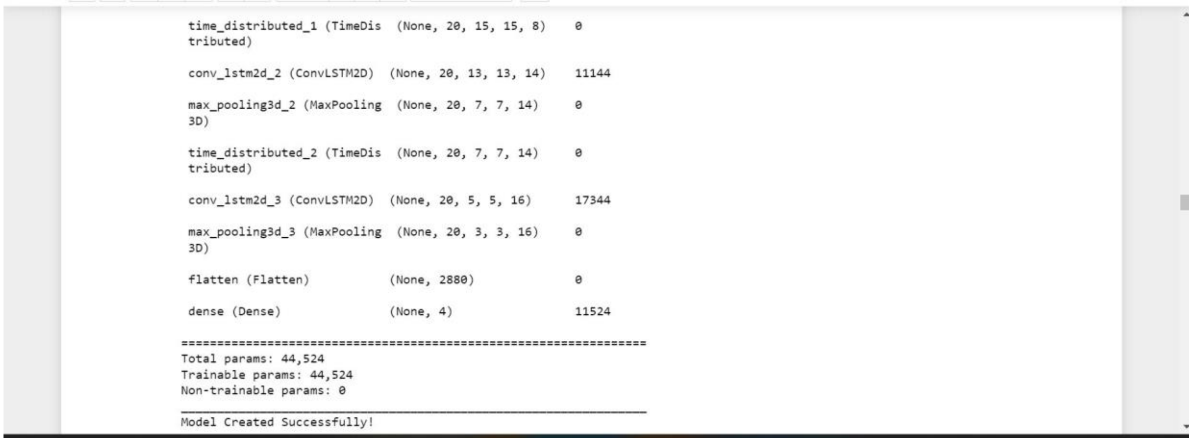
Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, 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.


jupyter
Human_Action_Recogntion_using_CNN_+_LSTM Last Checkpoint: Last Monday at 1:24 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

Now we will utilize the function `create_convlstm_model()` created above, to construct the required `convlstm` model.

```
In [12]: # Construct the required convlstm model.
         convlstm_model = create_convlstm_model()

         # Display the success message.
         print("Model Created Successfully!")
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_lstm2d (ConvLSTM2D)    (None, 20, 62, 62, 4)     1024

 max_pooling3d (MaxPooling3D  (None, 20, 31, 31, 4)    0
 )

 time_distributed (TimeDistr  (None, 20, 31, 31, 4)    0
 ibuted)

 conv_lstm2d_1 (ConvLSTM2D)  (None, 20, 29, 29, 8)     3488

 max_pooling3d_1 (MaxPooling  (None, 20, 15, 15, 8)    0
 3D)

 time_distributed_1 (TimeDis  (None, 20, 15, 15, 8)    0
```



```
 time_distributed_1 (TimeDis  (None, 20, 15, 15, 8)    0
 tributed)

 conv_lstm2d_2 (ConvLSTM2D)  (None, 20, 13, 13, 14)    11144

 max_pooling3d_2 (MaxPooling  (None, 20, 7, 7, 14)     0
 3D)

 time_distributed_2 (TimeDis  (None, 20, 7, 7, 14)     0
 tributed)

 conv_lstm2d_3 (ConvLSTM2D)  (None, 20, 5, 5, 16)      17344

 max_pooling3d_3 (MaxPooling  (None, 20, 3, 3, 16)     0
 3D)

 flatten (Flatten)           (None, 2880)              0

 dense (Dense)               (None, 4)                 11524

=================================================================
Total params: 44,524
Trainable params: 44,524
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully!
```

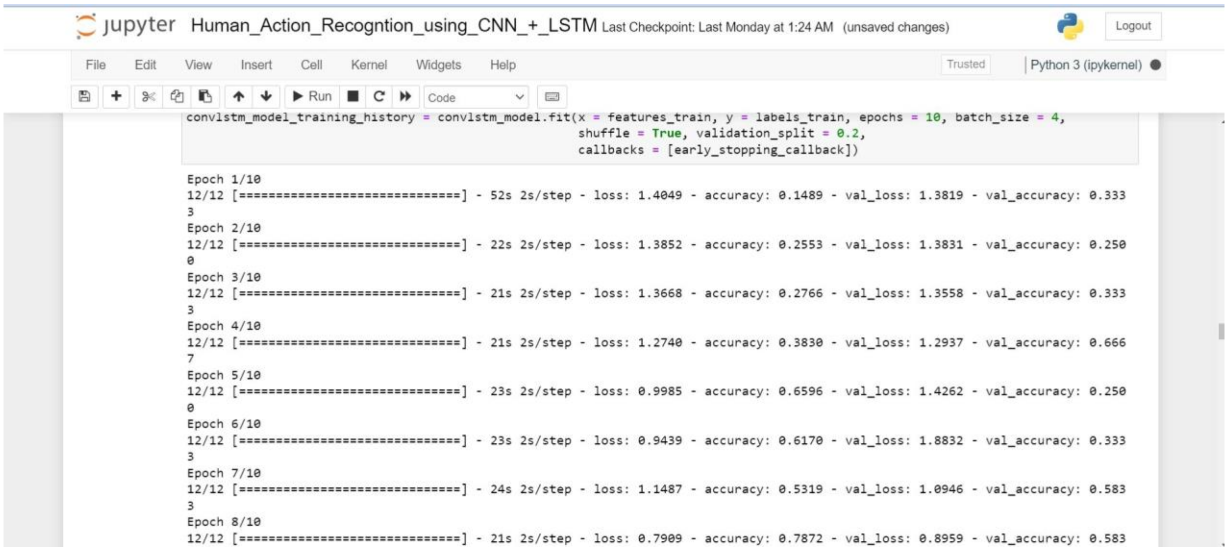
The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, 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.

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:

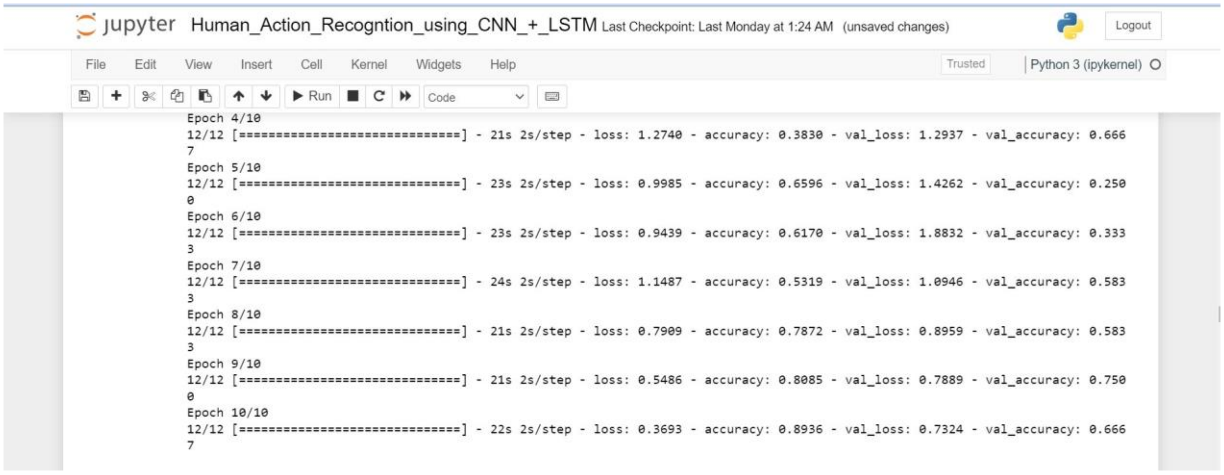
- User forgot to fill in a field.

