MACHINE LEARNING

(Human Activity Prediction Using Smartphone Data)

Summer Internship Report Submitted in partial

fulfillment of the requirement for undergraduate degree

Of

Bachelor of Technology

In

Computer Science and Engineering

By

Gopi Bollineni

221710307029

Under the Guidance of

_____

Assistant Professor

Department Of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University) Hyderabad-502329

DECLARATION

I submitted this industrial training work entitled **"Human Activity Prediction Using Smartphone Data**" to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology**" in "**Computer Science and Engineering**". I declare that it was carried out independently by me under the guidance of **Mr. XXXX**, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: Hyderabad — Gopi Bollineni

Date: 12-07-2020 — 221710307029

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated:12-7-2020

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"Human Activity Prediction"** is being submitted by Gopi Bollineni (221710307029) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science And Engineering** at GITAM (Deemed to Be University), Hyderabad during the academic year 2019-20.

It is faithful record work carried out by him at the **Computer Science And Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Dr. S.Phani Kumar

Assistant Professor

Professor and HOD

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank, respected **Dr. N. Siva Prasad**, Pro Vice-Chancellor, GITAM Hyderabad, and Dr. CH. Sanjay, Principal, GITAM Hyderabad.

I would like to thank respected **Prof. S. Phani Kumar**, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present the internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties _____ who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Gopi Bollineni

221710307029

ABSTRACT

Human activity recognition has wide applications in medical research and human survey systems. In this project, we design a robust activity recognition system based on a smartphone. The system uses a 3-dimensional smartphone accelerometer and a gyroscope as the sensors to collect time-series signals, from which 31 features are generated in both time and frequency domain.

Activities are classified into 6 different categories, i.e, walking, walking downstairs, walking upstairs, sitting, standing, lying. These activities are predicted using 5 different passive learning methods, i.e., logistic regression algorithm, k-nearest neighbors algorithm, Gaussian naive bayes algorithm, random forest classifier.

Dimensionality reduction is performed through both feature extraction and subset selection. Experiment results show that the classification rate of passive learning reaches 95.8% and it is robust to common positions and poses of the cell phone.

Table of contents:

List of Figures used:

1. MACHINE LEARNING

1.1. Introduction:

Over the past two decades Machine Learning has become one of the mainstays of information technology and with that, a rather central, albeit usually hidden, part of our life. With the ever increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress.

Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might not be completely known at design time. Machine learning methods can be used for on-the-job improvement of existing machine designs. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down. Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign. New knowledge about tasks is constantly being discovered by humans. Vocabulary changes. There is a constant stream of new events in the world. Continuing redesign of AI systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

1.2. Importance of Machine Learning:

Machine learning is a branch of artificial intelligence that aims at enabling machines to perform their jobs skillfully by using intelligent software. The statistical learning methods constitute the backbone of intelligent software that is used to develop machine intelligence. Because machine learning algorithms require data to learn, the discipline must have connection with the discipline of databases. Similarly, there are familiar terms such as Knowledge Discovery from Data (KDD), data mining, and pattern recognition. One wonders how to view the big picture in which such connection is illustrated.

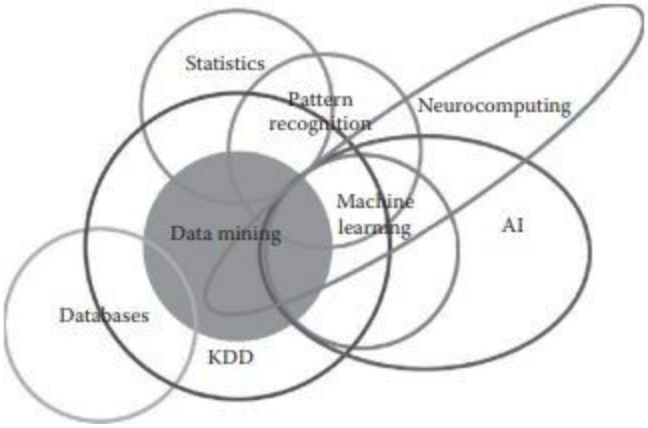


Fig 1.1 usage of Machine learning in different fields

There are some tasks that humans perform effortlessly or with some efforts, but we are unable to explain how we perform them. For example, we can recognize the speech of our friends without much difficulty. If we are asked how we recognize the voices, the answer is very difficult for us to explain. Because of the lack of understanding of such phenomenon (speech recognition in this case), we cannot craft algorithms for such scenarios. Machine learning algorithms are helpful in bridging this gap of understanding.

The idea is very simple. We are not targeting to understand the underlying processes that help us learn. We write computer programs that will make machines learn and enable them to

perform tasks, such as prediction. The goal of learning is to construct a model that takes the input and produces the desired result. Sometimes, we can understand the model, whereas, at other times, it can also be like a black box for us, the working of which cannot be intuitively explained. The model can be considered as an approximation of the process we want machines to mimic. In such a situation, it is possible that we obtain errors for some input, but most of the time, the model provides correct answers. Hence, another measure of performance (besides performance of metrics of speed and memory usage) of a machine learning algorithm will be the accuracy of results.

1.3. Uses of Machine Learning:

Artificial Intelligence (AI) is everywhere. Possibility is that you are using it in one way or the other and you don't even know about it. One of the popular applications of AI is Machine Learning (ML), in which computers, software, and devices perform via cognition (very similar to the human brain). Herein, we share a few examples of machine learning that we use everyday and perhaps have no idea that they are driven by ML. These are some of the uses and applications of ML.

i. Virtual Personal Assistants:

Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. All you need to do is activate them and ask "What is my schedule for today?", "What are the flights from Germany to London", or similar questions. For answering, your personal assistant looks out for the information, recalls your related queries, or sends a command to other resources (like phone

apps) to collect info. You can even instruct assistants for certain tasks like “Set an alarm for 6 AM next morning”, “Remind me to visit the Visa Office day after tomorrow”.

Machine learning is an important part of these personal assistants as they collect and refine the information on the basis of your previous involvement with them. Later, this set of data utilized to render results that are tailored to your preferences.

Virtual Assistants are integrated to a variety of platforms. For example:

- Smart Speakers: Amazon Echo and Google Home
- Smartphones: Samsung Bixby on Samsung S10
- Mobile Apps: Google Allo

ii. Predictions while commuting:

Traffic Predictions*:* We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. While this helps in preventing the traffic and does congestion analysis, the underlying problem is that there are less number of cars that are equipped with GPS. Machine learning in such scenarios helps to estimate the regions where congestion can be found on the basis of daily experiences.

Online Transportation Networks*:* When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours? The answer is machine learning. Jeff Schneider, the engineering lead at Uber ATC reveals in an interview that they use ML to define price surge hours by predicting the rider demand. In the entire cycle of the services, ML is playing a major role.

iii. Social Media Services:

From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits. Here are a few examples that you must be noticing, using, and loving in your social media accounts, without realizing that these wonderful features are nothing but the applications of ML.

- **People You May Know:** Machine learning works on a simple concept: understanding with experiences. Facebook continuously notices the friends that you connect with, the profiles that you visit very often, your interests, workplace, or a group that you share with someone etc. On the basis of continuous learning, a list of Facebook users are suggested that you can become friends with.
- **Face Recognition:** You upload a picture of you with a friend and Facebook instantly recognizes that friend. Facebook checks the poses and projections in the picture, notice the unique features, and then match them with the people in your friend list. The entire process at the backend is complicated and takes care of the precision factor but seems to be a simple application of ML at the front end.
- **Similar Pins:** Machine learning is the core element of Computer Vision, which is a technique to extract useful information from images and videos. Pinterest uses computer vision to identify the objects (or pins) in the images and recommends similar pins accordingly.

iv. Search Engine Result Refining:

Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results. If you open the top results and stay on the web page for long, the search engine assumes that the the results it displayed were in accordance to the query. Similarly, if you reach the second or third page of the search results but do not open any of the results, the search engine estimates that the results served did not match requirement. This way, the algorithms working at the backend improve the search results.

v. Product Recommendations:

You shopped for a product online few days back and then you keep receiving emails for shopping suggestions. If not this, then you might have noticed that the shopping website or the app recommends you some items that somehow matches with your taste. On the basis of your behaviour with the website/app, past purchases, items liked or added to cart, brand preferences etc., the product recommendations are made.

vi. Online Fraud Detection:

Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. For example: Paypal is using ML for protection against money laundering. The company uses a set of tools that helps them to compare millions of transactions taking place and distinguish between legitimate or illegitimate transactions taking place between the buyers and sellers.

Fig 1.2 Uses of Machine learning

1.4. Types of Machine Learning:

There are 3 types of Machine learning which are widely used in today's world these are:

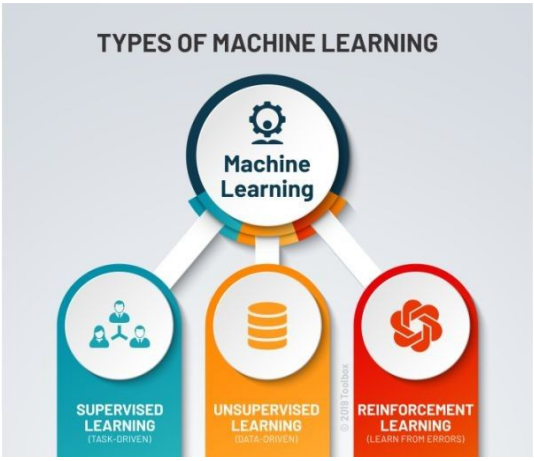


Fig 1.3 types of ML

1.4.1 Supervised Learning:

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem. The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

1.4.2 Unsupervised Learning:

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set

problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

1.4.3 Reinforcement Learning:

It directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or 'reinforced', and non-favorable outputs are discouraged or 'punished'.Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result. In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

2. DEEP LEARNING

2.1. Deep Learning Importance:

Deep learning algorithms run data through several “layers” of neural network algorithms, each of which passes a simplified representation of the data to the next layer.Most machine learning algorithms work well on datasets that have up to a few hundred features, or columns.

