THAKUR COLLEGE OF SCIENCE AND COMMERCE
KANDIVALI EAST,
MUMBAI

A PROJECT ON
HUMAN ACTIVITY RECOGNIZATION THROUGH SMARTPHONE DATASET.

A PROJECT SUBMITTED TO

THE UNIVERSITY OF MUMBAI FOR PARTIAL COMPLETION OF THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

UNDER THE FACULTY OF SCIENCE

BY

Nabil Hafiz Mohd Shah

UNDER THE GUIDANCE OF
DR. GIRISH TERE

Thakur College of Science and Commerce
Kandivali (East),
Mumbai.

[Permanently Affiliated to UNIVERSITY OF MUMBAI]

ACADEMIC YEAR
2023 – 2024

THAKUR COLLEGE OF SCIENCE AND COMMERCE

KANDIVALI (EAST)

MUMBAI

A PROJECT REPORT ON

HUMAN ACTIVITY RECOGNIZATION THROUGH SMARTPHONE DATASET

For

Thakur College of Science and Commerce

By

Nabil Hafiz Mohd Shah

Submitted in partial fulfilment of

Bachelors of Science (Computer Science)

[UNIVERSITY OF MUMBAI]

Thakur Degree College of Science and Commerce Kandivali (East), Mumbai.

ACADEMIC YEAR

2023 – 2024

Thakur Educational Trust's (Regd.)
THAKUR COLLEGE OF SCIENCE & COMMERCE
AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI
NAAC Accredited Grade 'A' (3rd Cycle) & ISO 9001: 2015 (Certified)
Best College Award by University of Mumbai for the Year 2018-2019
CELEBRATING 25 YEARS OF GLORY

Department of Computer Science

(2023-2024)

Certificate of Approval

This is to certify that the project work entitled **"HUMAN ACTIVITY RECOGNIZATION THROUGH SMARTPHONE DATASET"** is prepared by **Nabil Hafiz Mohd Shah** a student of **"Third Year Bachelor of Science (Computer Science)"** Program of Thakur College of Science & Commerce (University of Mumbai).

This is the original study work and important sources used have been duly acknowledged in the report. The report is submitted in partial fulfilment of Bachelors of Science (Computer Science) course as per rules of University of Mumbai.

_____ _____

Project Guide,
Dr.Girish Tere

HOD,
Prof. Ashish Trivedi

_____

External Examiner

DECLARATION BY LEARNER

I the undersigned **Mr. Nabil Hafiz Mohd Shah** hereby, declare that the work embodied in this project work titled **"HUMAN ACTIVITY RECOGNIZATION THROUGH SMARTPHONE DATASET"** forms my own contribution to the research work carried out under the guidance of **DR.GIRISH TERE** is a result of my own research work and has not been previously submitted to any other University for any other Degree/Diploma to this or any other University.

Wherever reference has been made to previous works of others, it has been clearly indicated as such and included in the bibliography. I, hereby further declare that all information of this document has been obtained and presented in accordance with academic rules and ethical conduct.

Certified By

DR. GIRISH TERE

Name and signature of Learner

Nabil Hafiz Mohd Shah

INDEX

THAKUR COLLEGE OF SCIENCE AND COMMERCE

Kandivali (East), Mumbai.

Students Name: Nabil Hafiz Mohd Shah

Project Name : Human Activity Recognition Through Smartphone Dataset.

College Name: Thakur College of Science and Commerce.

Phases	Expected Date of Completion	Actual Date of Completion	SIGNATURE
Preliminary Investigation			
System Analysis			
System Designing			
System Coding			
System Implementation			
Report Submission			

ACKNOWLEDGEMENT

Achievement is finding out what you would be doing rather than what you have to do. It is not until you undertake such a project that you realize how much effort and hard work it really is, what are your capabilities and how well you can present yourself or other things. It gives me immense pleasure to present this report towards the fulfilment of my project.

It has been rightly said that we are built on the shoulder of others. For everything I have achieved, the credit goes to all those who had helped me to complete this project successfully.

I take this opportunity to express my profound gratitude to management of Thakur Degree College of Science & Commerce for giving me this opportunity to accomplish this project work.

I am very much thankful to **Mrs. C. T. Chakraborty** - Principal of Thakur College for their kind co-operation in the completion of my project. A special vote of thanks to our HOD **Mr. ASHISH TRIVEDI** and to our project guide **DR. GIRISH TERE**

Finally, I would like to thank all my friends & entire Computer Science department who directly or indirectly helped me in completion of this project & to my family without whose support, motivation & encouragement this would not have been possible.

(NABIL SHAH)

ORGANIZATION OVERVIEW

The Thakur College of Science and Commerce (TCSC) is a college in Kandivali in Mumbai of Maharashtra, India running by Thakur EducationalTrust.

Thakur College was started in 1992 to serve the needs of students passing SSC examination from the schools around Kandivali area and Thakur Vidhya Mandir which has already established itself as one of the schools in the area. It offers courses at primarily the higher secondary and under-graduate levels. The courses at the undergraduate and post-graduate level are offered in affiliation with Mumbai University, Mumbai. An ISO 9001:2008 College with A grade as assessed by the National Assessment and Accreditation Council NAAC.

Name: Thakur College of Science and Commerce Founded: 1997

Address: Thakur College of Science and Commerce, Thakur Village Kandivali(E), Mumbai – 400 001

Motto: Journey towards Excellence Total Staff: 200

Number of Students: 12500 Email: Helpdesk@tcsc.org.in

Hardware Requirements:

1. 1.6 GHz or faster processor
2. 1 GB of RAM
3. 1 GB of free Internal Storage
4. Port requirements: Port 8000 plus 5
5. Unique, random ports 5 CPU cores

Software Requirements:

1. Operating System: Window, Linux, Mac, Android
2. Software: Jupyter notebook, visual studio
3. Language: Python

Description:

In the landscape of human activity recognition utilizing smartphone datasets, the precise identification of various activities holds significant importance for enhancing user experience and optimizing resource allocation. However, manual analysis often introduces challenges, as uncertainties persist in accurately discerning activities, leading to inefficiencies in data processing. To address this, leveraging machine learning techniques, particularly the Random Forest Algorithm, within an automated activity recognition system proves to be a promising solution.

By harnessing the power of machine learning, this automated system efficiently predicts and classifies human activities based on data collected from smartphones. The Random Forest Algorithm, known for its ability to handle complex datasets and provide accurate predictions, serves as a robust framework for this purpose. Trained on a diverse and comprehensive dataset encompassing various activities and scenarios, the system enables the machine to comprehend intricate patterns and nuances inherent in human behaviour.

As a result, the automated activity recognition system autonomously evaluates human activities, offering quick and precise results. This not only enhances user experience by improving the functionality of applications reliant on activity recognition, such as fitness trackers and healthcare monitoring tools, but also benefits data analysts. With reduced reliance on manual interpretation, analysts can focus on more complex analysis tasks, thereby enhancing overall efficiency and effectiveness in data processing.

Furthermore, the implications of such technology extend beyond individual user experiences to broader societal benefits. For instance, in healthcare settings, accurate activity recognition can aid in remote patient monitoring, facilitating early detection of health issues and personalized interventions. In urban planning, it can contribute to better understanding of pedestrian and traffic patterns, leading to improved infrastructure design and resource allocation.

Introduction

What is Human Activity Recognition Through Smartphone Dataset Using Machine learning?

Human activity recognition through smartphone datasets using machine learning involves the process of analyzing sensor data collected from smartphones to automatically identify and classify different human activities, such as walking, running, sitting, or standing. This is achieved by training machine learning models on labeled datasets, where each data point consists of sensor readings (e.g., accelerometer, gyroscope) along with the corresponding activity label. The trained models learn patterns and relationships in the sensor data to accurately predict the activity being performed by the user. This technology finds applications in various domains such as healthcare monitoring, fitness tracking, and behavior analysis, offering insights into human behavior and enabling the development of intelligent mobile applications.