- 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.


jupyter
Human_Action_Recogntion_using_CNN_+_LSTM Last Checkpoint: Last Monday at 1:24 AM (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code

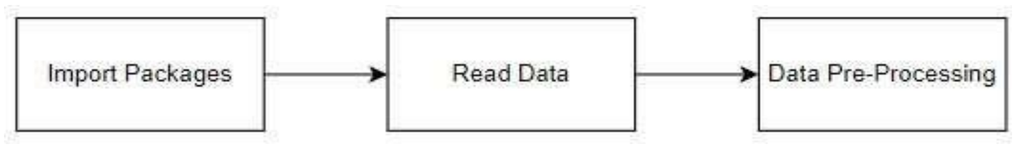
```
convlstm_model_training_history = convlstm_model.fit(x = features_train, y = labels_train, epochs = 10, batch_size = 4,
                                                     shuffle = True, validation_split = 0.2,
                                                     callbacks = [early_stopping_callback])
```

```
Epoch 1/10
12/12 [==============================] - 52s 2s/step - loss: 1.4049 - accuracy: 0.1489 - val_loss: 1.3819 - val_accuracy: 0.333
3
Epoch 2/10
12/12 [==============================] - 22s 2s/step - loss: 1.3852 - accuracy: 0.2553 - val_loss: 1.3831 - val_accuracy: 0.250
0
Epoch 3/10
12/12 [==============================] - 21s 2s/step - loss: 1.3668 - accuracy: 0.2766 - val_loss: 1.3558 - val_accuracy: 0.333
3
Epoch 4/10
12/12 [==============================] - 21s 2s/step - loss: 1.2740 - accuracy: 0.3830 - val_loss: 1.2937 - val_accuracy: 0.666
7
Epoch 5/10
12/12 [==============================] - 23s 2s/step - loss: 0.9985 - accuracy: 0.6596 - val_loss: 1.4262 - val_accuracy: 0.250
0
Epoch 6/10
12/12 [==============================] - 23s 2s/step - loss: 0.9439 - accuracy: 0.6170 - val_loss: 1.8832 - val_accuracy: 0.333
3
Epoch 7/10
12/12 [==============================] - 24s 2s/step - loss: 1.1487 - accuracy: 0.5319 - val_loss: 1.0946 - val_accuracy: 0.583
3
Epoch 8/10
12/12 [==============================] - 21s 2s/step - loss: 0.7909 - accuracy: 0.7872 - val_loss: 0.8959 - val_accuracy: 0.583
```


jupyter
Human_Action_Recogntion_using_CNN_+_LSTM Last Checkpoint: Last Monday at 1:24 AM (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code

```
Epoch 4/10
12/12 [==============================] - 21s 2s/step - loss: 1.2740 - accuracy: 0.3830 - val_loss: 1.2937 - val_accuracy: 0.666
7
Epoch 5/10
12/12 [==============================] - 23s 2s/step - loss: 0.9985 - accuracy: 0.6596 - val_loss: 1.4262 - val_accuracy: 0.250
0
Epoch 6/10
12/12 [==============================] - 23s 2s/step - loss: 0.9439 - accuracy: 0.6170 - val_loss: 1.8832 - val_accuracy: 0.333
3
Epoch 7/10
12/12 [==============================] - 24s 2s/step - loss: 1.1487 - accuracy: 0.5319 - val_loss: 1.0946 - val_accuracy: 0.583
3
Epoch 8/10
12/12 [==============================] - 21s 2s/step - loss: 0.7909 - accuracy: 0.7872 - val_loss: 0.8959 - val_accuracy: 0.583
3
Epoch 9/10
12/12 [==============================] - 21s 2s/step - loss: 0.5486 - accuracy: 0.8085 - val_loss: 0.7889 - val_accuracy: 0.750
0
Epoch 10/10
12/12 [==============================] - 22s 2s/step - loss: 0.3693 - accuracy: 0.8936 - val_loss: 0.7324 - val_accuracy: 0.666
7
```

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

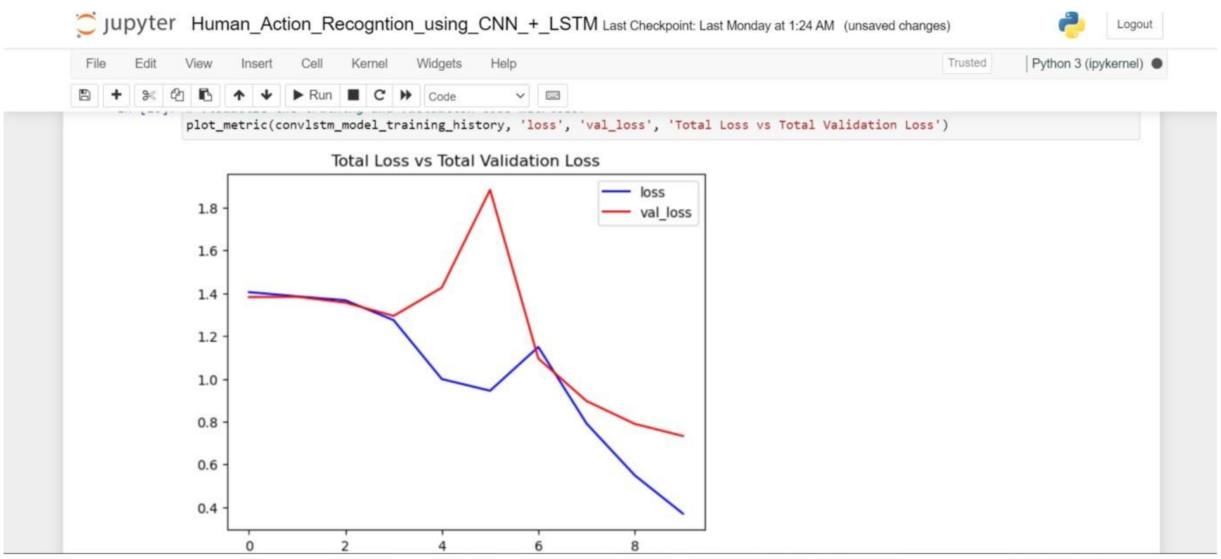
input : data

output : removing noisy data

DATA VALIDATION/ CLEANING/PREPARING PROCESS

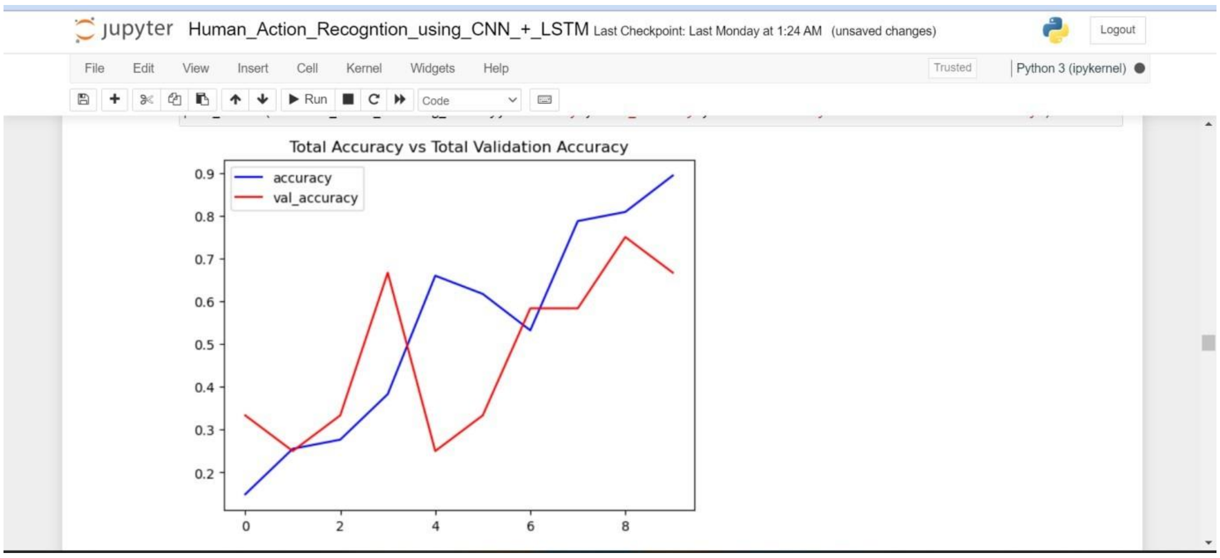
Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill 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 analytic and decision making.