Basically deep learning is itself a subset of machine learning but in this case the machine learns in a way in which humans are supposed to learn. The structure of deep learning models is highly similar to a human brain with a large number of neurons and nodes like neurons in the human brain thus resulting in an artificial neural network. In applying traditional machine learning algorithms we have to manually select input features from complex data set and then train them which becomes a very tedious job for ML scientist but in neural networks we don’t have to manually select useful input features, there are various layers of neural networks for handling complexity of the data set and algorithm as well. In my recent project on human activity recognition , when we applied traditional machine learning algorithm like K-NN then we have to separately detect human and its activity also had to select impactful input parameters manually which became a very tedious task as data set was way too complex but the complexity dramatically reduced on applying artificial neural network, such is the power of deep learning. Yes it’s correct that deep learning algorithms take lots of time for training sometimes even weeks as well but its execution on new data is so fast that it's not even comparable with traditional ML algorithms. Deep learning has enabled Industrial Experts to overcome challenges which were

impossible a decade ago like Speech and Image recognition and Natural Language Processing. Majority of the Industries are currently depending on it , be it Journalism, Entertainment, Online Retail Store, Automobile, Banking and Finance, Healthcare, Manufacturing or even Digital Sector. Video recommendations, Mail Services, Self Driving cars, Intelligent Chat bots, Voice Assistants are just trending achievements of Deep Learning.

Furthermore, Deep learning can most profoundly be considered as the future of Artificial Intelligence due to constant rapid increase in amount of data as well as the gradual development in the hardware field as well, resulting in better computational power.

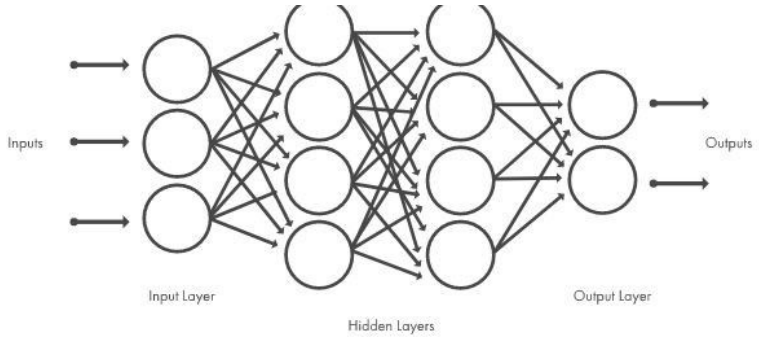


Fig 2.1 Deep Neural network

2.2. Uses of Deep Learning:

i. Translations:

Although automatic machine translation isn't new, deep learning is helping enhance automatic translation of text by using stacked networks of neural networks and allowing translations from images.

ii. Adding color to black-and-white images and videos:

It used to be a very time-consuming process where humans had to add color to black-and-white images and videos by hand can now be automatically done with deep-learning models.

iii. Language recognition:

Deep learning machines are beginning to differentiate dialects of a language. A machine decides that someone is speaking English and then engages an AI that is learning to tell the differences between dialects. Once the dialect is determined, another AI will step in that specializes in that particular dialect. All of this happens without involvement from a human.

iv. Autonomous vehicles:

There's not just one AI model at work as an autonomous vehicle drives down the street. Some deep-learning models specialize in streets signs while others are trained to recognize pedestrians. As a car navigates down the road, it can be informed by up to millions of individual AI models that allow the car to act.

v. Computer vision:

Deep learning has delivered super-human accuracy for image classification, object detection, image restoration and image segmentation—even handwritten digits can be recognized. Deep

learning using enormous neural networks is teaching machines to automate the tasks performed by human visual systems.

vi.Text generation:

The machines learn the punctuation, grammar and style of a piece of text and can use the model it developed to automatically create entirely new text with the proper spelling, grammar and style of the example text. Everything from Shakespeare to Wikipedia entries have been created.

vii. Deep-learning robots:

Deep-learning applications for robots are plentiful and powerful from an impressive deep-learning system that can teach a robot just by observing the actions of a human completing a task to a housekeeping robot that's provided with input from several other AIs in order to take action. Just like how a human brain processes input from past experiences, current input from senses and any additional data that is provided, deep-learning models will help robots execute tasks based on the input of many different AI opinions.

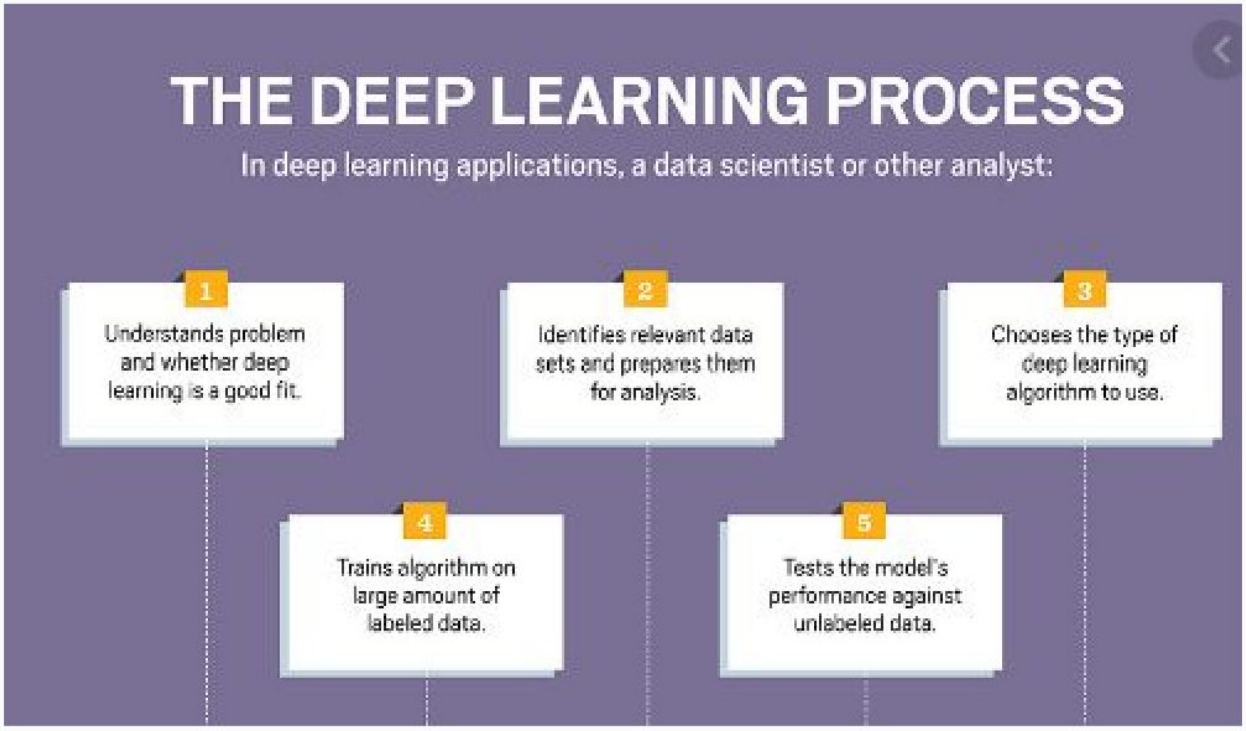


Fig 2.2 The deep learning process

2.3. Relation between Data Mining, Machine Learning and Deep Learning:

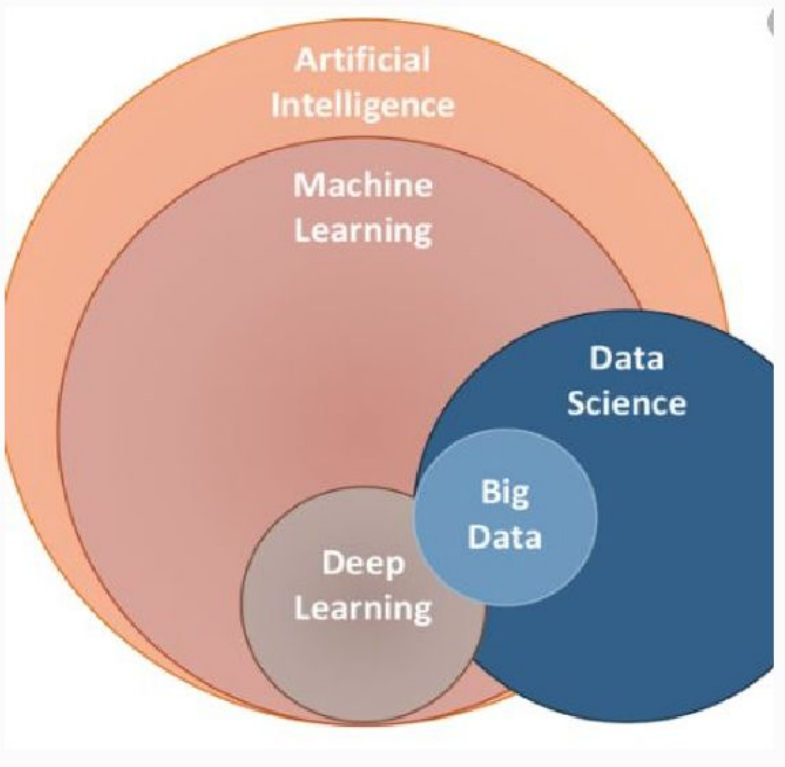


Fig 2.3 Relation between DM,ML,DL

The deep learning, data mining and machine learning share a foundation in data science, and there certainly is overlap between the two. Data mining can use machine learning algorithms to improve the accuracy and depth of analysis, and vice-versa; machine learning can use mined data as its foundation, refining the dataset to achieve better results.