EXISTING SYSTEM

In the realm of human activity recognition using smartphone datasets, conventional methods rely on manual assessment of various factors, leading to time-consuming processes. To streamline and enhance accuracy, researchers employ artificial neural network models for predicting activity patterns. A Feed-forward back propagation neural network is utilized to forecast activity trends. Recognizing the challenges of accurate predictions, researchers implement ensemble techniques such as bagging and boosting, eventually adopting the random forest technique. This amalgamation of classifiers significantly enhances the efficiency of activity recognition.

In their study, researchers explore various ensemble techniques tailored for binary and multi-class classification in the context of human activity recognition. Notably, they introduce a novel technique named COB, which exhibits effective classification performance. However, it is acknowledged that COB may be susceptible to noise and outlier data in the prediction process. Despite this compromise, researchers conclude that ensemble-based algorithms markedly improve the results for training datasets, showcasing their potential to revolutionize the accuracy and efficiency of human activity recognition.

DRAWBACK OF EXISTING SYSTEM

Analysing and managing diverse details for human activity recognition using smartphone datasets manually is both time-intensive and susceptible to errors. The intricate nature of activity patterns requires thorough examination, making manual verification prone to inaccuracies and potentially leading to erroneous activity classifications. The implementation of machine learning techniques offers a solution to these challenges, automating the activity recognition process for smartphone.

PROPOSED SYSTEM

To address challenges in human activity recognition using smartphone datasets, an automated system is developed, leveraging machine learning, particularly the Random Forest Algorithm. Through comprehensive training on a diverse dataset, the model learns intricate patterns, enhancing its understanding of activity recognition nuances. This trained model accurately evaluates human activity, providing precise outcomes. By employing machine learning algorithms on the training set, the optimal performer is identified, and subsequent predictions on the test set are generated. Implementation is streamlined through the Flask Framework, ensuring efficient deployment of the human activity recognition model.

ADVANTAGE:

The human activity recognition system significantly reduces the time required for activity classification, introducing a fully automated process to minimize human error and enhance efficiency. Swift recognition of various activities expedites data analysis, effectively eliminating delays in understanding human behaviour patterns.

Tools and Techniques Used:

- Python
- Python Libraries: Pandas, NumPy, Matplotlib, Scikit learn, Seaborn.
- Visual Studio Code
- Jupyter Notebook

What is Python?

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

What are Python Libraries?

The Python Standard Library contains the exact syntax, semantics, and tokens of Python. It contains built-in modules that provide access to basic system functionality like I/O and some other core modules. Most of the Python Libraries are written in the C programming language. The Python standard library consists of more than 200 core modules. All these work together to make Python a high-level programming language. Python Standard Library plays a very important role. Without it, the programmers can't have access to the functionalities of Python. But other than this, there are several other libraries in Python that make a programmer's life easier.

A Python library is merely a bunch of code scripts or modules of codes that we can utilize in a program for specific operations, as stated above.

We use libraries to don't have to rewrite code already written in our program. However, here's how it works. The library files have a DLL extension in the MS Windows environment (Dynamic Load Libraries). When we import a library to our program and run it, the linker looks for that library automatically. It extracts the library's functions and then interprets the program accordingly. This is how we use library methods in our program. We'll look at how we integrate libraries into our Python programs in more detail later.

Pandas:

Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools. It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc.

Numpy:

The name Numpy stands for Numerical Python. It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data. It consists of in-built mathematical functions for easy computations. Even libraries like TensorFlow use Numpy internally to perform several operations on tensors. Array Interface is one of the key features of this library.

Matplotlib:

This library is responsible for plotting numerical data. And that's why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.

Scikit-learn:

It is a famous Python library to work with complex data. Scikit-learn is an open- source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc.

This library works in association with Numpy and SciPy.

Seaborn:

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas. Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset.

Visual Studio Code:

Visual Studio Code is a code editor in layman's terms. Visual Studio Code is <a free-editor that helps the programmer write code, helps in debugging and corrects the code using the intelli-sense method

In normal terms, it facilitates users to write the code in an easy manner. Many people say that it is half of an IDE and an editor, but the decision is up to to the coders. Any program/software that we see or use works on the code that runs in the background.

Traditionally coding was used to do in the traditional editors or even in the basic editors like notepad! These editors used to provide basic support to the coders.

Some of them were so basic that it was very difficult in writing basic English level programs in them. As time went by, some programming languages needed a specific framework and support for further coding and development it, which was not possible using these editors.

VI Editor, Sublime Text Editor, is one of the many kinds of editors that came into existence. The most prominent and which supports almost every coding language is VISUAL STUDIO CODE. Its features let the user modify the editor as per the usage, which means the user is able to download the libraries from the internet and integrate it with the code as per his requirements.

Reasons for using the Python language in Machine Learning.

It has a huge number of libraries and frameworks: The Python language comes with many libraries and frameworks that make coding easy. This also saves a significant amount of time.

The most popular libraries are NumPy, which is used for scientific calculations; SciPy for more advanced computations; and scikit, for learning data mining and data analysis. These libraries work alongside powerful frameworks like TensorFlow, CNTK, and Apache Spark. These libraries and frameworks are essential when it comes to machine and deep learning projects.

Simplicity: Python code is concise and readable even to new developers, which is beneficial to machine and deep learning projects. Due to its simple syntax, the development of applications with Python is fast when compared to many programming languages. Furthermore, it allows the developer to test algorithms without implementing them. Readable code is also vital for collaborative coding. Many individuals can work together on a complex project.

The massive online support: Python is an open-source programming language and enjoys excellent support from many resources and quality documentation worldwide. It also has a large and active community of developers who provide their assistance at any stage of development. Most scientists have adopted Python for Machine Learning and Deep Learning projects, which means most of the brightest minds worldwide, can be found in Python communities.

Fast development: Python has a syntax that is easy to understand and friendly. Furthermore, the numerous frameworks and libraries boost software development. By using out-of-box solutions, a lot can be done with a few lines of code. Python is good for developing prototypes, which boosts productivity.

Flexible integrations: Python projects can be integrated with other systems coded in different programming languages. This means that it is much easier to blend it with other AI projects written in other languages. Also, since it is extensible and portable, Python can be used to perform cross languages tasks. The adaptability of Python makes it easy for data scientists and developers to train machine learning models.

Fast code tests: Python provides a lot of code review and test tools. Developers can quickly check the correctness and quality of the code. AI projects tend to be time-consuming, so a well-structured environment for testing and checking for bugs is needed. Python is the ideal language since it supports these features.

Performance: Some developers argue that Python is relatively slow compared to other programming languages. As much as speed is not one of Python's strong suits, it provides the solution known as Cython. It is a superset of Python language designed to achieve code performance the same as C language Developers can use Cython to code C extensions the same way they code in Python, as its syntax is almost the same. Cython increases the language performance significantly.

Visualization tools: Python comes with a wide variety of libraries. Some of these frameworks offer good visualization tools. In AI, Machine learning, and Deep learning, it is important to present data in a human-readable format. Therefore, Python is a perfect choice for implementing this feature. Some libraries like Matplotlib enable data scientists to generate charts, histograms, and plots to represent data and visualization better. Also, the different APIs that Python supports enhance the visualization process.

Steps to solve machine learning projects:

- Data selection.

- Data description: A story of what the data is all about and the features present in the data.

- Performing both statistical and graphical data analysis.

- Data transformation and derivation of new attributes, if necessary. Selection of machine learning algorithms based on the patterns observed in EDA.

- Data standardization and normalization. Creation of train and test data sets.

- Model training using machine learning algorithms.

- Calculation of model accuracy: both training and testing accuracy. Hyper parameter tuning to achieve a better accuracy.

- Saving the created model file. Deployment strategies for model. Production deployment and testing.

Random Forest for Human Activity Recognition Through Smartphone Dataset using Machine Learning:

The Random Forest algorithm enhances the adaptability and decision-making capabilities of individual trees, constituting another valuable machine learning approach founded on the ensemble learning theorem. By amalgamating outcomes from diverse decision trees, it strives to optimize the training process. In the realm of human activity recognition using smartphone datasets, where accurate classification is crucial, certain features may carry more significance than others. Specifically, identifying features whose exclusion would enhance overall performance is essential. Leveraging the foundational principles of decision trees and their feature selection based on information gain, the Random Forest algorithm incorporates these advantages to deliver superior performance in predicting human activity patterns.