Exploration data analysis of visualization

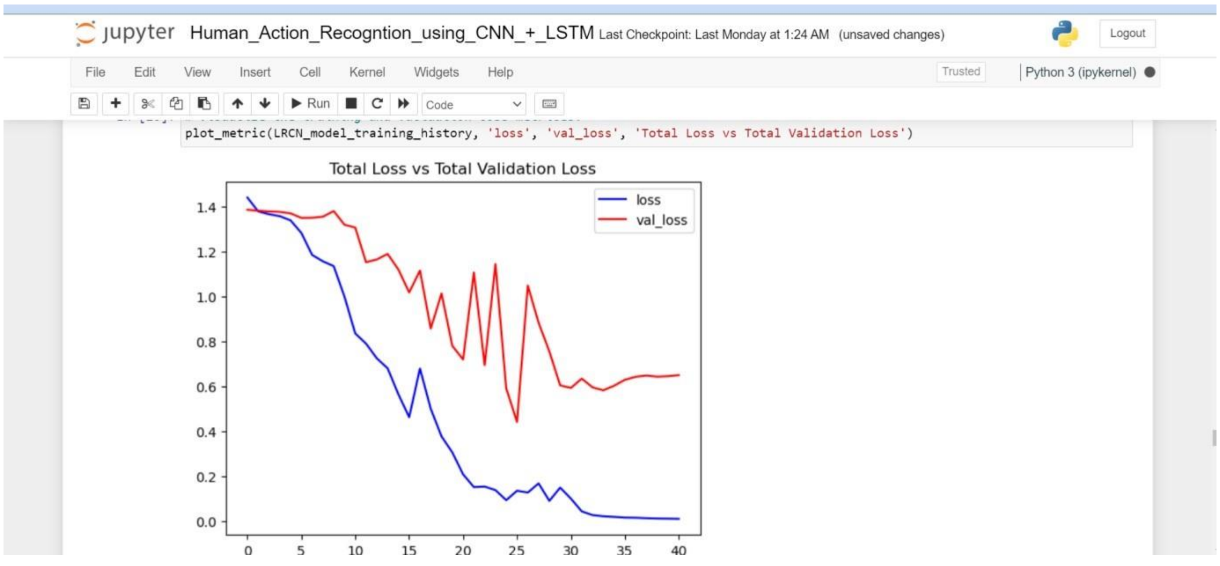

jupyter Human_Action_Recogntion_using_CNN_+_LSTM Last Checkpoint: Last Monday at 1:24 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Total Loss vs Total Validation Loss
loss
val_loss

```
plot_metric(convlstm_model_training_history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```

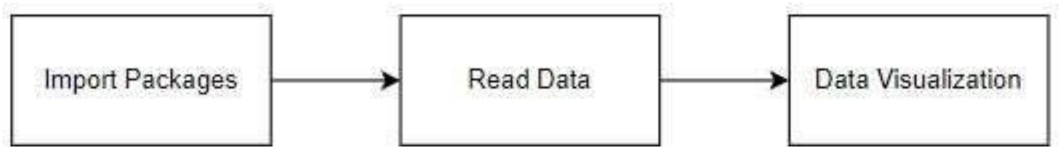
Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and 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.


```
plot_metric(LRCN_model_training_history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```
jupyter Human_Action_Recogntion_using_CNN_+_LSTM Last Checkpoint: Last Monday at 1:24 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Total Loss vs Total Validation Loss
loss
val_loss

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 sci-kit-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,

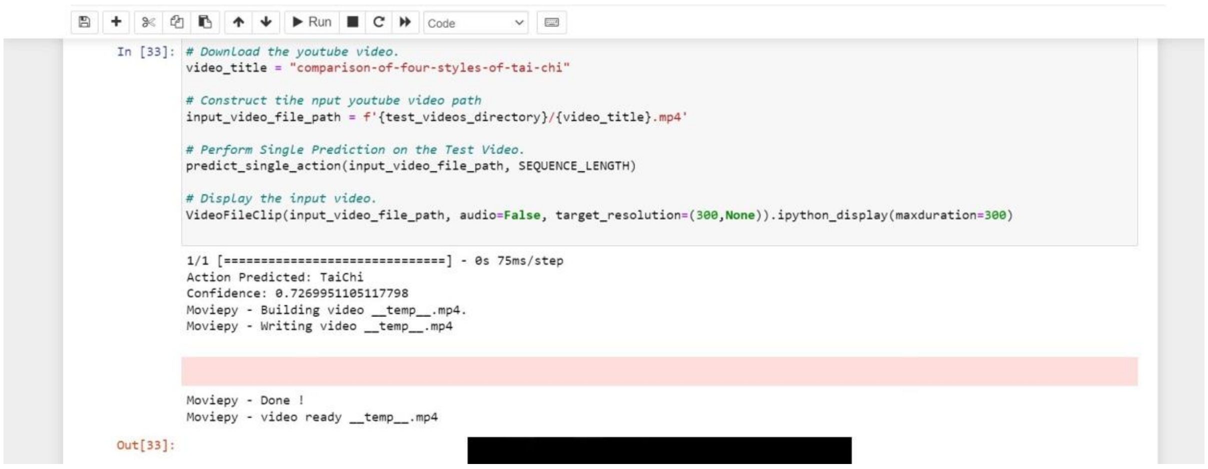
It is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose 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 have to done preprocessing, linear model with logistic regression method, cross validating by K Fold 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.



```
In [33]: # Download the youtube video.
         video_title = "comparison-of-four-styles-of-tai-chi"

         # Construct tihe nput youtube video path
         input_video_file_path = f'{test_videos_directory}/{video_title}.mp4'

         # Perform Single Prediction on the Test Video.
         predict_single_action(input_video_file_path, SEQUENCE_LENGTH)

         # Display the input video.
         VideoFileClip(input_video_file_path, audio=False, target_resolution=(300,None)).ipython_display(maxduration=300)

         1/1 [==============================] - 0s 75ms/step
         Action Predicted: TaiChi
         Confidence: 0.7269951105117798
         Moviepy - Building video __temp__.mp4.
         Moviepy - Writing video __temp__.mp4


         Moviepy - Done !
         Moviepy - video ready __temp__.mp4
Out[33]:
```

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.

True Positive Rate(TPR) = TP / (TP + FN)

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

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

ACCURACY CALCULATION:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.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?)

Precision = TP / (TP + FP)

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. 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)

Recall = TP / (TP + 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:

F- Measure = 2TP / (2TP + FP + FN)

F1-SCORE FORMULA:

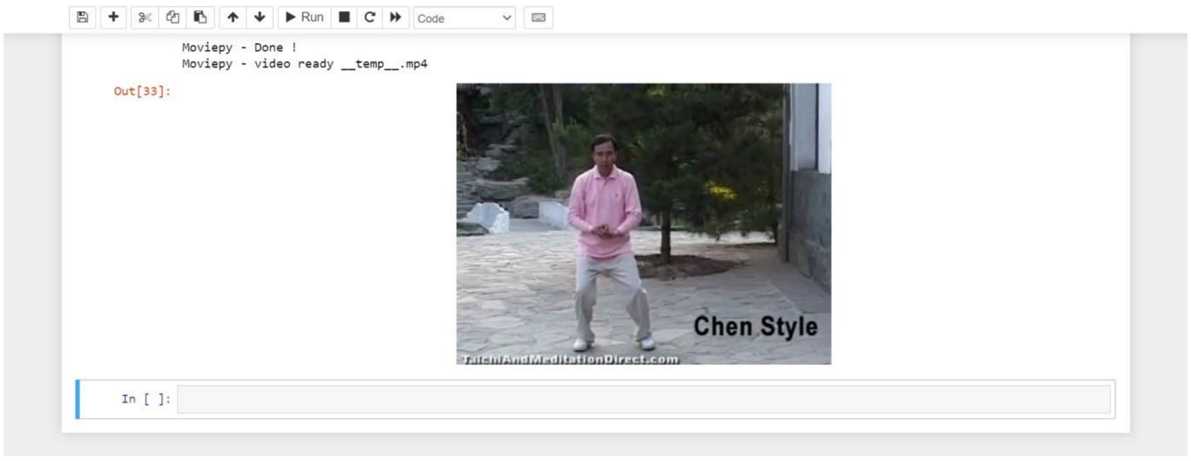
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

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.



```
        Moviepy - Done !
        Moviepy - video ready __temp__.mp4
Out[33]:
```

```
In [ ]:
```

CHAPTER-6

RESULTS AND DISCUSSION

The camera detects the human in the first place and then the activity that is taking place. The deep learning model used is ResNet which typically uses 2D kernels but to enhance the efficiency of activity recognition 3D kernels are enabled. The activity recognition model is trained on kinetics400 dataset which identifies 400 different types of activities. And to increase the accuracy 400 videos of each activity are present.