You could also argue that data mining and machine learning are similar in that they both seek to address the question of how we can learn from data. However, the way in which they achieve this end, and their applications, form the basis of some significant differences.

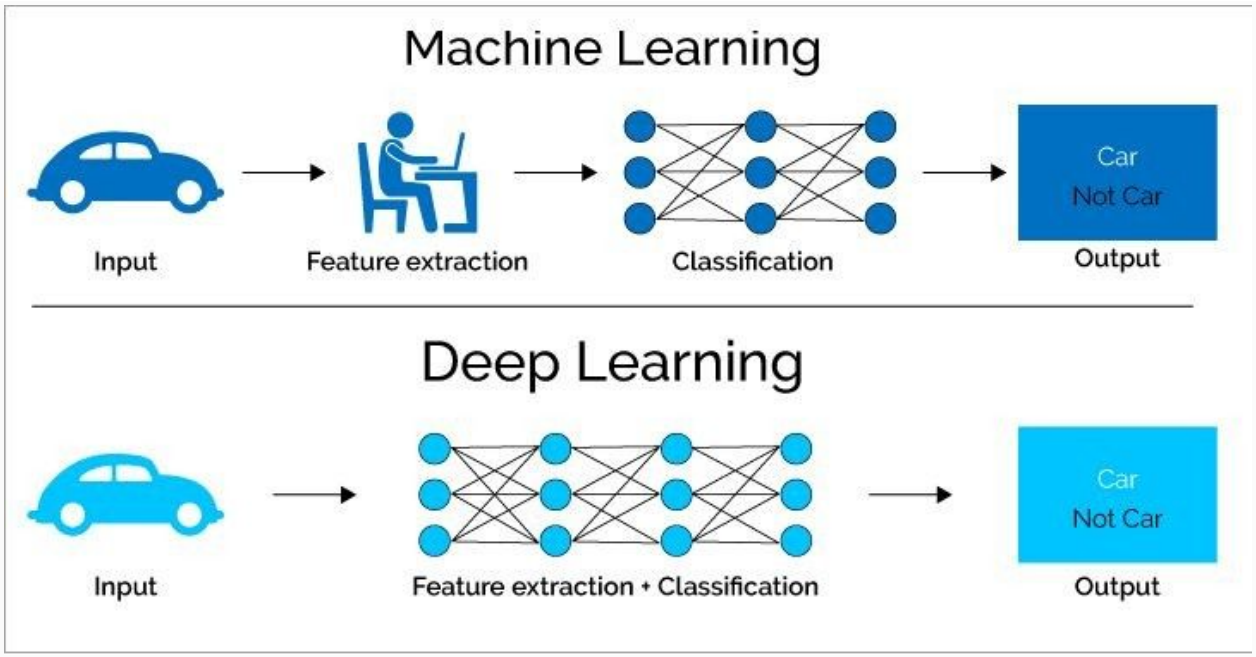


Fig 2.4 process in machine learning and deep learning

Machine Learning comprises of the ability of the machine to learn from trained data set and predict the outcome automatically. It is a subset of artificial intelligence.

Deep Learning is a subset of machine learning. It works in the same way on the machine just like how the human brain processes information. Like a brain can

identify the patterns by comparing it with previously memorized patterns, deep learning also uses this concept.

Deep learning can automatically find out the attributes from raw data while machine learning selects these features manually which further needs processing. It also employs artificial neural networks with many hidden layers, big data, and high computer resources.

Data Mining is a process of discovering hidden patterns and rules from the existing data. It uses relatively simple rules such as association, correlation rules for the decision-making process, etc. Deep Learning is used for complex problem processing such as voice recognition etc. It uses Artificial Neural Networks with many hidden layers for processing. At times data mining also uses deep learning algorithms for processing the data.

3. PYTHON

3.1. Introduction:

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is dynamically typed and garbage-collected.It supports multiple programming paradigms,including structured , object-oriented,and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

3.2. Setup of Python:

- Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

3.2.1 Installation(using python IDLE):

- To start, go to python.org/downloads and then click on the button to download the latest version of Python.
- We can download python IDLE in windows,mac and linux operating systems also.

Figure 3.2.1 : Python download

- Run the .exe file that you just downloaded and start the installation of Python by clicking on Install Now.
- We can give environmental variable i.e path after completion of downloading

Fig 3.2.1.1 python installation

- When python is installed, a program called IDLE is also installed along with it.It provides a graphical user interface to work with python.

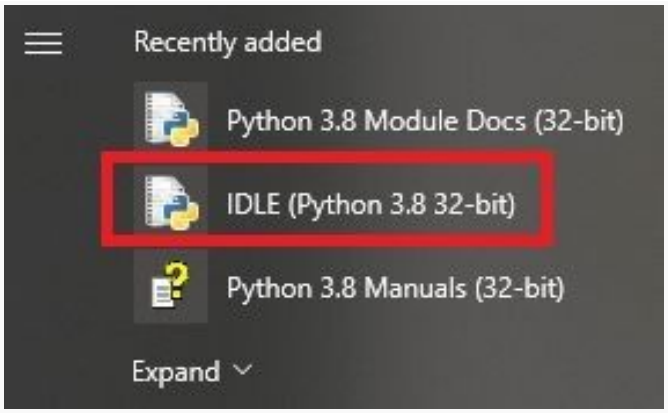


Fig 3.2.1.2 IDLE

3.2.2. Python Installation using Anaconda:

- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages. Anaconda for Windows installation:
    1. Go to the following link: Anaconda.com/downloads

2. Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
3. Select path(i.e. add anaconda to path & register anaconda as default python 3.4)
4. Click finish