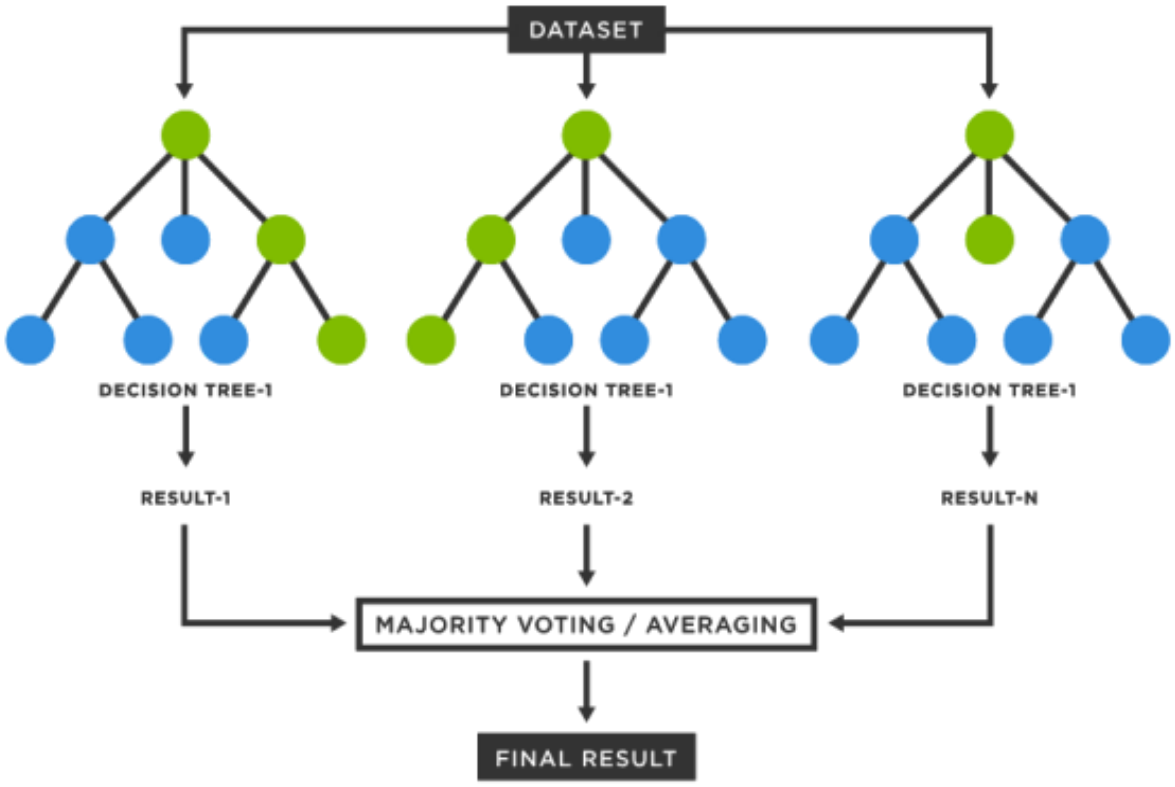


Figure 1 Random Forest Algorithm

Logistic Regression for Human Activity Recognition Through Smartphone Dataset using Machine Learning:

Logistic regression offers a straightforward yet effective method for classifying human activities based on smartphone sensor data. By pre-processing and splitting the dataset, training and evaluating the model, and refining its performance through hyperparameter tuning, logistic regression can provide reliable predictions. Continuous monitoring and potential exploration of advanced algorithms ensure adaptability and optimization for real-world deployment, contributing to the seamless integration of Human Activity Recognition technology into various applications.

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

The above equation is the final equation for Logistic Regression.

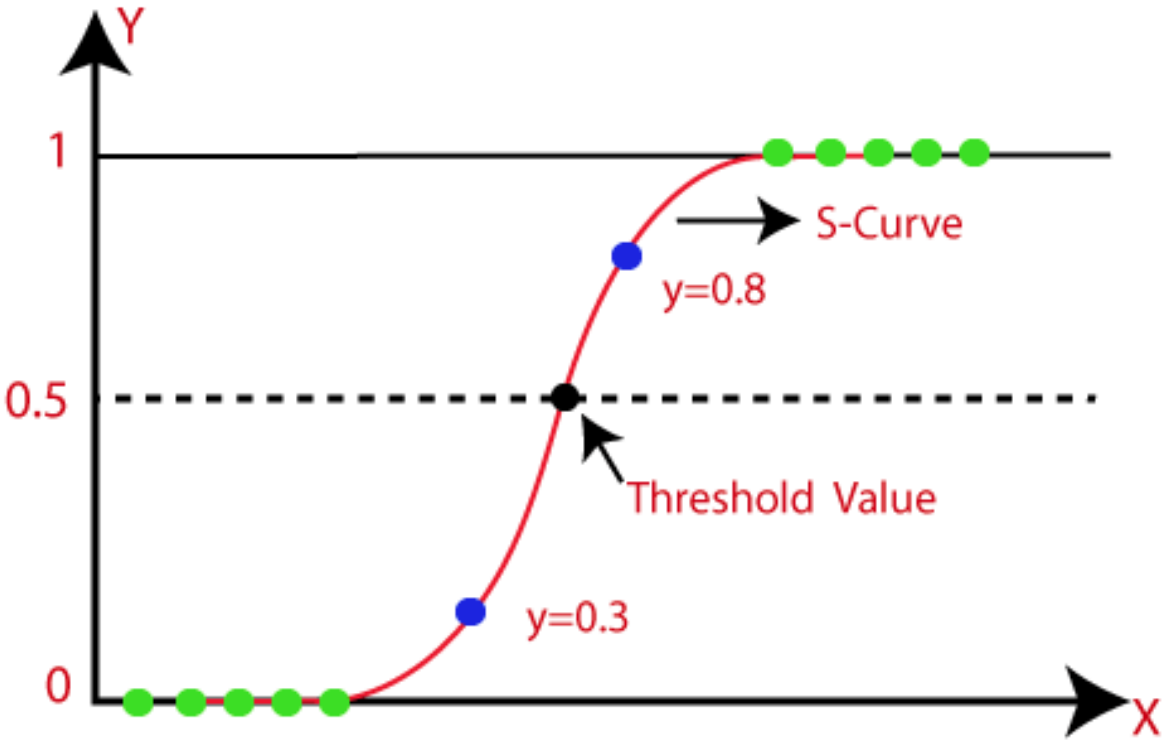


Figure 2 Logistic Regression

Decision Tree for Human Activity Recognition Through Smartphone Dataset using Machine Learning:

Decision trees offer a straightforward yet powerful approach for Human Activity Recognition (HAR) through smartphone datasets. Their ability to handle non-linear relationships, scalability, interpretability, and efficiency make them suitable for diverse applications. By training and evaluating decision tree models, users can achieve accurate activity classification, paving the way for practical implementation in real-world scenarios such as healthcare, fitness tracking, and context-aware computing. Additionally, the interpretability of decision trees facilitates understanding and trust in the model's predictions, making them valuable tools for both researchers and end-users alike.