The ResNet model for human activity recognition with 2D CNN uses 152 layers for activity identification. This 2D CNN ResNet model performs tasks like detection, segmentation and captioning but our model has additional 3D CNN which processes videos unlike images in 2D CNN which also has 152 layers which increases the chances of success of action recognition. By following tasks like detection, summarization, optical flow, segmentation and captioning.

As a result higher accuracy in human activity recognition is achieved.

CHAPTER-7
CONCLUSION

7.1 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.

Human activity analysis is a popular activity in the growing industry and we have applied different machine learning algorithm. Comparative study performed among the applied various techniques KNN, SVM, Random forest, Neural Networks, Logistic regression and Naive Bayes.

In them, Logistic Regression and neural network gave good results whereas Naive Bayes result was not good. The implementation of Neural Network on Python gave better results than the one provided in the Orange tool.The limitations of this work is though the efficiency of neural network is good, the model is not dynamic.

The inability of getting trained with real time data will force us to train the model everytime new data comes.In future, these results can be used for making smart watches and similar devices which can track a user's activity and notify him/her of the daily activity log. They can also be used for monitoring elderly people, prison inmates, or anyone who needs constant supervision.

7.2 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. Aminian, K and Robert, Ph and Buchser, EE and Rutschmann, B and Hayoz, D and Depairon, M Physical activity monitoring based on accelerometer: validation and comparison with video observation, Medical & biological engineering & computing, 1999; 37:3-304

2. Bao, Ling and Intille, Stephen S, Pervasive computing, Activity recognition from user-annotated acceleration data, 2004, Springer.

3. Casale, Pierluigi and Pujol, Oriol and Radeva, Petia, Human activity recognition from accelerometer data using a wearable device, Pattern Recognition and Image Analysis,2011;289, Springer.

APPENDIX

SOURCE CODE

Step 1: Visualize the Data with its Labels

Step 2: Preprocess the Dataset

Step 3: Split the Data into Train and Test Set

Step 4: Implement the ConvLSTM Approach

Step 4.1: Construct the Model

Step 4.2: Compile & Train the Model

Step 4.3: Plot Model's Loss & Accuracy Curves

Step 5: implement the LRCN Approach

Step 5.1: Construct the Model

Step 5.2: Compile & Train the Model

Step 5.3: Plot Model's Loss & Accuracy Curves

Step 6: Test the Best Performing Model on YouTube videos

```
# Discard the output of this cell.
#%%capture

# Install the required libraries.
!pip install tensorflow opencv-contrib-python youtube-dl moviepy pydot
!pip install git+https://github.com/TahaAnwar/pafy.git#egg=pafy
# Import the required libraries.
import os
import cv2
import pafy
import math
import  random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt

from moviepy.editor import *
%matplotlib inline

from sklearn.model_selection import train_test_split

from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
seed_constant = 27
np.random.seed(seed_constant)
random.seed(seed_constant)
tf.random.set_seed(seed_constant)
# Specify the height and width to which each video frame will be resized in our datas
et.
IMAGE_HEIGHT , IMAGE_WIDTH = 64, 64

# Specify the number of frames of a video that will be fed to the model as one seque
nce.
SEQUENCE_LENGTH = 20

# Specify the directory containing the UCF50 dataset.
```

```
DATASET_DIR = "dataset/UCF50"


# Specify the list containing the names of the classes used for training. Feel free to c
hoose any set of classes.
CLASSES_LIST = ["WalkingWithDog", "TaiChi", "Swing", "HorseRace"]
def frames_extraction(video_path):
    '''
    This function will extract the required frames from a video after resizing and norma
lizing them.
    Args:
        video_path: The path of the video in the disk, whose frames are to be extracted.
    Returns:
        frames_list: A list containing the resized and normalized frames of the video.
    '''


    # Declare a list to store video frames.
    frames_list = []


    # Read the Video File using the VideoCapture object.
    video_reader = cv2.VideoCapture(video_path)


    # Get the total number of frames in the video.
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))


    # Calculate the the interval after which frames will be added to the list.
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)


    # Iterate through the Video Frames.
    for frame_counter in range(SEQUENCE_LENGTH):
        # Set the current frame position of the video.

        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frame
s_window)


        # Reading the frame from the video.

        success, frame = video_reader.read()


        # Check if Video frame is not successfully read then break the loop

        if not success:

            break


        # Resize the Frame to fixed height and width.

        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))


        # Normalize the resized frame by dividing it with 255 so that each pixel value th
en lies between 0 and 1

        normalized_frame = resized_frame / 255


        # Append the normalized frame into the frames list

        frames_list.append(normalized_frame)


    # Release the VideoCapture object.

    video_reader.release()


    # Return the frames list.

    return frames_list

def create_dataset():

    '''

    This function will extract the data of the selected classes and create the required d
ataset.
```

```
    Returns:
        features:          A list containing the extracted frames of the videos.
        labels:            A list containing the indexes of the classes associated with the vid
eos.
        video_files_paths: A list containing the paths of the videos in the disk.
    '''

    # Declared Empty Lists to store the features, labels and video file path values.
    features = []
    labels = []
    video_files_paths = []

    # Iterating through all the classes mentioned in the classes list
    for class_index, class_name in enumerate(CLASSES_LIST):

        # Display the name of the class whose data is being extracted.
        print(f'Extracting Data of Class: {class_name}')

        # Get the list of video files present in the specific class name directory.
        files_list = os.listdir(os.path.join(DATASET_DIR, class_name))

        # Iterate through all the files present in the files list.
        for file_name in files_list:

            # Get the complete video path.
            video_file_path = os.path.join(DATASET_DIR, class_name, file_name)

            # Extract the frames of the video file.
            frames = frames_extraction(video_file_path)
            # Check if the extracted frames are equal to the SEQUENCE_LENGTH speci
fied above.
            # So ignore the vides having frames less than the SEQUENCE_LENGTH.
            if len(frames) == SEQUENCE_LENGTH:

                # Append the data to their repective lists.
                features.append(frames)
                labels.append(class_index)
                video_files_paths.append(video_file_path)

    # Converting the list to numpy arrays
    features = np.asarray(features)
    labels = np.array(labels)

    # Return the frames, class index, and video file path.
    return features, labels, video_files_paths
# Create the dataset.
features, labels, video_files_paths = create_dataset()
# Using Keras's to_categorical method to convert labels into one-hot-
encoded vectors
one_hot_encoded_labels = to_categorical(labels)
# Split the Data into Train ( 75% ) and Test Set ( 25% ).
features_train, features_test, labels_train, labels_test = train_test_split
(features, one_hot_encoded_labels,
 test_size = 0.25, shuffle = True,
random_state = seed_constant)
def create_convlstm_model():
    '''
    This function will construct the required convlstm model.
    Returns:
        model: It is the required constructed convlstm model.
    '''

    # We will use a Sequential model for model construction
    model = Sequential()

    # Define the Model Architecture.
    ########################################################################
#################################################################

    model.add(ConvLSTM2D(filters = 4, kernel_size = (3, 3), activation = 'tanh',data_f
ormat = "channels_last",
                         recurrent_dropout=0.2, return_sequences=True, input_shape = (SE
QUENCE_LENGTH,
                                                                              IMAGE_HEIGHT, IMAGE_WI
DTH, 3)))

    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='chan
nels_last'))
    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 8, kernel_size = (3, 3), activation = 'tanh', data_f
ormat = "channels_last",
                         recurrent_dropout=0.2, return_sequences=True))

    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='chan
nels_last'))
    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 14, kernel_size = (3, 3), activation = 'tanh', data
_format = "channels_last",
                         recurrent_dropout=0.2, return_sequences=True))
    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='chan
nels_last'))
    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters = 16, kernel_size = (3, 3), activation = 'tanh', data
_format = "channels_last",
                         recurrent_dropout=0.2, return_sequences=True))

    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='chan
nels_last'))
    #model.add(TimeDistributed(Dropout(0.2)))

    model.add(Flatten())

    model.add(Dense(len(CLASSES_LIST), activation = "softmax"))

    ###################################################################
#########################################################

    # Display the models summary.
    model.summary()

    # Return the constructed convlstm model.
    return model
# Construct the required convlstm model.
convlstm_model = create_convlstm_model()

# Display the success message.
print("Model Created Successfully!")
# Plot the structure of the contructed model.
```

```
plot_model(convlstm_model, to_file = 'convlstm_model_structure_plot.png', show_sh
apes = True, show_layer_names = True)

# Create an Instance of Early Stopping Callback

early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 10, mode =
 'min', restore_best_weights = True)


# Compile the model and specify loss function, optimizer and metrics values to the m
odel

convlstm_model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam', metri
cs = ["accuracy"])


# Start training the model.

convlstm_model_training_history = convlstm_model.fit(x = features_train, y = labels_
train, epochs = 50, batch_size = 4, shuffle = True, validation_split = 0.2,

    callbacks = [early_stopping_callback])

# Evaluate the trained model.

model_evaluation_history = convlstm_model.evaluate(features_test, labels_test)

# Get the loss and accuracy from model_evaluation_history.

model_evaluation_loss, model_evaluation_accuracy = model_evaluation_history


# Define the string date format.

# Get the current Date and Time in a DateTime Object.

# Convert the DateTime object to string according to the style mentioned in date_tim
e_format string.

date_time_format = '%Y_%m_%d__%H_%M_%S'

current_date_time_dt = dt.datetime.now()

current_date_time_string = dt.datetime.strftime(current_date_time_dt, date_time_for
mat)


# Define a useful name for our model to make it easy for us while navigating through
multiple saved models.

model_file_name = f'convlstm_model___Date_Time_{current_date_time_string}___L
oss_{model_evaluation_loss}___Accuracy_{model_evaluation_accuracy}.h5'
```

```
# Save your Model.
convlstm_model.save(model_file_name)
def plot_metric(model_training_history, metric_name_1, metric_name_2, plot_name):
    '''
    This function will plot the metrics passed to it in a graph.
    Args:
        model_training_history: A history object containing a record of training and valid
ation
                                loss values and metrics values at successive epochs
        metric_name_1:          The name of the first metric that needs to be plotted in th
e graph.
        metric_name_2:          The name of the second metric that needs to be plotted i
n the graph.
        plot_name:              The title of the graph.
    '''

    # Get metric values using metric names as identifiers.
    metric_value_1 = model_training_history.history[metric_name_1]
    metric_value_2 = model_training_history.history[metric_name_2]

    # Construct a range object which will be used as x-
axis (horizontal plane) of the graph.
    epochs = range(len(metric_value_1))

    # Plot the Graph.
    plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'red', label = metric_name_2)

    # Add title to the plot.
    plt.title(str(plot_name))
    # Add legend to the plot.
    plt.legend()
# Visualize the training and validation loss metrices.
plot_metric(convlstm_model_training_history, 'loss', 'val_loss', 'Total Loss vs Total V
alidation Loss')
# Visualize the training and validation accuracy metrices.
plot_metric(convlstm_model_training_history, 'accuracy', 'val_accuracy', 'Total Accur
acy vs Total Validation Accuracy')
def create_LRCN_model():
    '''
    This function will construct the required LRCN model.
    Returns:
        model: It is the required constructed LRCN model.
    '''

    # We will use a Sequential model for model construction.
    model = Sequential()

    # Define the Model Architecture.
    ####################################################################
##########################################################

    model.add(TimeDistributed(Conv2D(16, (3, 3), padding='same',activation = 'relu'),
                              input_shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE
_WIDTH, 3)))

    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Conv2D(32, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((2, 2))))
    model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation = 'relu')))
    model.add(TimeDistributed(MaxPooling2D((2, 2))))
    #model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Flatten()))

    model.add(LSTM(32))

    model.add(Dense(len(CLASSES_LIST), activation = 'softmax'))

    ##################################################################
##########################################################

    # Display the models summary.
    model.summary()

    # Return the constructed LRCN model.
    return model
# Construct the required LRCN model.
LRCN_model = create_LRCN_model()

# Display the success message.
```

```
print("Model Created Successfully!")

# Plot the structure of the contructed LRCN model.

plot_model(LRCN_model, to_file = 'LRCN_model_structure_plot.png', show_shapes
= True, show_layer_names = True)

# Create an Instance of Early Stopping Callback.

early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 15, mode =
 'min', restore_best_weights = True)


# Compile the model and specify loss function, optimizer and metrics to the model.

LRCN_model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam', metrics
= ["accuracy"])


# Start training the model.

LRCN_model_training_history = LRCN_model.fit(x = features_train, y = labels_train,
epochs = 70, batch_size = 4 ,

                                             shuffle = True, validation_split = 0.2, callbacks = [early_s
topping_callback])

# Evaluate the trained model.

model_evaluation_history = LRCN_model.evaluate(features_test, labels_test)

# Get the loss and accuracy from model_evaluation_history.

model_evaluation_loss, model_evaluation_accuracy = model_evaluation_history


# Define the string date format.

# Get the current Date and Time in a DateTime Object.

# Convert the DateTime object to string according to the style mentioned in date_tim
e_format string.

date_time_format = '%Y_%m_%d__%H_%M_%S'

current_date_time_dt = dt.datetime.now()

current_date_time_string = dt.datetime.strftime(current_date_time_dt, date_time_for
mat)


# Define a useful name for our model to make it easy for us while navigating through
multiple saved models.
```