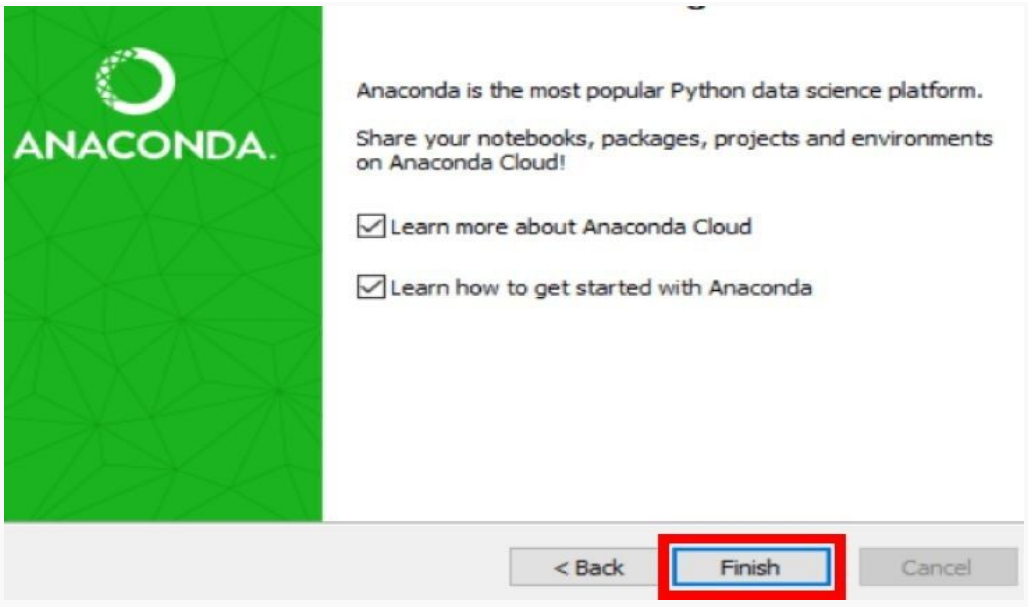


Fig 3.2.2.1 After installation

5. Open jupyter notebook

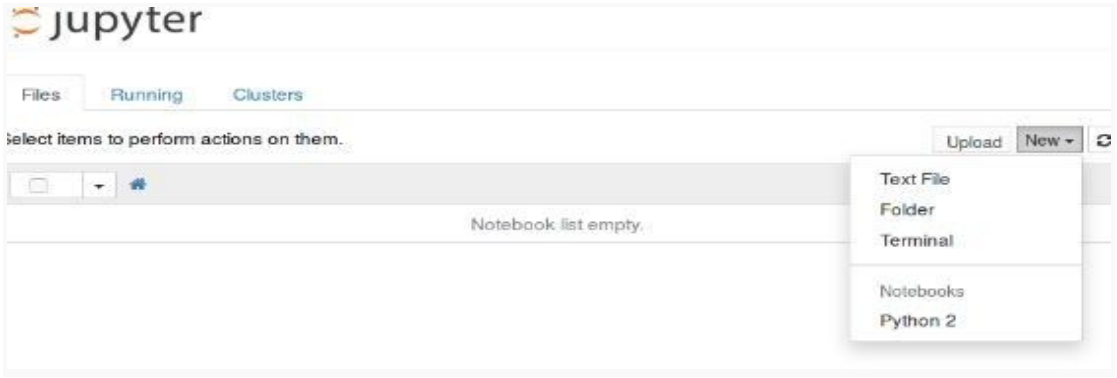


Fig 3.2.2.2 jupyter notebook

3.3. Features:

1. **Readable:** Python is a very readable language.
2. **Easy to Learn:** Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn.
3. **Cross platform:** Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.
4. **Open Source:** Python is an open source programming language.
5. **Large standard library:** Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.
6. **Free:** Python is free to download and use. This means you can download it for free and use it in your application. Python is an example of a FLOSS (Free/Libre Open Source Software), which means you can freely distribute copies of this software, read its source code and modify it.
7. **Supports exception handling:** If you are new, you may wonder what is an exception? An exception is an event that can occur during program exception and can disrupt the normal flow of program. Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.
8. **Advanced features:** Supports generators and list comprehensions. We will cover these features later.
9. **Automatic memory management:** Python supports automatic memory management which means the memory is cleared and freed automatically. You do not have to bother clearing the memory.

3.4. Variable Types:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Python has five standard data types –

- Numbers
- Strings
- Lists
- Tuples
- Dictionary

3.4.1 Python Numbers:

Number data types store numeric values. They are immutable data types, means that changing the value of a number data type results in a newly allocated object.

Python supports four different numerical types −

- **int (signed integers)** − They are often called just integers or ints, are positive or negative whole numbers with no decimal point.
- **long (long integers)** − Also called longs, they are integers of unlimited size, written like integers and followed by an uppercase or lowercase L.
- **float (floating point real values)** − Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10(2.5e2=2.5x10^2=250).

3.4.2. Python Strings:

In Python, Strings can be created by simply enclosing characters in quotes. Python does not support character types. These are treated as length-one strings, and are also considered as substrings. Substrings are immutable and can't be changed once created.Strings are the ordered blocks of text that are enclosed in single or double quotations. Thus, whatever is written in quotes, is considered as string. Though it can be written in single or double quotations, double quotation marks allow the user to extend strings over multiple lines without backslashes, which is usually the signal of continuation of an expression, e.g., 'abc', "ABC".

3.4.3. Python Lists:

- List is a collection data type in python. It is ordered and allows duplicate entries as well. Lists in python need not be homogeneous, which means it can contain different data types like integers, strings and other collection data types. It is mutable in nature and allows indexing to access the members in a list.
- To declare a list, we use the square brackets.
- List is like any other array that we declare in other programming languages. Lists in python are often used to implement stacks and queues. The lists are mutable in nature. Therefore, the values can be changed even after a list is declared.

3.4.4 Python Tuples:

- A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also.

3.4.5 Python Dictionary:

- It is a collection data type just like a list or a set, but there are certain features that make python dictionary unique. A dictionary in python is not ordered and is changeable as well. We can make changes in a dictionary unlike sets or strings which are immutable in nature. Dictionary contains key-value pairs like a map that we have in other programming languages. A dictionary has indexes. Since the value of the keys we declare in a dictionary are always unique, we can use them as indexes to access the elements in a dictionary.

3.5. Functions:

3.5.1. Defining a Function:

- Function blocks begin with the keyword def followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or docstring.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

3.5.2 Calling a Function:

- Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.
- Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

3.6 OOPs Concepts:

3.6.1 Class:

- Python is an object oriented programming language. Unlike procedure oriented programming, where the main emphasis is on functions, object oriented programming stresses on objects.
- An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.
- We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.
- As many houses can be made from a house's blueprint, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is calledinstantiation.
- Like function definitions begin with the def keyword in Python, class definitions begin with a class keyword.
- The first string inside the class is called docstring and has a brief description about the class.

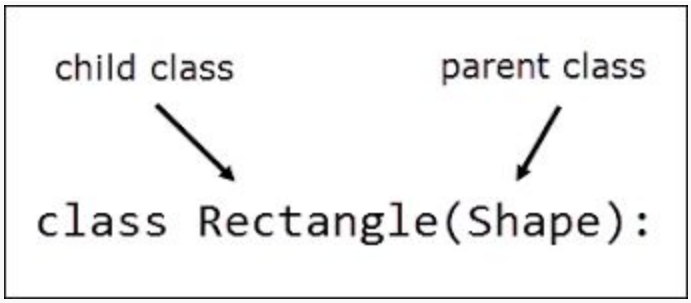


Fig 3.6.1 Class defining

- As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

```
class Person:
    "This is a person class"
    age = 10

    def greet(self):
        print('Hello')


# Output: 10
print(Person.age)

# Output: <function Person.greet>
print(Person.greet)

# Output: 'This is my second class'
print(Person.__doc__)
```

Fig 3.6.1.1 Example of class

4. HUMAN ACTIVITY PREDICTION USING SMARTPHONE DATA

4.1. Project Requirements:

4.1.1 Packages Used:

- **Numpy:** In Python we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster that traditional

Python lists.The array object in NumPy is called ndata, it provides a lot of supporting functions that make working with ndarray very easy.Arrays are very frequently used in data science, where speed and resources are very important.

- **Pandas:**Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.
- **Seaborn:** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Matplotlib: Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

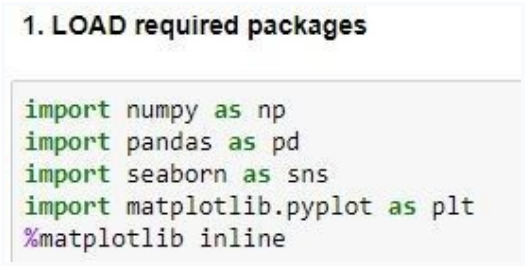


Fig 4.1.1 packages

4.1.2. Versions of Packages:

The versions of the packages are found by following command

```
#versions of packages
import numpy
import matplotlib
print('numpy:',numpy.__version__)
print('pandas:',pd.__version__)
print('seaborn:',sns.__version__)
print('matplotlib:',matplotlib.__version__)
```

```
numpy: 1.18.5
pandas: 1.0.5
seaborn: 0.10.1
matplotlib: 3.2.2
```

Fig 4.1.2 versions

4.1.3 Algorithms Used:

Here , 3 algorithms are used they are:

- Logistic regression
- K-Nearest Neighbors Classifier
- Gaussian Naive Bayes Classifier
- Random Forest

4.2. Problem Statement:

Statement: John wants to find out the amount of calories he is burning everyday, for this, he needs to know the various activities he's doing all day using the means of his smartphone. In order to do this he collects the Accelerometer and Gyroscope readings built into his smartphone

and applies several machine learning algorithms to find out the best algorithm to predict and classify his daily activities.

Human activity recognition is the problem of classifying sequences of accelerometer data recorded by specialized harnesses or smart phones into known well-defined movements.

It is a challenging problem given the large number of observations produced each second, the temporal nature of the observations, and the lack of a clear way to relate accelerometer data to known movements.

In this problem we try to classify human activities into WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING and LAYING using machine learning algorithms, and compare how accurately each model is capable of classifying the same.

4.3. Dataset Description:

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years.

Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, 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 signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor

acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity.

The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

For each record it is provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables.
- Its activity label.
- An identifier of the subject who carried out the experiment.

Subsequently, the body linear acceleration and angular velocity were derived in time to obtain Jerk signals (tBodyAccJerk-XYZ and tBodyGyroJerk-XYZ). Also the magnitude of these three-dimensional signals were calculated using the Euclidean norm (tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag, tBodyGyroJerkMag).

Finally a Fast Fourier Transform (FFT) was applied to some of these signals producing fBodyAcc-XYZ, fBodyAccJerk-XYZ, fBodyGyro-XYZ, fBodyAccJerkMag, fBodyGyroMag, fBodyGyroJerkMag (Note the 'f' to indicate frequency domain signals).

These signals were used to estimate variables of the feature vector for each pattern:'-XYZ' is used to denote 3-axial signals in the X, Y and Z directions.

- tBodyAcc-XYZ
- tGravityAcc-XYZ
- tBodyAccJerk-XYZ
- tBodyGyro-XYZ

- tBodyGyroJerk-XYZ
- tBodyAccMag
- tGravityAccMag
- tBodyAccJerkMag
- tBodyGyroMag
- tBodyGyroJerkMag
- fBodyAcc-XYZ
- fBodyAccJerk-XYZ
- fBodyGyro-XYZ
- fBodyAccMag
- fBodyAccJerkMag
- fBodyGyroMag
- fBodyGyroJerkMag

The set of variables that were estimated from these signals are:

- mean(): Mean value
- std(): Standard deviation
- mad(): Median absolute deviation
- max(): Largest value in array
- min(): Smallest value in array
- sma(): Signal magnitude area
- energy(): Energy measure. Sum of the squares divided by the number of values.
- iqr(): Interquartile range
- entropy(): Signal entropy
- arCoeff(): Autoregression coefficients with Burg order equal to 4
- correlation(): correlation coefficient between two signals
- maxInds(): index of the frequency component with largest magnitude
- meanFreq(): Weighted average of the frequency components to obtain a mean frequency

- skewness(): skewness of the frequency domain signal
- kurtosis(): kurtosis of the frequency domain signal
- bandsEnergy(): Energy of a frequency interval within the 64 bins of the FFT of each window.
- angle(): Angle between two vectors.