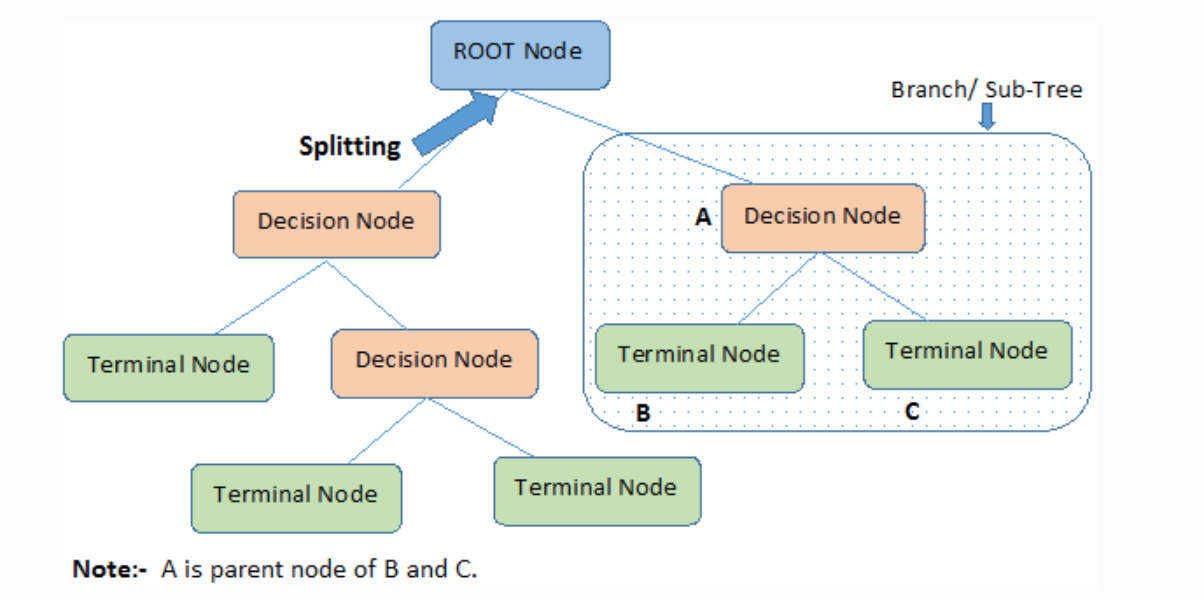


Figure 3 Decision Tree

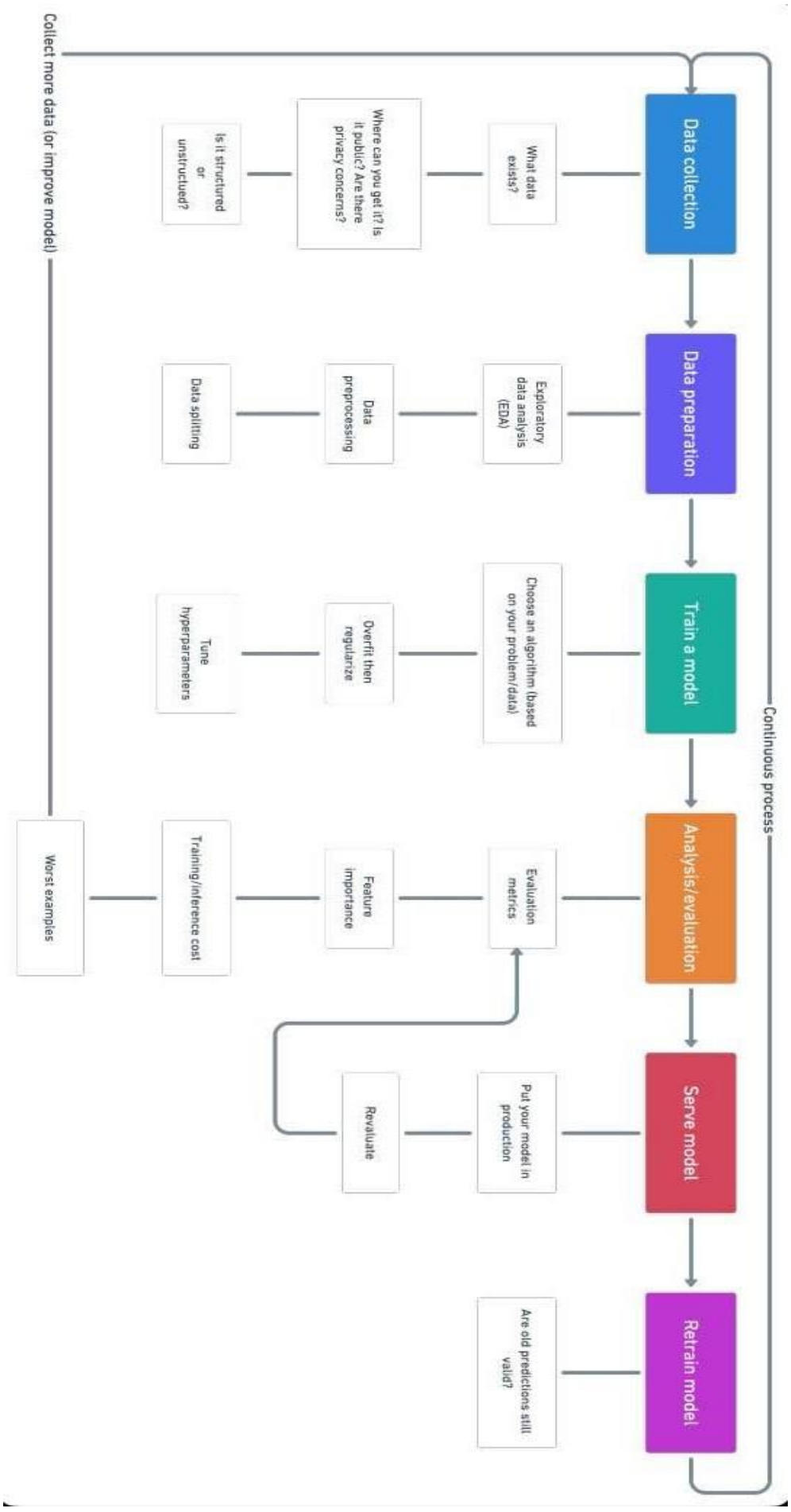


Figure 4 Basic Flow for Machine Learning working

Data Collection:

Questions to ask?

What kind of problem are we trying to solve?

What data sources already exist?

What privacy concerns are there?

Is the data public?

Where should we store the files?

Types of data:

Structured data: appears in tabulated format (rows and columns style, like what you'd find in an Excel spreadsheet). It contains different types of data, for example numerical, categorical, time series.

- ·**Nominal/categorical** – One thing or another (mutually exclusive).
  For example, for car scales, color is a category. A car may be blue but not white. An order does not matter.

- ·**Numerical:** Any continuous value where the difference between them matters. For example, when selling houses, $107,850 is more than $56,400.

- ·**Ordinal:** Data which has order but the distance between values is unknown.

- ·**Unstructured data:** Data with no rigid structure (images, video, speech, natural language text)

Data preparation:

Exploratory data analysis (EDA), learning about the data you're working with

1.What are the feature variables (input) and the target variable (output) For example, for predicting heart disease, the feature variables may be person's age, weight, average heart rate, and level of physical activity. And the target variable will be whether or not they have a disease.

2.What kind of do you have? Structured, unstructured, numerical, time series. Are there missing values? Should you remove them or fill them feature imputation.

3.Where are the outliers? How many of them are there? Why are they there? Are there any questions you could ask a domain expert about the data? For example, would a heart disease physician be able to shed some light on your heart disease dataset?

Data pre-processing, preparing your data to be modelled:

•Feature imputation: filling missing values (a machine learning model can't learn on data that's isn't there)

•Feature encoding (turning values into numbers). A machine learning model requires all values to be numerical)

•One hot encoding: Turn all unique values into lists of 0's and 1's where the target value is 1 and the rest are 0's. For example, when a car colours green, red blue, a green, a car's colour future would be represented as [1, 0, and 0] and a red one would be [0, 1, and 0].

•Label Encoder: Turn labels into distinct numerical values. For example, if your target variables are different animals, such as dog, cat, bird, these could become 0, 1, and 2, respectively.

•Embedding encoding: Learn a representation amongst all the different data points. For example, a language model is a representation of how different words relate to each other. Embedding is also becoming more widely available for structured (tabular) data.

•Feature normalization (scaling) or standardization: When your numerical variables are on different scales (e.g. number_of_bathroom is between 1 and 5 and size_of_land between 500 and 20000 sq. feet), some machine learning algorithms don't perform very well. Scaling and standardization help to fix this.

•Feature engineering: transform data into (potentially) more meaningful representation by adding in domain knowledge

Data splitting:

1.Training set (usually 70-80% of data): Model learns on this.

2.Validation set (usually10-15% of data): Model hyperparameters are tuned on this

3.Test set (usually 10-15% of data): Models' final performance is evaluated on this. If you have done it right, hopefully, the results on the test set give a good indication of how the model should perform in the real world. Do not use this dataset to tune the model.