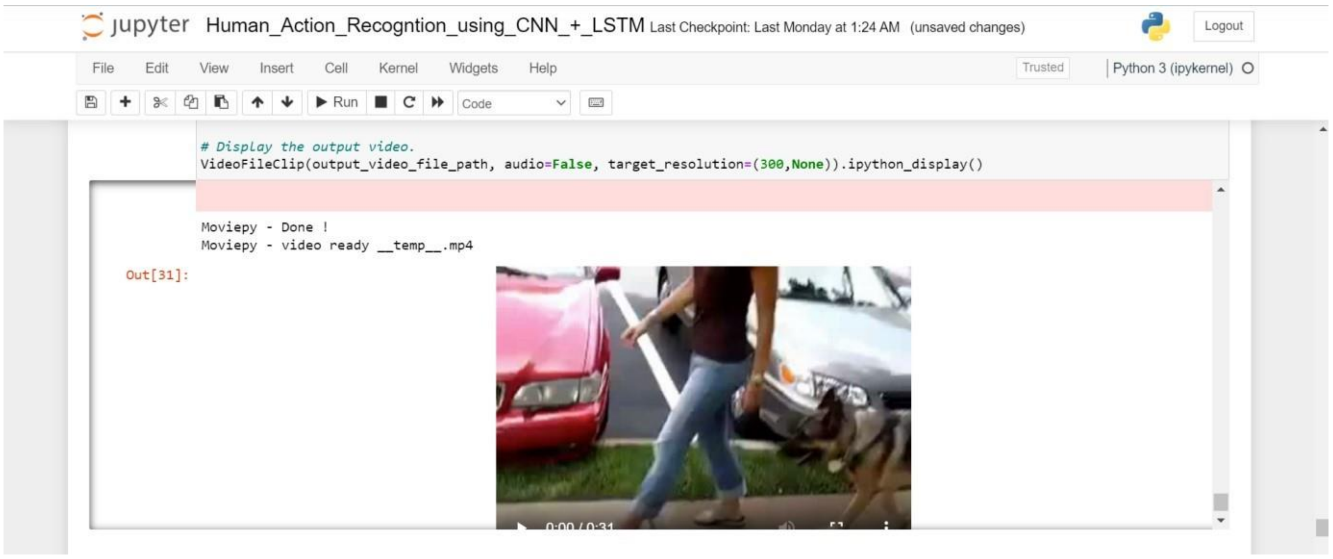
```
model_file_name = f'LRCN_model___Date_Time_{current_date_time_string}___Los
s_{model_evaluation_loss}___Accuracy_{model_evaluation_accuracy}.h5'


# Save the Model.
LRCN_model.save(model_file_name)
# Visualize the training and validation loss metrices.
plot_metric(LRCN_model_training_history, 'loss', 'val_loss', 'Total Loss vs Total Valid
ation Loss')
# Visualize the training and validation accuracy metrices.
plot_metric(LRCN_model_training_history, 'accuracy', 'val_accuracy', 'Total Accurac
y vs Total Validation Accuracy')
def download_youtube_videos(youtube_video_url, output_directory):
    '''
   This function downloads the youtube video whose URL is passed to it as an argu
ment.
   Args:
      youtube_video_url: URL of the video that is required to be downloaded.
      output_directory: The directory path to which the video needs to be stored after
 downloading.
   Returns:
      title: The title of the downloaded youtube video.
   '''


    # Create a video object which contains useful information about the video.
    video = pafy.new(youtube_video_url)


    # Retrieve the title of the video.
    title = video.title


    # Get the best available quality object for the video.
    video_best = video.getbest()
    # Construct the output file path.
    output_file_path = f'{output_directory}/{title}.mp4'

    # Download the youtube video at the best available quality and store it to the cont
ructed path.
    video_best.download(filepath = output_file_path, quiet = True)

    # Return the video title.
    return title
# Make the Output directory if it does not exist
test_videos_directory = 'test_videos'
os.makedirs(test_videos_directory, exist_ok = True)

# Download a YouTube Video.
video_title = download_youtube_videos('https://www.youtube.com/watch?v=8u0qjm
HIOcE', test_videos_directory)

# Get the YouTube Video's path we just downloaded.
input_video_file_path = f'{test_videos_directory}/{video_title}.mp4'
def predict_on_video(video_file_path, output_file_path, SEQUENCE_LENGTH):
   '''
   This function will perform action recognition on a video using the LRCN model.
   Args:
   video_file_path: The path of the video stored in the disk on which the action recog
nition is to be performed.
   output_file_path: The path where the ouput video with the predicted action being p
erformed overlayed will be stored.
   SEQUENCE_LENGTH: The fixed number of frames of a video that can be passe
d to the model as one sequence.
   '''
```

```
    # Initialize the VideoCapture object to read from the video file.
    video_reader = cv2.VideoCapture(video_file_path)

    # Get the width and height of the video.
    original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_video_height = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))

    # Initialize the VideoWriter Object to store the output video in the disk.
    video_writer = cv2.VideoWriter(output_file_path, cv2.VideoWriter_fourcc('M', 'P', '4
', 'V'),
                                   video_reader.get(cv2.CAP_PROP_FPS), (original_video_widt
h, original_video_height))

    # Declare a queue to store video frames.
    frames_queue = deque(maxlen = SEQUENCE_LENGTH)

    # Initialize a variable to store the predicted action being performed in the video.
    predicted_class_name = ''

    # Iterate until the video is accessed successfully.
    while video_reader.isOpened():

        # Read the frame.
        ok, frame = video_reader.read()

        # Check if frame is not read properly then break the loop.
        if not ok:
            break
        # Resize the Frame to fixed Dimensions.
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

        # Normalize the resized frame by dividing it with 255 so that each pixel value th
en lies between 0 and 1.
        normalized_frame = resized_frame / 255

        # Appending the pre-processed frame into the frames list.
        frames_queue.append(normalized_frame)

        # Check if the number of frames in the queue are equal to the fixed sequence le
ngth.
        if len(frames_queue) == SEQUENCE_LENGTH:

            # Pass the normalized frames to the model and get the predicted probabilitie
s.
            predicted_labels_probabilities = LRCN_model.predict(np.expand_dims(frame
s_queue, axis = 0))[0]

            # Get the index of class with highest probability.
            predicted_label = np.argmax(predicted_labels_probabilities)

            # Get the class name using the retrieved index.
            predicted_class_name = CLASSES_LIST[predicted_label]

        # Write predicted class name on top of the frame.
        cv2.putText(frame, predicted_class_name, (10, 30), cv2.FONT_HERSHEY_SI
MPLEX, 1, (0, 255, 0), 2)

        # Write The frame into the disk using the VideoWriter Object.
        video_writer.write(frame)
```

```
    # Release the VideoCapture and VideoWriter objects.
    video_reader.release()
    video_writer.release()
# Construct the output video path.
output_video_file_path = f'{test_videos_directory}/{video_title}-Output-
SeqLen{SEQUENCE_LENGTH}.mp4'


# Perform Action Recognition on the Test Video.
predict_on_video(input_video_file_path, output_video_file_path, SEQUENCE_LENG
TH)


# Display the output video.
VideoFileClip(output_video_file_path, audio=False, target_resolution=(300,None)).ip
ython_display()
def predict_single_action(video_file_path, SEQUENCE_LENGTH):
    '''
    This function will perform single action recognition prediction on a video using the
LRCN model.
    Args:
    video_file_path: The path of the video stored in the disk on which the action recog
nition is to be performed.
    SEQUENCE_LENGTH: The fixed number of frames of a video that can be passe
d to the model as one sequence.
    '''


    # Initialize the VideoCapture object to read from the video file.
    video_reader = cv2.VideoCapture(video_file_path)


    # Get the width and height of the video.
    original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_video_height = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))
    # Declare a list to store video frames we will extract.
    frames_list = []

    # Initialize a variable to store the predicted action being performed in the video.
    predicted_class_name = ''

    # Get the number of frames in the video.
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    # Calculate the interval after which frames will be added to the list.
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH),1)

    # Iterating the number of times equal to the fixed length of sequence.
    for frame_counter in range(SEQUENCE_LENGTH):

        # Set the current frame position of the video.
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frame
s_window)

        # Read a frame.
        success, frame = video_reader.read()

        # Check if frame is not read properly then break the loop.
        if not success:
            break

        # Resize the Frame to fixed Dimensions.
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
        # Normalize the resized frame by dividing it with 255 so that each pixel value th
en lies between 0 and 1.

        normalized_frame = resized_frame / 255


        # Appending the pre-processed frame into the frames list

        frames_list.append(normalized_frame)


    # Passing the pre-
processed frames to the model and get the predicted probabilities.

    predicted_labels_probabilities = LRCN_model.predict(np.expand_dims(frames_list,
 axis = 0))[0]


    # Get the index of class with highest probability.

    predicted_label = np.argmax(predicted_labels_probabilities)


    # Get the class name using the retrieved index.

    predicted_class_name = CLASSES_LIST[predicted_label]


    # Display the predicted action along with the prediction confidence.

    print(f'Action Predicted: {predicted_class_name}\nConfidence: {predicted_labels_p
robabilities[predicted_label]}')


    # Release the VideoCapture object.

    video_reader.release()

    # Download the youtube video.

video_title = download_youtube_videos('https://youtu.be/fc3w827kwyA', test_videos_
directory)


# Construct tihe nput youtube video path

input_video_file_path = f'{test_videos_directory}/{video_title}.mp4'
```