Additional vectors obtained by averaging the signals in a signal window sample. These are used on the angle() variable:

- gravityMean
- tBodyAccMean
- tBodyAccJerkMean
- tBodyGyroMean
- tBodyGyroJerkMean

4.4. Objective of the Case Study:

Human Activity Recognition integrates the emerging area of sensor networks with novel data mining and machine learning techniques to model a wide range of human activities.Mobile devices (e.g. smartphones) provide sufficient sensor data and calculation power to enable physical activity recognition to provide an estimation of the energy consumption during everyday life. Sensor-based activity recognition researchers believe that by empowering ubiquitous computers and sensors to monitor the behavior of agents (under consent), these computers will be better suited to act on our behalf.

5. DATA PREPROCESSING/FEATURE ENGINEERING AND EDA

5.1. Statistical Analysis:

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a DataFrame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be accessed using the dataframe. Any missing value or NaN value has to be cleaned.

```
In [2]:
#reading the data:
train=pd.read_csv("train.csv")
test=pd.read_csv("test.csv")
print(train.shape)
print(test.shape)
print(type(train),type(test))

(7352, 563)
(2947, 563)
<class 'pandas.core.frame.DataFrame'> <class 'pandas.core.frame.DataFrame'>
```

Fig 5.1 loading data set

Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding nan values.Analyzes both numeric and object series, as well as DataFrame column sets of mixed data types. The output will vary depending on what is provided.

For numeric data, the result's index will include count, mean, std, min, max as well as lower, 50 and upper percentiles. By default the lower percentile is 25 and the upper percentile is 75. The 50 percentile is the same as the median.

For object data (e.g. strings or timestamps), the result's index will include count, unique, top,and freq. The top is the most common value. The freq is the most common value's frequency. Timestamps also include the first and last items.

If multiple object values have the highest count, then the count and top results will be arbitrarily chosen from among those with the highest count.

For mixed data types provided via a DataFrame, the default is to return only an analysis of numeric columns. If the data frame consists only of object and categorical data without any numeric columns, the default is to return an analysis of both the object and categorical columns. If include='all' is provided as an option, the result will include a union of attributes of each type.

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X
count	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000
mean	0.274488	-0.017695	-0.109141	-0.605438	-0.510938	-0.604754	-0.630512	-0.526907	-0.606150	-0.468604
std	0.070261	0.040811	0.056635	0.448734	0.502645	0.418687	0.424073	0.485942	0.414122	0.544547
min	-1.000000	-1.000000	-1.000000	-1.000000	-0.999873	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
25%	0.262975	-0.024863	-0.120993	-0.992754	-0.978129	-0.980233	-0.993591	-0.978162	-0.980251	-0.936219
50%	0.277193	-0.017219	-0.108676	-0.946196	-0.851897	-0.859365	-0.950709	-0.857328	-0.857143	-0.881637
75%	0.288461	-0.010783	-0.097794	-0.242813	-0.034231	-0.262415	-0.292680	-0.066701	-0.265671	-0.017129
max	1.000000	1.000000	1.000000	1.000000	0.916238	1.000000	1.000000	0.967664	1.000000	1.000000

8 rows × 562 columns

Fig 5.1.1 Statistical data

Note: As shown above there are 562 columns in our data, therefore, it is hard to display all the columns in this document. However, *DataFrame.describe(include='all')* statement can be used to display all the statistical values. Jupyter notebook by default skips a few columns in the output to increase the viewability of the data. Nevertheless, pandas provides a solution for this by adding the statement *pandas.options.display.max_columns='600'* all the columns can be viewed. Similarly, the *max_rows* method of the same module can be used to set the number of rows that can be viewed.

Observations:

- Both the train and the test data were scaled priorly by the researchers who took the privilege to collect the data. So, there wouldn't be any need for us to scale the data.
- The activity was carried out between 30 subjects as a various time-series data.

5.2. Data Type Conversions:

When doing data analysis, it is important to make sure you are using the correct data types; otherwise you may get unexpected results or errors. In the case of pandas, it will correctly infer data types in many cases and you can move on with your analysis without any further thought on the topic.Despite how well pandas works, at some point in your data analysis processes, you will likely need to explicitly convert data from one type to another. This article will discuss the basic pandas data types (aka dtypes), how they map to python and numpy data types and the options for converting from one pandas type to another.

```
:  tBodyAcc-mean()-X                 float64
   tBodyAcc-mean()-Y                 float64
   tBodyAcc-mean()-Z                 float64
   tBodyAcc-std()-X                  float64
   tBodyAcc-std()-Y                  float64
   tBodyAcc-std()-Z                  float64
   tBodyAcc-mad()-X                  float64
   tBodyAcc-mad()-Y                  float64
   tBodyAcc-mad()-Z                  float64
   tBodyAcc-max()-X                  float64
   tBodyAcc-max()-Y                  float64
   tBodyAcc-max()-Z                  float64
   tBodyAcc-min()-X                  float64
   tBodyAcc-min()-Y                  float64
   tBodyAcc-min()-Z                  float64
   tBodyAcc-sma()                    float64
   tBodyAcc-energy()-X               float64
   tBodyAcc-energy()-Y               float64
   tBodyAcc-energy()-Z               float64
   tBodyAcc-iqr()-X                  float64
   tBodyAcc-iqr()-Y                  float64
   tBodyAcc-iqr()-Z                  float64
   tBodyAcc-entropy()-X              float64
   tBodyAcc-entropy()-Y              float64
   tBodyAcc-entropy()-Z              float64
   tBodyAcc-arCoeff()-X,1            float64
```

Fig 5.2 data types

5.3. Detection of Outliers:

Perhaps the most common or familiar type of outlier is the observations that are far from the rest of the observations or the center of mass of observations.This is easy to understand when we have one or two variables and we can visualize the data as a histogram or scatter plot, although it becomes very challenging when we have many input variables defining a high-dimensional input feature space.In this case, simple statistical methods for identifying outliers can break down, such as methods that use standard deviations or the interquartile range.It can be important to identify and remove outliers from data when training machine learning algorithms for predictive modeling.Outliers can skew statistical measures and data distributions, providing a misleading representation of the underlying data and relationships. Removing outliers from training data prior to modeling can result in a better fit of the data and, in turn, more skillful predictions.Thankfully, there are a variety of automatic model-based methods for identifying outliers in input data. Importantly, each method approaches the definition of an outlier in slightly different ways, providing alternate approaches to preparing a training dataset that can be evaluated and compared, just like any other data preparation step in a modeling pipeline.

However, there are no outliers in our data set.

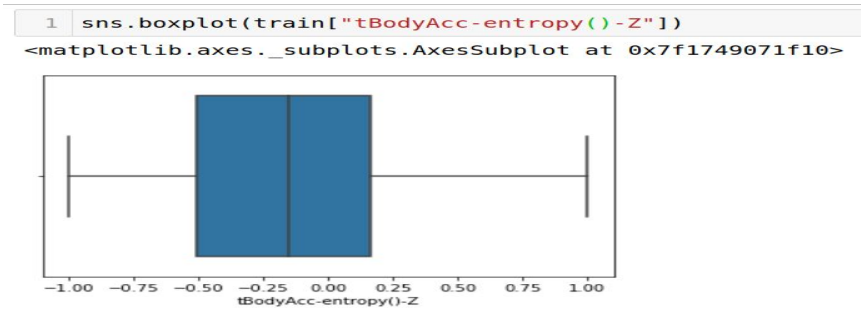


Fig 5.3 detection of outlier using boxplot

5.4. Handling Missing Values:

There are a number of schemes that have been developed to indicate the presence of missing data in a table or DataFrame. Generally, they revolve around one of two strategies: using a mask that globally indicates missing values, or choosing a sentinel value that indicates a missing entry.In the masking approach, the mask might be an entirely separate Boolean array, or it may involve appropriation of one bit in the data representation to locally indicate the null status of a value.In the sentinel approach, the sentinel value could be some data-specific convention, such as indicating a missing integer value with -9999 or some rare bit pattern, or it could be a more global convention, such as indicating a missing floating-point value with NaN (Not a Number), a special value which is part of the IEEE floating-point specification.

However, we have no missing values in our DataSet.

```
#checking for NULL values:
pd.options.display.max_rows=600
train.isnull().sum()
#there are no NULL values in the data
```

```
tBodyAcc-mean()-X                     0
tBodyAcc-mean()-Y                     0
tBodyAcc-mean()-Z                     0
tBodyAcc-std()-X                      0
tBodyAcc-std()-Y                      0
tBodyAcc-std()-Z                      0
tBodyAcc-mad()-X                      0
tBodyAcc-mad()-Y                      0
tBodyAcc-mad()-Z                      0
tBodyAcc-max()-X                      0
tBodyAcc-max()-Y                      0
tBodyAcc-max()-Z                      0
tBodyAcc-min()-X                      0
tBodyAcc-min()-Y                      0
tBodyAcc-min()-Z                      0
tBodyAcc-sma()                        0
tBodyAcc-energy()-X                   0
tBodyAcc-energy()-Y                   0
tBodyAcc-energy()-Z                   0
```

Fig 5.4 Checking for missing values

5.5. Encoding Categorical Data:

Categorical Variables are of two types: Nominal and Ordinal

- Nominal: The categories do not have any numeric ordering in between them.They don't have any ordered relationship between each of them. Examples: Male or Female, any colour.
- Ordinal: The categories have a numerical ordering in between them. Example: Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer satisfaction survey, high low medium Categorical data can be handled by using dummy variables, which are also called as indicator variables.