Figure 5 Data splitting

Train model on data(3 steps: Choose an algorithm, overfit the model, reduce overfitting with regularization)

Choosing an algorithm:

1.Supervised algorithms – Linear Regression, Logistic Regression, KNN,

SVMs, Decision tree and Random forests, AdaBoost/Gradient Boosting

Machine(boosting)

2. Unsupervised algorithms- Clustering, dimensionality reduction(PCA, Autoencoders, t-SNE), An anomaly detection

Underfitting – happens when your model doesn't perform as well as you'd like on your data. Try training for a longer or more advanced model.

Overfitting– happens when your validation loss starts to increase or when the model performs better on the training set than on the test set.

Regularization: a collection of technologies to prevent/reduce overfitting (e.g. L1, L2, Dropout, Early stopping, Data augmentation, Batch normalization)

Hyperparameter Tuning – run a bunch of experiments with different settings and see which works best

Analysis/Evaluation:

Evaluation metrics

1.Classification- Accuracy, Precision, Recall, F1, Confusion matrix, Mean average precision (object detection)

2.Regression – MSE,MAE,R^2

3.Task-based metric – E.g. for the self-driving car, you might want to know the number of disengagements.

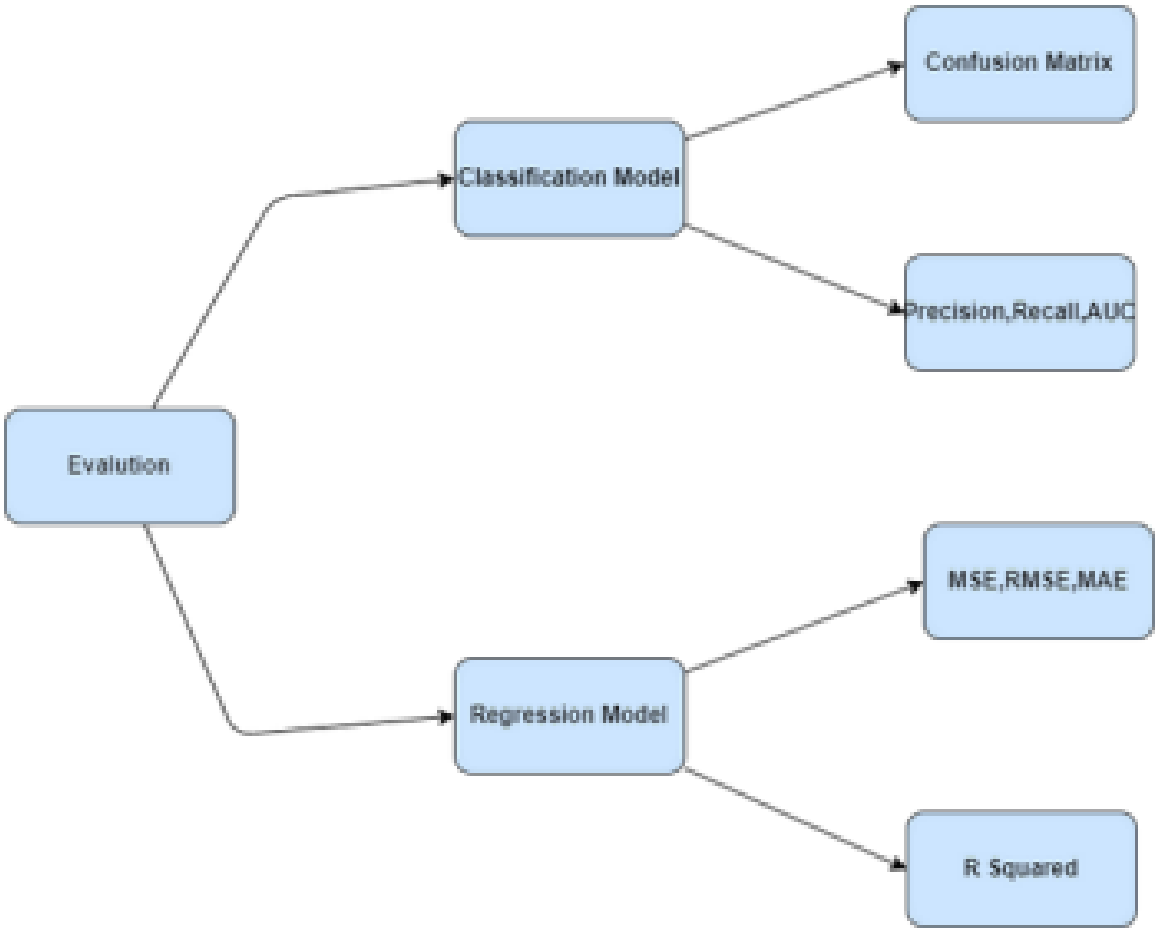


Figure 6 Evaluation

Retrain model:

See how the model performs after serving (or before serving) based on various evaluation metrics and revisit the above steps as required (remember, machine learning is very experimental, so this is where you'll want to track your data and experiments.

You'll also find your models predictions start to 8age9 (usually not in a fine-wine style) or 8drift9, as in when data sources change or upgrade(new hardware, etc.). This is when you'll want to retrain it.

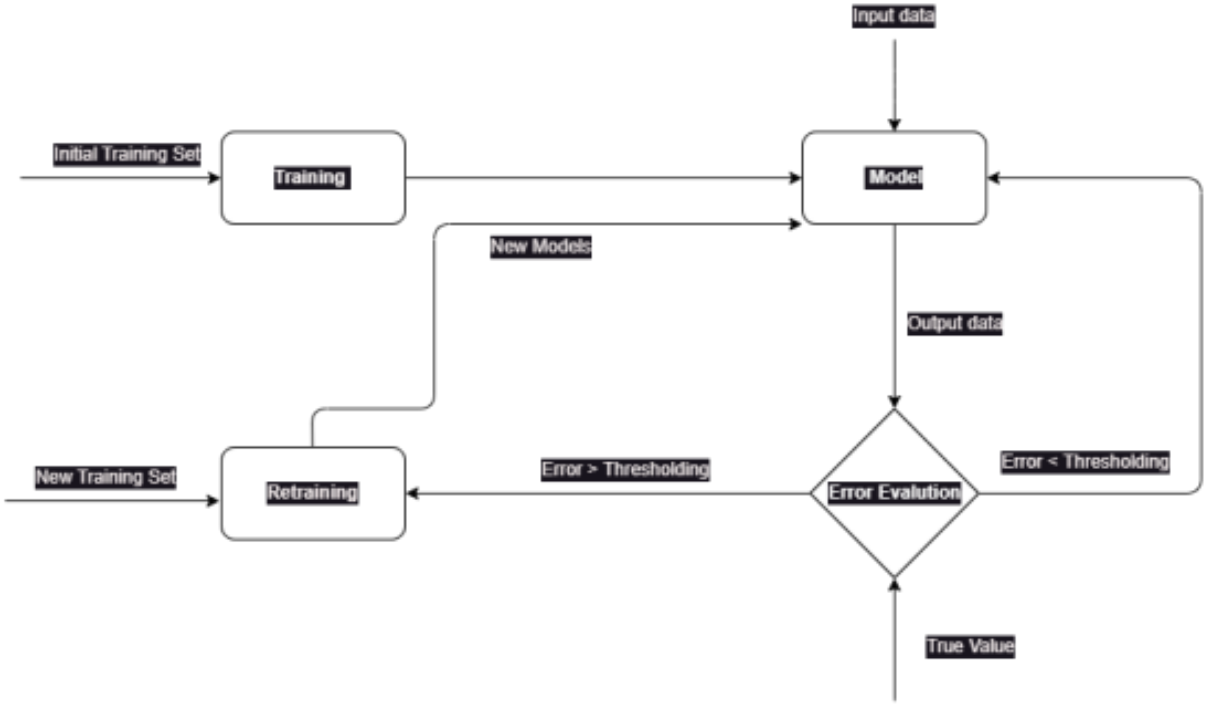


Figure 7 Retrain Model

UML Diagrams:

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system.

Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

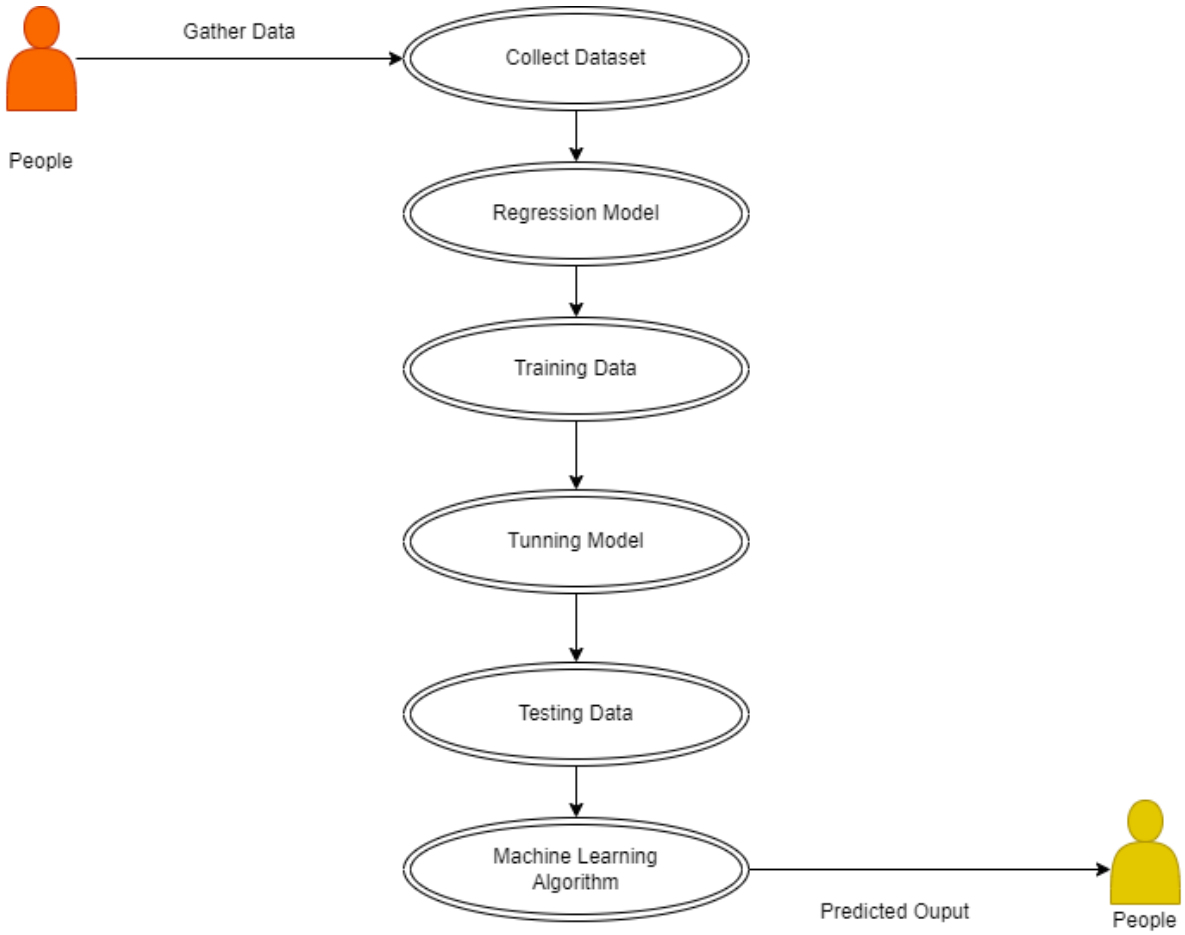


Figure 8 UML Diagram

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature.

Active class is used in a class diagram to represent the concurrency of the system.

Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

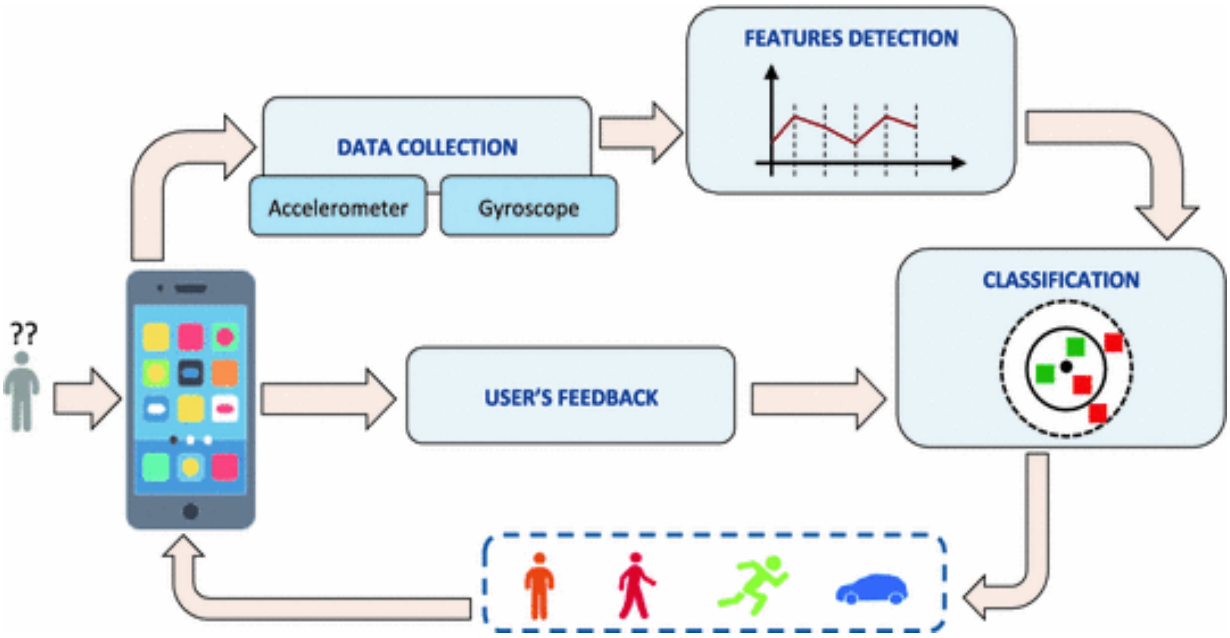


Figure 9 Real Time Execution/Deployment

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed.

Deployment diagrams are used for visualizing the deployment view of a system.

This is generally used by the deployment team.

If the above descriptions and usages are observed carefully then it is very clear that

all the diagrams have some relationship with one another.

Component diagrams are

dependent upon the classes, interfaces, etc. which are part of class/object diagram.

Again, the deployment diagram is dependent upon the components, which are used

to make component diagrams.

Class diagram

Is generally a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams will represent the whole system.

Class diagrams use packages for organization and dependency arrows to represent relationships where changes in one class may impact another. Collections of class diagrams break down a system into manageable parts, each potentially representing a module or subsystem