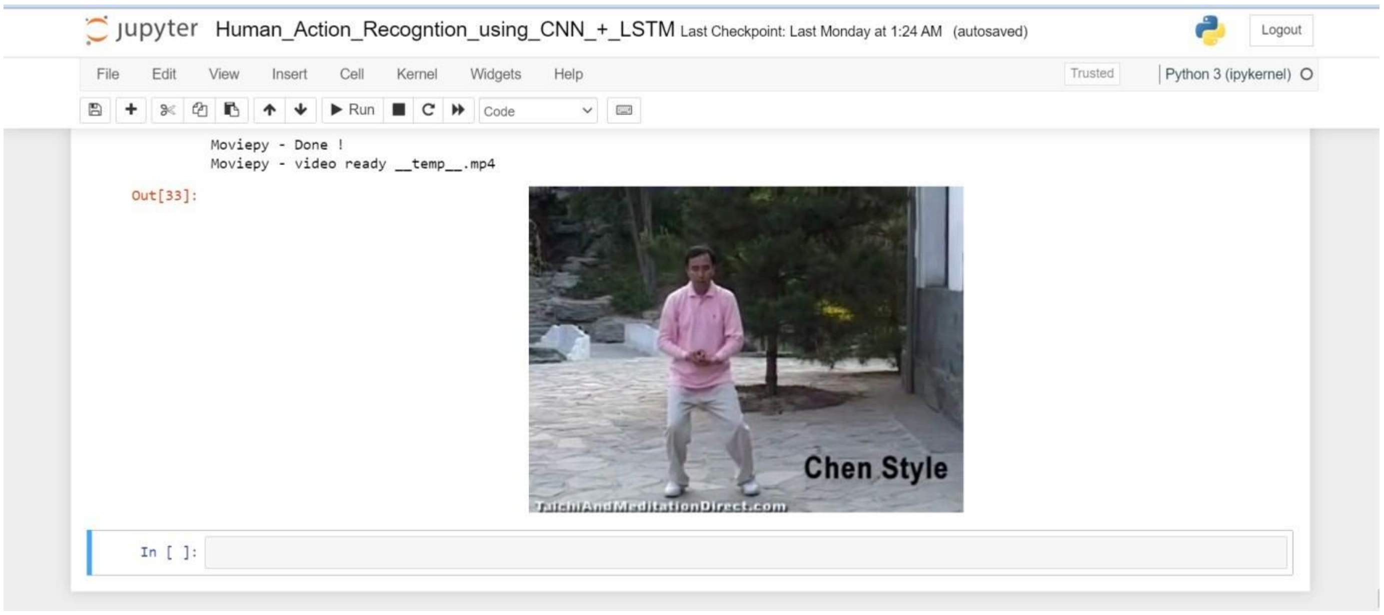
```
# Perform Single Prediction on the Test Video.

predict_single_action(input_video_file_path, SEQUENCE_LENGTH)

# Display the input video.

VideoFileClip(input_video_file_path, audio=False, target_resolution=(300,None)).ipyt
hon_display()
```