However, we need not use encoding in for this given dataset as the only Categorical values in the dataset are of the column ‘Activity’ and these values are already encoded in the dataset priorly.

```
1 train.select_dtypes(include=['float','int']).columns
```

```
Index(['tBodyAcc-mean()-X', 'tBodyAcc-mean()-Y', 'tBodyAcc-mean()-Z',
       'tBodyAcc-std()-X', 'tBodyAcc-std()-Y', 'tBodyAcc-std()-Z',
       'tBodyAcc-mad()-X', 'tBodyAcc-mad()-Y', 'tBodyAcc-mad()-Z',
       'tBodyAcc-max()-X', 'tBodyAcc-max()-Y', 'tBodyAcc-max()-Z',
       'tBodyAcc-min()-X', 'tBodyAcc-min()-Y', 'tBodyAcc-min()-Z',
       'tBodyAcc-sma()', 'tBodyAcc-energy()-X', 'tBodyAcc-energy()-Y',
       'tBodyAcc-energy()-Z', 'tBodyAcc-iqr()-X', 'tBodyAcc-iqr()-Y',
       'tBodyAcc-iqr()-Z', 'tBodyAcc-entropy()-X', 'tBodyAcc-entropy()-Y',
       'tBodyAcc-entropy()-Z', 'tBodyAcc-arCoeff()-X,1',
       'tBodyAcc-arCoeff()-X,2', 'tBodyAcc-arCoeff()-X,3',
       'tBodyAcc-arCoeff()-X,4', 'tBodyAcc-arCoeff()-Y,1',
       'tBodyAcc-arCoeff()-Y,2', 'tBodyAcc-arCoeff()-Y,3',
```

Fig 5.5 numerical in the dataset

5.6. Generating Plots:

5.6.1. Check for imbalances in the Target Values:

```
plt.subplots(figsize=(12,5))
sns.countplot(train.Activity)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f462e4c8d50>
```

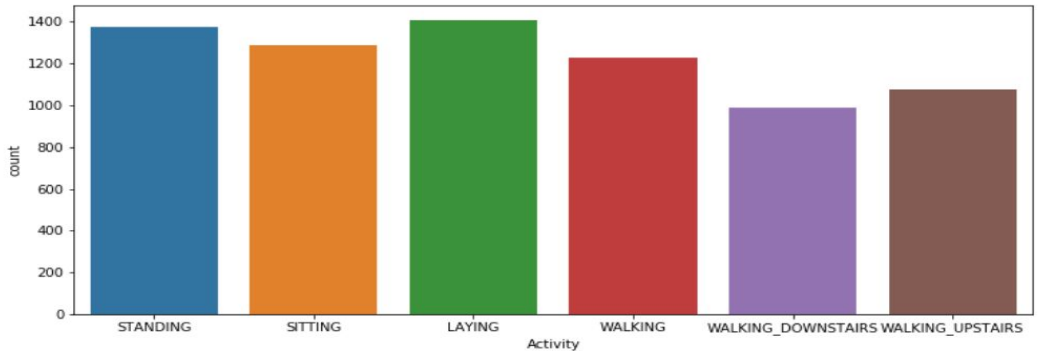


Fig 5.6.1 Imbalances in the dataset(bar)

```
plt.subplots(figsize=(7,7))
plt.pie(train.Activity.value_counts(),labels=train.Activity.unique(),autopct='%1.2f%%')
plt.show()
#we can see that all the features are equally distributed
```

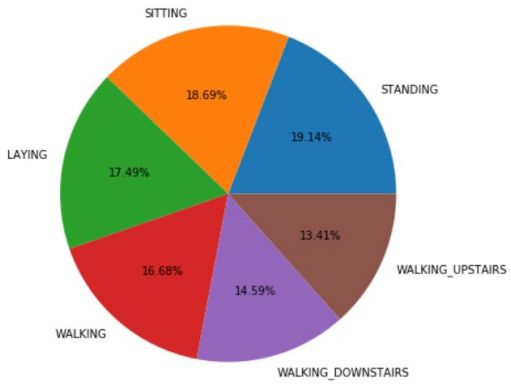


Fig 5.6.2 Imbalances in the dataset(pie)

We can see that there are not many imbalances between the values of the Target column.

5.6.2. Visualise the data between theTarget and Features:

Correlation:

```
1 plt.subplots(figsize=(10,10))
2 sns.heatmap(x_train.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efd03ca3290>
```

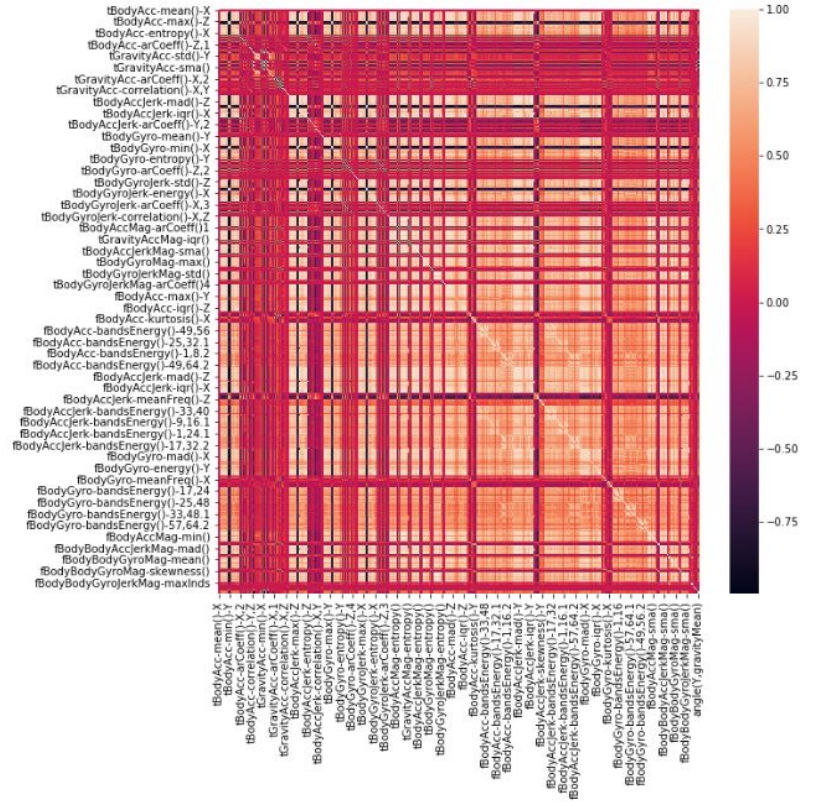


Fig 5.6.3 Heatmap for Correlation

6. FEATURE SELECTION

6.1. Select Relevant Features for analysis:

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

- **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.
- **Improves Accuracy**: Less misleading data means modeling accuracy improves.
- **Reduces Training Time**: fewer data points reduce algorithm complexity and algorithms train faster.

Feature Selection Methods:

I will share 3 Feature selection techniques that are easy to use and also gives good results.

1. Univariate Selection
2. Feature Importance
3. Correlation Matrix with Heatmap

6.2. Drop the Irrelevant Features:

We consider the data from all the columns as input to our models as it can be observed from the FEATURE Engineering process that , there are numerous columns but on the ground level, all the columns are just an engineered product of the ACCELEROMETER and GYROSCOPE inputs recorded in a continuous 'time-series'.

There's more than one reason to include all the Features as input to this data, one of them being the 'JERK' values, actions like the ones we're classifying in here, they all generate JERKS

while the actions are being performed. Therefore, this value is important and cannot be ignored from being passed as an input in spite of being highly correlated to the other values.

So, here I'm trying to consider every column of the data to predict my classes.

6.3. Train-Test-Split:

One of the first decisions to make when starting a modeling project is how to utilize the existing data. One common technique is to split the data into two groups typically referred to as the training and testing sets. The training set is used to develop models and feature sets; they are the substrate for estimating parameters, comparing models, and all of the other activities required to reach a final model. The test set is used only at the conclusion of these activities for estimating a final, unbiased assessment of the model's performance. It is critical that the test set not be used prior to this point. Looking at the test sets results would bias the outcomes since the testing data will have become part of the model development process.

There are a number of ways to split the data into training and testing sets. The most common approach is to use some version of random sampling. Completely random sampling is a straightforward strategy to implement and usually protects the process from being biased towards any characteristic of the data. However this approach can be problematic when the response is not evenly distributed across the outcome. A less risky splitting strategy would be to use a stratified random sample based on the outcome. For classification models, this is accomplished by selecting samples at random within each class. This approach ensures that the frequency distribution of the outcome is approximately equal within the training and test sets. When the outcome is numeric, artificial strata can be constructed based on the quartiles of the data. For example, in the Ames housing price data, the quartiles of the outcome distribution would break the data into four artificial groups containing roughly 230 houses. The training/test split would then be conducted within these four groups and the four different training set portions are pooled together (and the same for the test set).