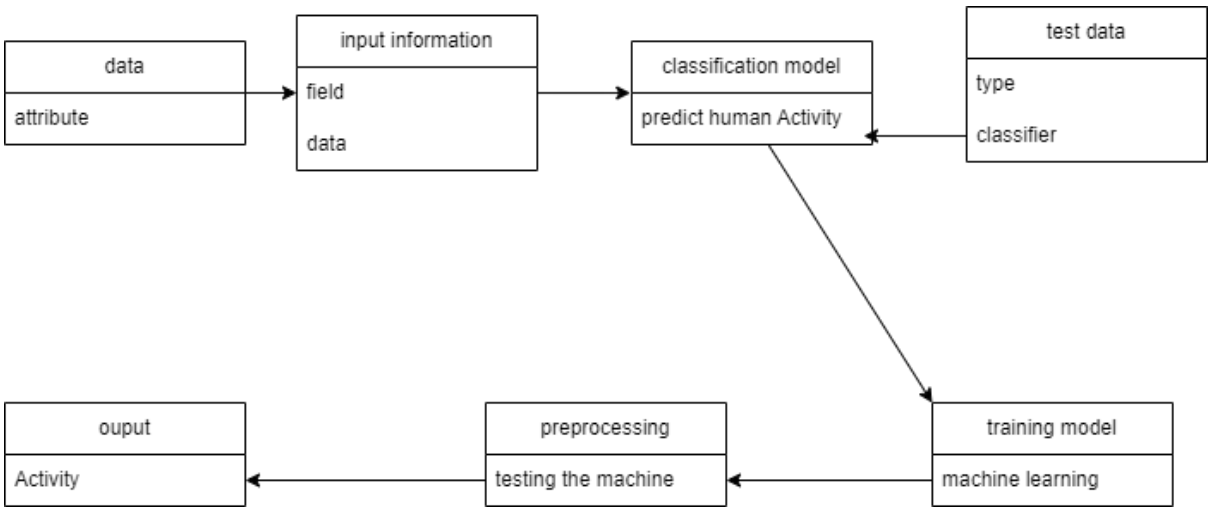


Figure 10 Class Diagram

Activity diagrams

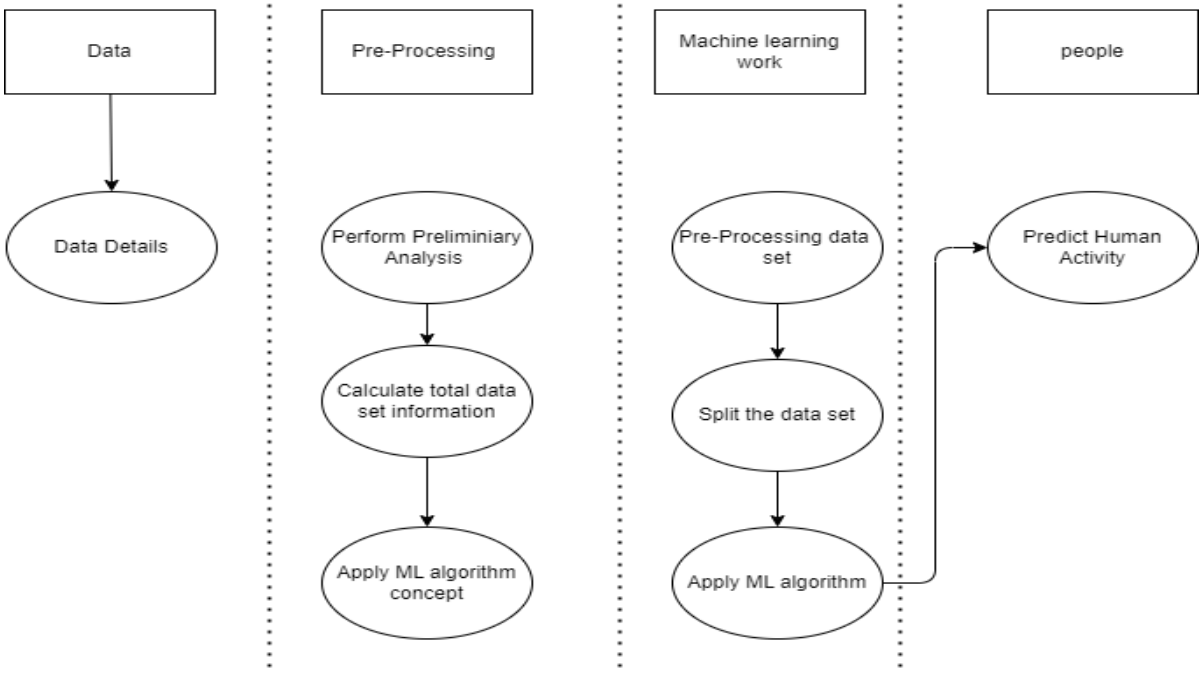


Figure 11 Activity Diagram

not only visualize the dynamic nature of a system, but they are also used for constructing executable system by using forward and reverse engineering techniques. Activity diagram is some time considered as the flow chart, but it is not.

Sequence diagrams

model the flow of logic within our system in a visual manner, enabling both to document and validate our logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artefact for dynamic modelling, which focuses on identifying the behaviour within the system.

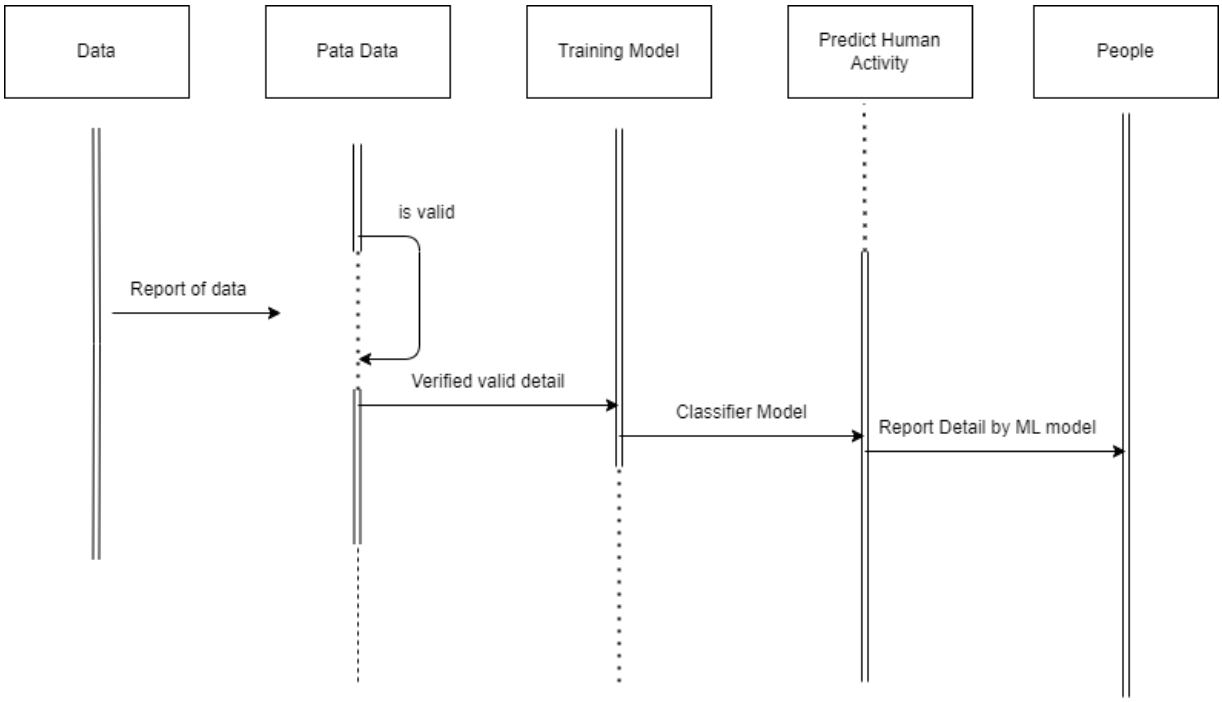


Figure 12 Sequence Diagram

An entity relationship diagram (ERD)

Is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An Entity relationship model is a data modelling technique that helps define business processes and can be used as the foundation for a relational database.