SCREEN SHOTS:


jupyter
Human_Action_Recogntion_using_CNN_+_LSTM Last Checkpoint: Last Monday at 1:24 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Out[31]:
0:00 / 0:31

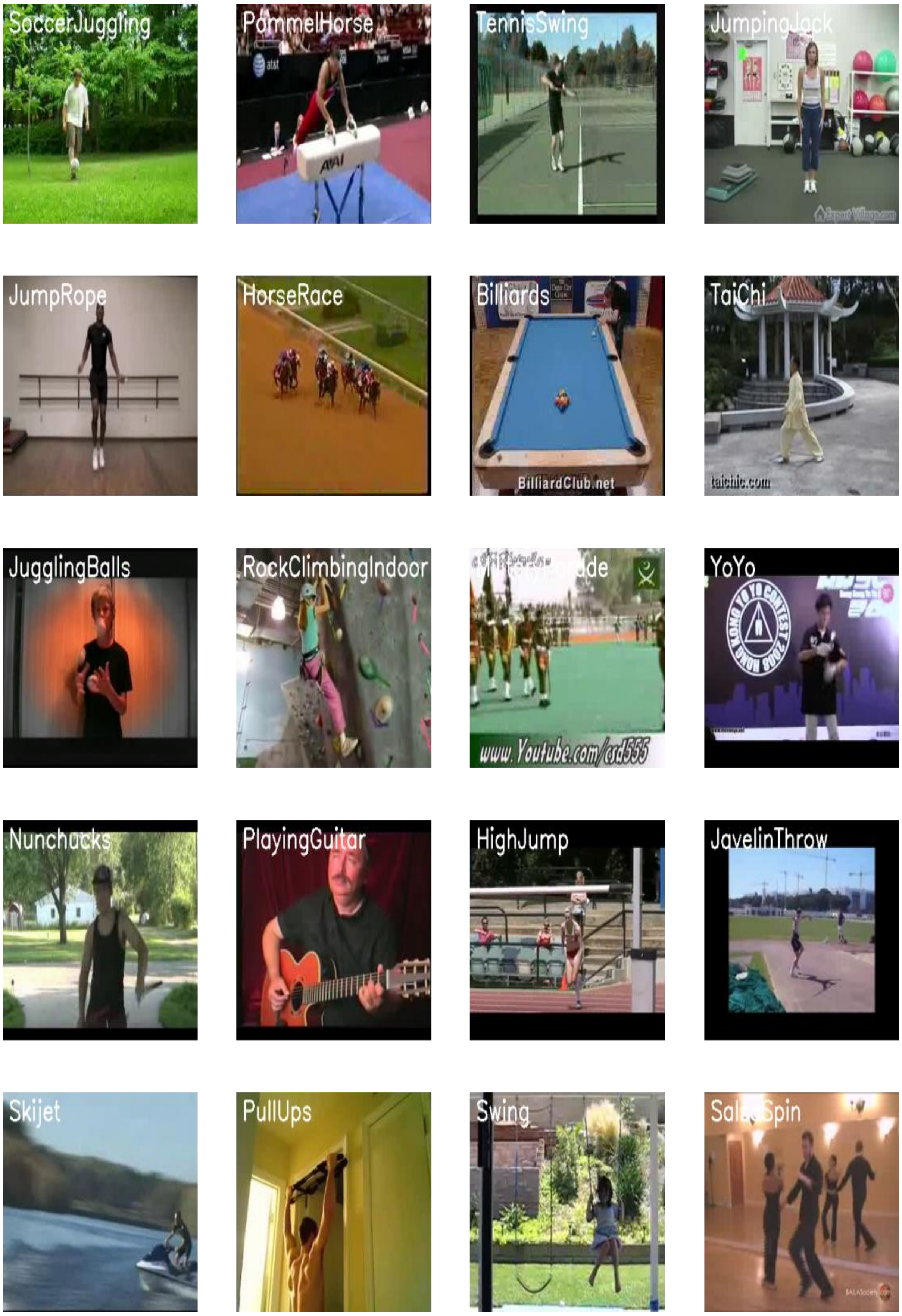
```
# Display the output video.
VideoFileClip(output_video_file_path, audio=False, target_resolution=(300,None)).ipython_display()
```

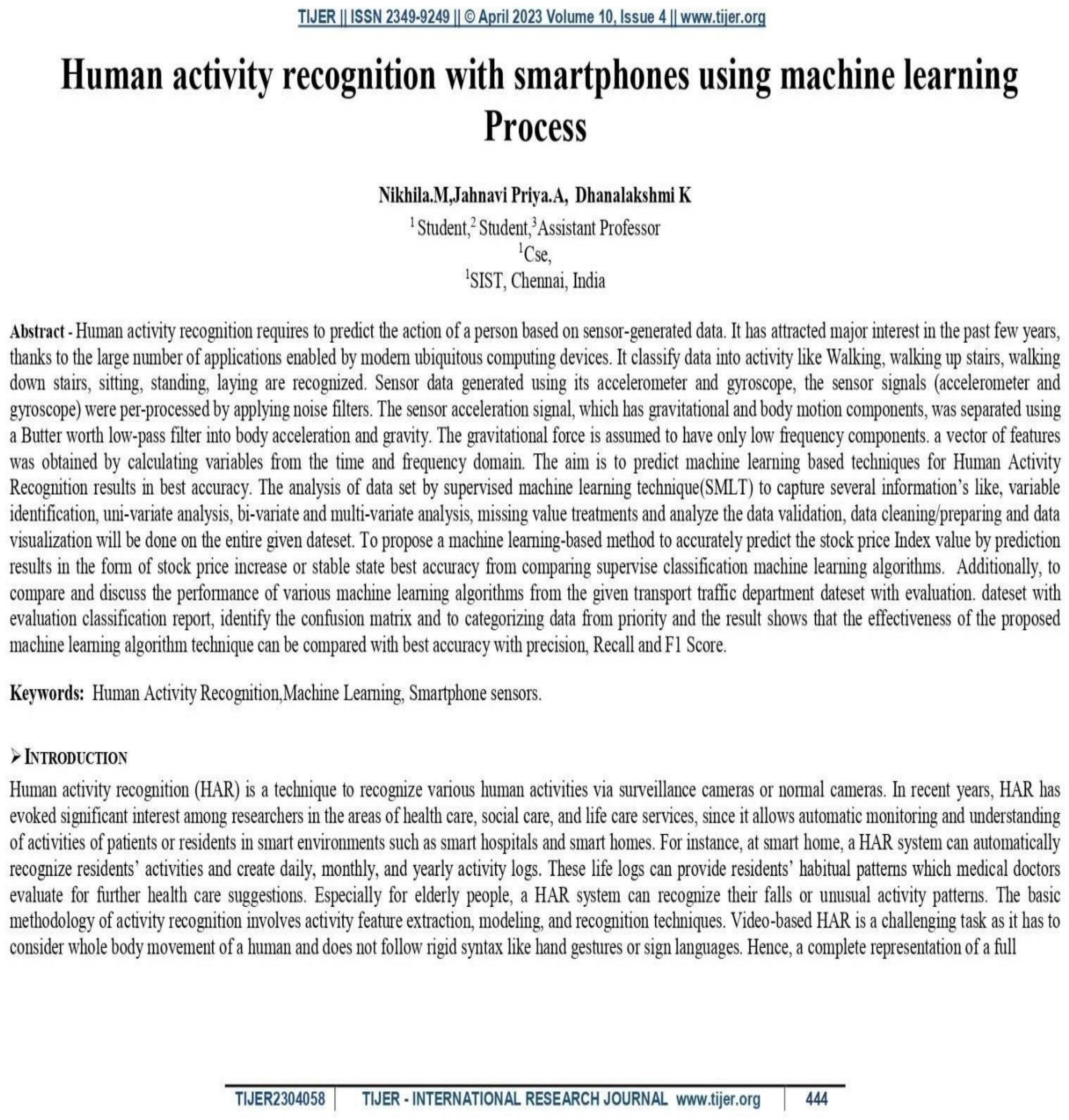
```
Moviepy - Done !
Moviepy - video ready __temp__.mp4
```


jupyter
Human_Action_Recogntion_using_CNN_+_LSTM Last Checkpoint: Last Monday at 1:24 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Out[33]:
Chen Style
TaichiAndMeditationDirect.com
In [ ]:

```
Moviepy - Done !
Moviepy - video ready __temp__.mp4
```