Splitting the data into INPUT and OUTPUT DataFrames:

x_train--> train INPUT; y_train --> train OUTPUT
x_test--> test Input; y_test--> test INPUT

the subject column in all the data is not useful for our model so we're gonna 'drop' that column

Here we're going to split the 'train' DataFrame into x_train,y_train,x_test,y_test and feed them to the models and then, we will be passing the 'test' DataFrame as raw data to obtain predictions from the model

```
train.drop(['subject'],axis=1,inplace=True)
test.drop(['subject'],axis=1,inplace=True)
```

Train-Test-Split

```
x=train.drop(['Activity'],axis=1)
y=train.Activity
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

Fig 6.3.1 Train-Test-split

6.4. Feature Scaling:

It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm. Real world dataset contains features that highly vary in magnitudes, units, and range. Normalisation should be performed when the scale of a feature is irrelevant or misleading and not should Normalise when the scale is meaningful.

The algorithms which use Euclidean Distance measure are sensitive to Magnitudes.Here feature scaling helps to weigh all the features equally.

Formally, If a feature in the dataset is big in scale compared to others then in algorithms where Euclidean distance is measured this big scaled feature becomes dominating and needs to be normalized.

$$z = \frac{x - \mu}{\sigma}$$

Fig 6.4.1 formula for scaling

However, for our data there is no need for Feature Scaling as the values are already scaled in our given dataset.

- **K-Means** uses the Euclidean distance measure here feature scaling matters.
- **K-Nearest-Neighbours** also require feature scaling.
- **Principal Component Analysis (PCA)**:Tries to get the feature with maximum variance, here too feature scaling is required.
- **Gradient Descent**: Calculation speed increase as Theta calculation becomes faster after feature scaling.

7. MODEL BUILDING AND EVALUATION

7.1. Brief about the Algorithms used:

i. Logistic Regression:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of the target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same −

- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other.
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
lr_train_predict=lr.predict(x_train)
lr_test_predict=lr.predict(x_test)
```

Fig 7.1.1 Logistic Regression Training and predicting

ii. K-Nearest Neighbors Classifier:

The sklearn.neighbors module implements the k-nearest neighbors algorithm.Neighbors-based classification is a type of *instance-based learning* or *non-generalizing learning*: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

```
from sklearn.neighbors import KNeighborsClassifier
knc_init=KNeighborsClassifier(n_neighbors=20,metric='euclidean')
knc_init.fit(x_train,y_train)
knc_init_train_predict=knc_init.predict(x_train)
knc_init_test_predict=knc_init.predict(x_test)
```

Fig 7.1.2 KNN Training and predicting

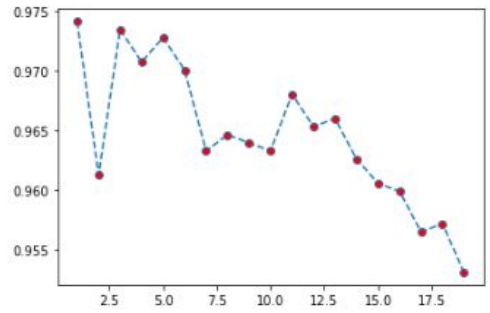
In this algorithm the optimum value for the n_neighbors attribute can be found out by plotting an elbow curve as shown below:

```
score=[]
for k in range(1,20):
    knn_model=KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(x_train,y_train)
    pred_test=knn_model.predict(x_test)
    score.append(accuracy_score(y_test,pred_test))
print(score)
```

```
[0.9741672331747111, 0.9612508497620667, 0.973487423521414, 0.9707681849082257, 0.972807613868117, 0.9700883752549
286, 0.9632902787219578, 0.964649898028552, 0.9639700883752549, 0.9632902787219578, 0.9680489462950373, 0.96532970
76818491, 0.9660095173351462, 0.9626104690686608, 0.9605710401087696, 0.9598912304554724, 0.9564921821889871, 0.95
71719918422842, 0.9530931339225017]
```

```
#plotting the score
plt.plot(range(1,20),score,marker='o',markerfacecolor='r',linestyle='--')
```

```
[<matplotlib.lines.Line2D at 0x7efcf8dff8d0>]
```



Here it can be observed that the highest accuracy is achieved through the K-Nearest Neighbors Algorithm at the value where **n_neighbors=1**

Fig 7.1.3 Elbow Curve

From the above plot it can be inferred that the optimum value of n_neighbors attribute is 1(one). Now, we can build the KNN model on the obtained n_neighbors value.

```
#So, let's build a model with the same parameter value
knc=KNeighborsClassifier(n_neighbors=1,metric='euclidean')
knc.fit(x_train,y_train)
knc_train_predict=knc.predict(x_train)
knc_test_predict=knc.predict(x_test)
```

Fig 7.1.4 KNN final model

iii. Gaussian Naive Bayes Classifier:

The "sklearn.naive_bayes" module implements Naive Bayes algorithms. These are supervised learning methods based on applying Bayes' theorem with strong (naive) feature independence assumptions.GaussianNB implements the Gaussian Naive Bayes algorithm for classification.

The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Fig 7.1.5 Naive Bayes Formula

The parameters σ_y and μ_y are estimated using maximum likelihood.

```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb.fit(x_train,y_train)
nb_train_predict=nb.predict(x_train)
nb_test_predict=nb.predict(x_test)
```

Fig 7.1.6 GaussianNB Training and Prediction

iv. Random Forest:

Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression tasks). In this post we'll learn how the random forest algorithm works, how it differs from other algorithms and how to use it.

Random forest is a supervised learning algorithm. The "forest" it builds is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random forest is a supervised learning algorithm. The "forest" it builds is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems.

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(criterion='entropy')
rfc.fit(x_train,y_train)
rfc_train_predict=rfc.predict(x_train)
rfc_test_predict=rfc.predict(x_test)
```

Fig 7.1.7 Random Forest Training and Prediction

7.2. Train the Models:

Splitting the data : after the preprocessing is done then the data is split into train and test sets.

- In Machine Learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'test set'. An important point to note is that during training the classifier only uses the training set . The test set must not be used during training the classifier. The test set will only be available during testing the classifier.
- Training set - a subset to train a model.(Model learns patterns between Input and Output)

- Test set - a subset to test the trained model.(To test whether the model has correctly learnt)
- The amount or percentage of Splitting can be taken as specified.
- First we need to identify the input and output variables and we need to separate the input set and output set.
- In scikit learn library we have a package called model_selection in which train_test_split method is available. We need to import this method.
- This method splits the input and output data to train and test based on the percentage specified by the user and assigns them to four different variables.

7.3. Make Predictions:

Then we have to test the model for the test set ,that is done as follows:

- We have a method called predict , using this method we need to predict the output for input test set and we need to compare the out but with the output test data.
- If the predicted values and the original values are close then we can say that model is trained with good accuracy.

7.4. Validate the Models:

Model validation is the process of evaluating a trained model on a test data set. This provides the generalization ability of a trained model. Here I provide a step by step approach to complete the first iteration of model validation in minutes.

- The models are validated after completion of training and testing the model.
- Checking the accuracy scores as metrics to validate the models.
- We have to check the accuracy among the models and validate the best model among those.

We will be using accuracy_score, confusion_matrix, classification_report methods from the sklearn.metrics to validate the models.

Here we used two numpy arrays train_accu_score and test_accu_score to store the train and test accuracies of the models respectively, in order to compare them in the end.

7.4.1 Accuracy Score:

The accuracy_score function of the sklearn.metrics module computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions.

In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly matches with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

If $\hat{y}_i$ is the predicted value of the *i*-th sample and y_i is the corresponding true value, then the fraction of correct predictions over n_{samples} is defined as:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

7.4.2 Confusion Matrix:

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

7.4.3 Classification Report:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification.

The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes. There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative:** when a case was negative and predicted negative
2. **TP / True Positive:** when a case was positive and predicted positive
3. **FN / False Negative:** when a case was positive but predicted negative
4. **FP / False Positive:** when a case was negative but predicted positive

i. Logistic Regression Validation:

```
##checking the metrics:
train_accu_score=np.zeros(4) ##to store the training accuracies of all the models
test_accu_score=np.zeros(4) ##to store the testing accuracies of all models
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
train_accu_score[0]=accuracy_score(y_train,lr_train_predict)
test_accu_score[0]=accuracy_score(y_test,lr_test_predict)
```

```
print("Logistic Regression Train Accuracy: ",train_accu_score[0])
print("Logistic Regression Test Accuracy: ",test_accu_score[0])
```

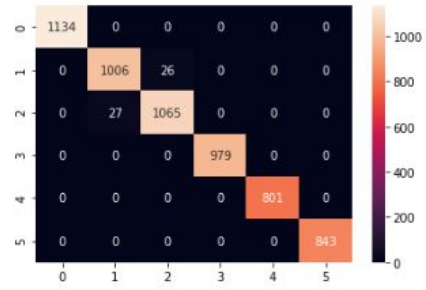
```
Logistic Regression Train Accuracy:  0.9909879272232613
Logistic Regression Test Accuracy:  0.9864038069340585
```

Fig 7.4.1 Logistic Regression accuracy_score

```
print("Logistic Regression Train Confusion Matrix: ")
sns.heatmap(confusion_matrix(y_train,lr_train_predict),annot=True,fmt='1.0f')
```

```
Logistic Regression Train Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7efd03e15410>
```




```
print("Logistic Regression Test Confusion Matrix: ")
sns.heatmap(confusion_matrix(y_test,lr_test_predict),annot=True,fmt='1.0f')
```

```
Logistic Regression Test Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7efd0003e290>
```

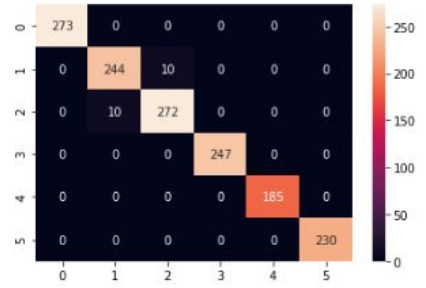


Fig 7.4.2 Logistic Regression confusion_matrix

ii. K-Nearest Neighbors Validation:

- Before finding optimum n_neighbors value

```
#checking the metrics
print("train: ",accuracy_score(y_train,knc_init_train_predict))
print("test: ",accuracy_score(y_test,knc_init_test_predict))
```

```
train:  0.9600408093861588
test:  0.9544527532290958
```

Fig 7.4.3 KNN accuracy_score before elbow curve

- After finding the optimum n_neighbors value using the Elbow Curve

```
#checking the metrics
train_accu_score[1]=accuracy_score(y_train,knc_train_predict)
test_accu_score[1]=accuracy_score(y_test,knc_test_predict)
print("KNN Classifier Train Accuracy: ",train_accu_score[1])
print("KNN Classifier Test Accuracy: ",test_accu_score[1])
```

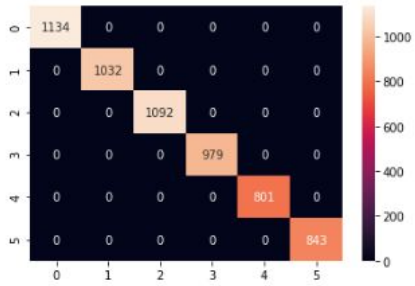
```
KNN Classifier Train Accuracy:  1.0
KNN Classifier Test Accuracy:  0.9741672331747111
```

Fig 7.4.4 KNN accuracy_score after elbow curve

```
print("K-Nearest Neighbors Train Confusion Matrix: ")
sns.heatmap(confusion_matrix(y_train,knc_train_predict),annot=True,fmt='1.0f')
```

```
K-Nearest Neighbors Train Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7efcf8dccb90>
```




```
print("K-Nearest Neighbors Test Confusion Matrix: ")
sns.heatmap(confusion_matrix(y_test,knc_test_predict),annot=True,fmt='1.0f')
```

```
K-Nearest Neighbors Test Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7efcf8bf6ed0>
```

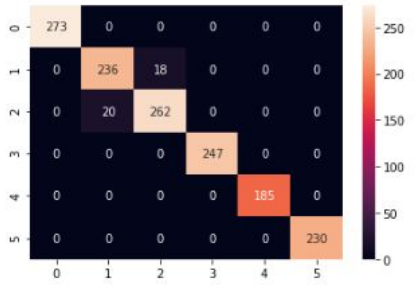


Fig 7.4.5 KNN confusion_matrix

iii. Gaussian Naive Bayes Classifier:

```
#checking the metrics
train_accu_score[2]=accuracy_score(y_train,nb_train_predict)
test_accu_score[2]=accuracy_score(y_test,nb_test_predict)
print("Naive Bayes Classifier Train Accuracy: ",train_accu_score[2])
print("Naive Bayes Test Accuracy: ",test_accu_score[2])
```

```
Naive Bayes Classifier Train Accuracy:  0.730998129569801
Naive Bayes Test Accuracy:  0.7307953772943576
```

Fig 7.4.6 GaussianNB accuracy_score

```
print("Naive Bayes Train Confusion Matrix: ")
sns.heatmap(confusion_matrix(y_train,nb_train_predict),annot=True,fmt='1.0f')
```

```
Naive Bayes Train Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7efcf8c6fb50>
```


```
print("Naive Bayes Test Confusion Matrix: ")
sns.heatmap(confusion_matrix(y_test,nb_test_predict),annot=True,fmt='1.0f')
```

```
Naive Bayes Test Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7efcf8b80250>
```

Fig 7.4.7 GaussianNB confusion_matrix

iv. Random Forest:

```
#checking the metrics
train_accu_score[3]=accuracy_score(y_train,rfc_train_predict)
test_accu_score[3]=accuracy_score(y_test,rfc_test_predict)
print("Random Forest Classifier Train Accuracy: ",train_accu_score[3])
print("Random Forest Classifier Test Accuracy: ",test_accu_score[3])
```

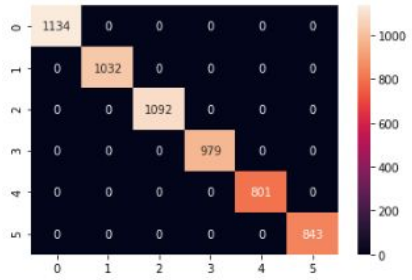
```
Random Forest Classifier Train Accuracy:  1.0
Random Forest Classifier Test Accuracy:  0.9809653297076818
```

Fig 7.4.8 Random Forest accuracy_score

```
print("Random Forest Classifier Train Confusion Matrix: ")
sns.heatmap(confusion_matrix(y_train,rfc_train_predict),annot=True,fmt='1.0f')
```

```
Random Forest Classifier Train Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7efcf39044d0>
```




```
print("Random Forest Classifier Test Confusion Matrix: ")
sns.heatmap(confusion_matrix(y_test,rfc_test_predict),annot=True,fmt='1.0f')
```

```
Random Forest Classifier Test Confusion Matrix:

<matplotlib.axes._subplots.AxesSubplot at 0x7efcf37a9fd0>
```

Fig 7.4.9 Random Forest confusion_matrix

7.5. Parameter Tuning:

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters, and have to be tuned so that the model can optimally solve the machine learning problem. Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss. Cross-validation is often used to estimate this generalization performance.

- **Grid search**:

  The traditional way of performing hyperparameter optimization has been grid search, or a parameter sweep, which is simply an exhaustive search through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.

  Since the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters, manually set bounds and discretization may be necessary before applying grid search.

  For example, a typical soft-margin SVM classifier equipped with an RBF kernel has at least two hyperparameters that need to be tuned for good performance on unseen data: a regularization constant C and a kernel hyperparameter γ. Both parameters are

continuous, so to perform grid search, one selects a finite set of "reasonable" values for each.

However, our model has achieved best possible accuracy without the hyper parameter tuning. Hence, this step is not necessary.

- **Random search:**

  Random Search replaces the exhaustive enumeration of all combinations by selecting them randomly. This can be simply applied to the discrete setting described above, but also generalizes to continuous and mixed spaces. It can outperform Grid search, especially when only a small number of hyperparameters affects the final performance of the machine learning algorithm In this case, the optimization problem is said to have a low intrinsic dimensionality. Random Search is also embarrassingly parallel, and additionally allows the inclusion of prior knowledge by specifying the distribution from which to sample.

- **Bayesian optimization:**

  Bayesian optimization is a global optimization method for noisy black-box functions. Applied to hyperparameter optimization, Bayesian optimization builds a probabilistic model of the function mapping from hyperparameter values to the objective evaluated on a validation set. By iteratively evaluating a promising hyperparameter configuration based on the current model, and then updating it, Bayesian optimization, aims to gather observations revealing as much information as possible about this function and, in particular, the location of the optimum. It tries to balance exploration (hyperparameters for which the outcome is most uncertain) and exploitation (hyperparameters expected close to the optimum). In practice, Bayesian optimization has been shown to obtain better results in fewer evaluations compared to grid search and

random search, due to the ability to reason about the quality of experiments before they are run.

7.6. Predictions from raw Data:

- After completion of accuracy we have to predict best model among them onto raw data.
- Load the test dataset to predict.
- Check the rows and columns and also shape.
- After completion, predict using the best algorithm onto the training dataset.
- Add a new column of predicted data i.e output.
- Check the relation between test and train dataset to verify output.
- As shown below we chose Random Forest as the best model to classify our data.

```
#splitting the test DataFrame into input and output
inp=test.drop(['Activity'],axis=1)
out=test.Activity
```

```
#using the pre-trained Logistic Regression model to classify out input from 'test' DataFrame
pred=lr.predict(inp)
print("Accuracy of Logistic regression on Unseen data: ",accuracy_score(out,pred))
```

```
Accuracy of Logistic regression on Unseen data:  0.9558873430607397
```

7.6.1 Predicting on Raw Data

8. CONCLUSION

It is concluded after performing thorough Exploratory Data analysis which include statistical models which are computed to get accuracy and also heat maps which are computed to get a clear understanding of the data set and it is come to point of getting the solution for the p90roblem statement being , that John wanted to find out the best algorithm to predict and classify the daily activities done by him.

From our experiments upon the given data using different machine learning models namely, Logistic regression, K-Nearest Neighbors Classifier, Gaussian Naive Bayes Classifier, Random Forest, we have found out that Logistic regression algorithm has given a high yield of accuracy. Hence, it can be concluded that the same algorithm is best suited for our data model.

9. REFERENCES

[1] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[2] https://en.wikipedia.org/wiki/Machine_learning

[3]https://towardsdatascience.com/supervised-machine-learning-model-validation-a-step-by-step-approach-771109ae0253

[4] https://builtin.com/data-science/random-forest-algorithm