Running


SoccerJuggling
PommelHorse
TennisSwing
JumpingJack
JumpRope
HorseRace
Billiards
BilliardClub.net
TaiChi
taichic.com
JugglingBalls
RockClimbingIndoor
www.Youtube.com/csd555
YoYo
Nunchucks
PlayingGuitar
HighJump
JavelinThrow
Skijet
PullUps
Swing
SalsaSpin



TIJER || ISSN 2349-9249 || © April 2023 Volume 10, Issue 4 || www.tijer.org

Human activity recognition with smartphones using machine learning Process

Nikhila.M,Jahnavi Priya.A, Dhanalakshmi K

[1] Student,[2] Student,[3]Assistant Professor

[1]Cse,

[1]SIST, Chennai, India

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 classify data into activity like Walking, walking up stairs, walking down stairs, sitting, standing, laying are recognized. Sensor data generated using its accelerometer and gyroscope, the sensor signals (accelerometer and gyroscope) were per-processed by applying noise filters. The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butter worth 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 data set 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 dateset. To propose a machine learning-based method to accurately predict the stock price Index value by prediction results in the form of stock price increase or stable state best accuracy from comparing supervise classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given transport traffic department dateset with evaluation. dateset 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.

Keywords: Human Activity Recognition,Machine Learning, Smartphone sensors.

➢ Introduction

Human activity recognition (HAR) is a technique to recognize various human activities via surveillance cameras or normal cameras. In recent years, HAR has evoked significant interest among researchers in the areas of health care, social care, and life care services, since it allows automatic monitoring and understanding of activities of patients or residents in smart environments such as smart hospitals and smart homes. For instance, at smart home, a HAR system can automatically recognize residents' activities and create daily, monthly, and yearly activity logs. These life logs can provide residents' habitual patterns which medical doctors evaluate for further health care suggestions. Especially for elderly people, a HAR system can recognize their falls or unusual activity patterns. The basic methodology of activity recognition involves activity feature extraction, modeling, and recognition techniques. Video-based HAR is a challenging task as it has to consider whole body movement of a human and does not follow rigid syntax like hand gestures or sign languages. Hence, a complete representation of a full human body is essential to characterize human movements properly in this regard. Though many researchers have been exploring video-based HAR systems due to their practical applications, accurate recognition of human activities still remains as a major challenge.

Human activity recognition, or HAR, is a challenging time series classification task. It involves predicting the movement of a person based on sensor data and traditionally involves deep domain expertise and methods from signal processing to correctly engineer features from the raw data in order to fit a machine learning model. Recently, deep learning methods such as convolutional neural networks and recurrent neural networks have shown capable and even achieve state-of-the-art results by automatically learning features from the raw sensor data. In this post, you will discover the problem of human activity recognition and the deep learning methods that are achieving state-of-the-art performance on this problem.

Activity classification is essentially a time series problem. Time-series classification is a type of supervised machine learning. It's used to predict future values from past data using statistical techniques, and it can be used for forecasting and offloading sensor data. As of today, neural networks have proven to be the most effective in performing activity recognition. In particular, two approaches, including Convolutional Neural Network Models and Recurrent Neural Network Models, are the most widely used for this task.

➢ PROBLEM STATEMENT

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.

➢ 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

- 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.

➢ SPECIFIC PREREQUISITES

● HARDWARE PREREQUISITES

• RAM	: 4GB or above 4GB
• Processor of Frequency	: 1.5GHz or above
• Processor	: Intel Pentium 4 or higher

- **SOFTWARE PREREQUISITES**

• Operating System	: Windows 8 and above
• Languages	: Python
• Tools used	: Visual Studio Code, Jupyter Notebook

- **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 sea born.

- **NON-FUNCTIONAL REQUIREMENTS**

Process of functional steps,

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

Prediction the result

➢ **ARCHITECTURE**

The system architecture for the proposed system has been shown in the above figure. The video is taken from the camera. The video is sent to the trained model. The model detects the actions of the human. The action is returned as the output.

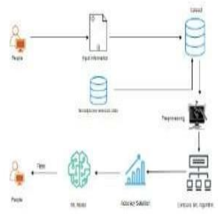
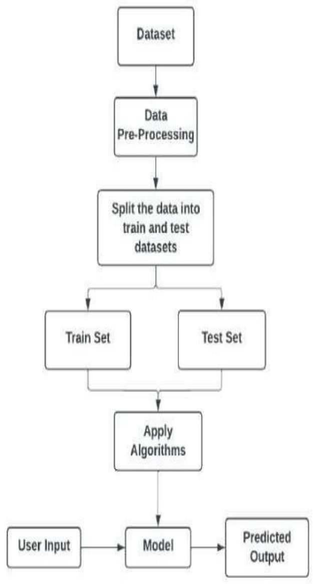


Fig.1 SYSTEM ARCHITECTURE



➢ CONCLUSION

Here we suggested a convolutional neural network (CNN) and long short-term memory (LSTM) method for human activity recognition. The above method hopes to enhance activity detection performance by making use of the former method's strong feature removal and the latter method's predicting future time sequence & categorization. Comparing the effectiveness of this CNN-LSTM network to that of other dl strategies that use raw sensor readings as input revealed that the latter is superior on both fronts. We tested our model on two different datasets, one private & one open (UCI HAR), and we found that it performed well in both areas. On the iSPL datasets in particular, it achieved better results than the competing models by a margin of over 1% in precision and almost 2% in Soft - max loss. The duration of time it required to execute the various systems and factor is determined was another parameter that was not assessed in this article but was evident in the trials. When contrasted to our method, the other algorithms took much longer to complete.

➢ REFERENCES

[1] Usharani, J.; Saktivel, U. Human Activity Recognition using Android Smartphone. In Proceedings of the International Conference on Innovations in Computing & Networking ICICN16, Bengaluru, Karnataka, 12–13 May 2016.

[2] Anguita, D.; Ghio, A.; Oneto, L.; Parra-Llanas, X.; Reyes-Ortiz, J. Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic. J. Univers. Comput. Sci. 2013, 19, 1295–1314.

[3] Uddin, Z.; Soylu, A. Human activity recognition using wearable sensors, discriminant analysis, and long short-term memory-based neural structured learning. *Sci. Rep.* 2021, *11*, 16455.

[4] Vakili, M.; Rezaei, M. Incremental Learning Techniques for Online Human Activity Recognition. *arXiv* **2021**, arXiv:2109.09435.

[5] Muangprathub, J.; Sriwichian, A.; Wanichsombat, A.; Kajornkasirat, S.; Nillaor, P.; Boonjing, V. A Novel Elderly Tracking System Using Machine Learning to Classify Signals from Mobile and Wearable Sensors. *Int. J. Environ. Res. Public Health* **2021**, *18*, 12652.

[6] Zhou, B.; Yang, J.; Li, Q. Smartphone-Based Activity Recognition for Indoor Localization Using a Convolutional Neural Network. *Sensors* 2019, *19*, 621.

[7] Murad, A.; Pyun, J.-Y. Deep Recurrent Neural Networks for Human Activity Recognition. *Sensors* **2017**, *17*, 2556.