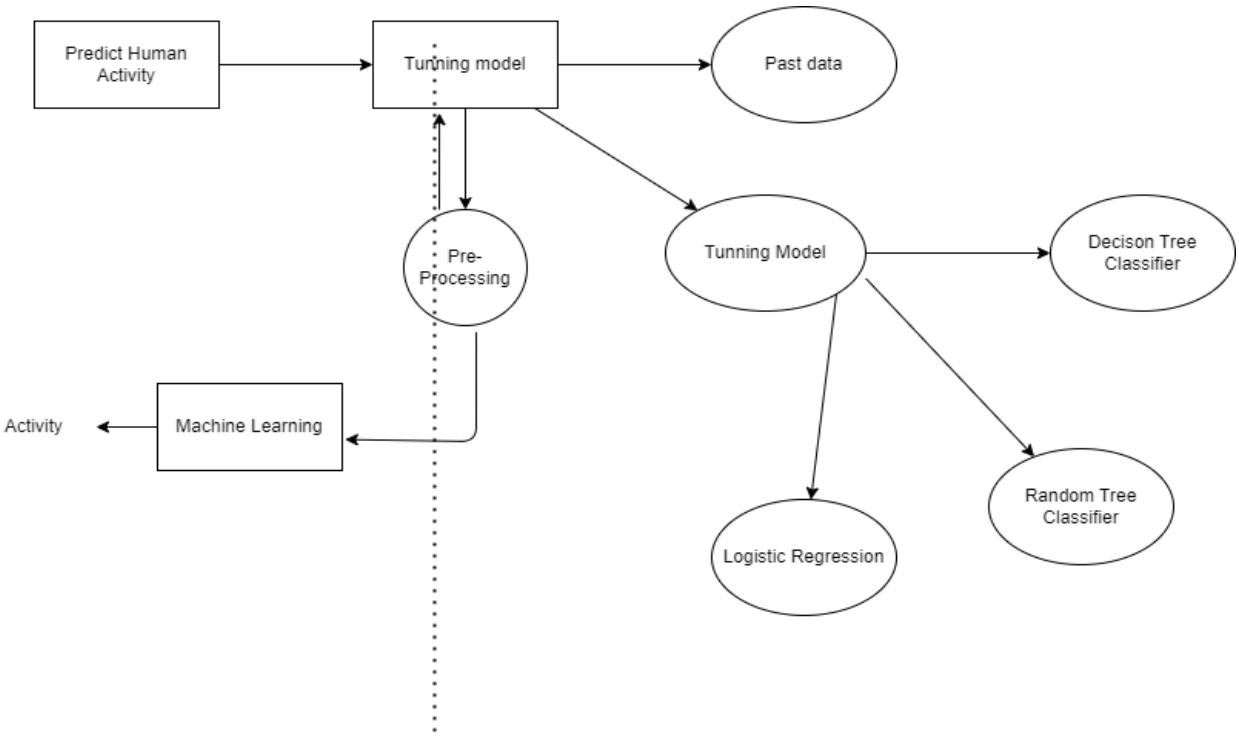


Figure 13 ERD

IMPLEMENTATION METHODOLOGY

Data Collection:

The dataset gathered for predicting flight fares is divided into a training set and a testing set, typically following an 80-20 proportion to ensure an adequate split. The data model, developed using the Random Forest algorithm, is then applied to the training set for learning patterns and relationships within the data. Subsequently, the model's performance is evaluated on the testing set to assess its predictive accuracy and generalization capabilities. This process involves making predictions on the test set to simulate real-world scenarios and measure how well the model can forecast flight fares for previously unseen data.

Pre-processing:

The collected data may contain missing values that may lead to inconsistency. To gain better results data need to be pre-processed and so it'll better the effectiveness of the algorithm. We should remove the outliers and we need to convert the variables. In order to flooring these issues we use chart function.

Data Pre-processing is a technique used for converting the raw data into a clean data set. Whenever a data is gathered from different sources, it is collected in raw format which is not feasible for the analysis of the model. For achieving better results, the model in Machine Learning method of the data has to be in a proper manner. Some of the specific Machine Learning model needs information to be in a particular format, for example, Random Forest algorithm doesn’t support null values. Therefore, to proceed further with random forest algorithm, null values need to be managed from the original raw dataset. In data pre-processing, we carry out data cleaning task with the help of Python’s Pandas library. Data cleaning is a process of removing missing, incomplete or duplicate data. The steps and techniques used for data cleaning will vary from each dataset. When combining multiple datasets, there are so many possibilities for the data to be duplicated or mislabelled or incomplete. The main objective of data cleaning is to detect and remove errors to increase the value of data in analytics and decision making to get accurate outputs.

Data Visualization:

Data visualization is a technique helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupted data, outliers, and much more. It can be used to express and demonstrate main relationships between plots and charts that are more visceral. Sometimes data does not make sense until we can look at in a visual form, such as with charts, graphs and plots. Being able to quickly visualize the data samples is an important skill both in applied statistics and in applied machine learning. It can also be used to remove outliers to get better and accurate results. It is implemented by using Python's Matplotlib library.

Train model on training data set:

Now we should train the model on the training dataset and make soothsaying's for the test dataset. We can divide our train dataset into two tract train and testimony. We can train the model on this training part and using that make soothsaying's for the testimony part. In this way, we can validate our soothsaying's as we've the true soothsaying's for the testimony part (which we don't have for the test dataset).

Correlating attributes:

Based on the observed correlation among attributes, it was noted that certain factors are more indicative of predicting flight fares accurately. Individual and significant attributes may encompass factors such as departure city, time of booking, flight distance, and historical pricing trends, with a particular emphasis on past travel history, given its perceived importance. The correlation among these attributes can be visually explored using corplot and boxplot visualizations in the Python platform for flight fare prediction.

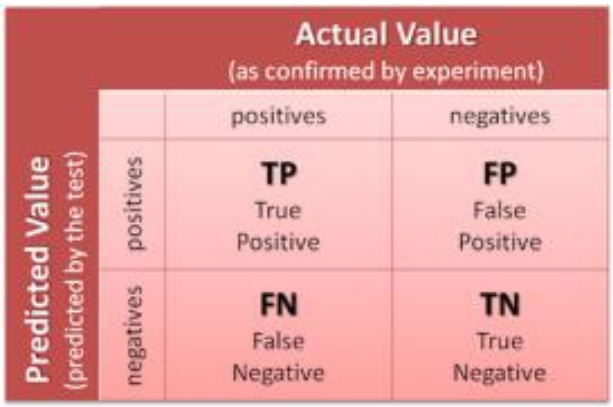


Figure 14 Correlating attributes

Random Forest Algorithm:

In the realm of human activity recognition using smartphone datasets, classification plays a pivotal role in determining the activity category to which a specific observation belongs. This capability holds immense value for various applications, such as predicting whether a certain activity falls into a particular category or forecasting the likelihood of a user engaging in a specific activity.

Similar to its application in flight fare prediction, the Random Forest algorithm operates as an ensemble of numerous individual decision trees. Each tree provides a prediction for the activity class, and the class with the most votes across all trees becomes the final prediction of the model.

The success of Random Forest is underpinned by the concept of the "wisdom of crowds," where a large number of relatively uncorrelated models (trees) collectively outperform individual models. In the context of human activity recognition, this means that the ensemble of trees can produce more accurate predictions by leveraging the low correlation between models. Just like diversified investments, uncorrelated models in a Random Forest protect each other from individual errors, leading to ensemble predictions that are more accurate than any individual prediction.

For Random Forest to perform well in human activity recognition, the prerequisites include a diverse set of features representing different aspects of human movement, and a sufficient number of trees in the ensemble to ensure robustness and accuracy in prediction.

1.There needs to be some actual signal in our features so that models built using those features do better than random guessing.

2.The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

Predicting the outcomes:

Using Random Forest algorithm, the outcomes of all applicant can be stored in any file.

Algorithm:

1. Import all the required python modules.
2. Import the database for both TESTING and TRAINING.
3. Check any NULLVALUES are exists.
4. If NULLVALUES exits ,fill the table with corresponding coding.
5. Exploratory Data Analysis for all ATTRIBUTES from the table.
6. Plot all graphs using MATPLOTLIB module.
7. Build the DECISIONTREE MODEL for the coding.
8. Send that output to CSV FILE.

GANTT CHART

A Gantt chart is a commonly used graphical depiction of a project schedule. It's a type of bar chart showing the start and finish dates of a project's elements such as resources, planning, and dependencies.

A Gantt chart is a visualization that helps in scheduling, managing, and monitoring specific tasks and resources in a project.

It consists of a list of tasks and bars depicting each task's progress.

The horizontal bars of different lengths represent the project timeline, which can include task sequences, duration, and the start and end dates for each task

It's the most widely used chart in project management. Gantt charts are used in heavy industries for projects like building dams, bridges, and highways, as well as software development and building out of other goods and services.

The Gantt chart is the most widely used chart in project management.

These charts are useful in planning a project and defining the sequence of tasks that require completion. In most instances, the chart is displayed as a horizontal bar chart.

Horizontal bars of different lengths represent the project timeline, which can include task sequences, duration, and the start and end dates for each task. The horizontal bar also shows how much of a task requires completion.

A Gantt chart helps in scheduling, managing, and monitoring specific tasks and resources in a project. The chart shows the project timeline,

which includes scheduled and completed work over a period. The Gantt chart aids project managers in communicating project status or plans and also helps ensure the project remains on track.

Benefits of a Gantt Chart:

The chart identifies tasks that may be executed in parallel and those that can't be started or finished until others are complete. It can help detect potential bottlenecks and identify tasks that may have been excluded from the project timeline.

The chart depicts things like task slack time or additional time for completion of a task that shouldn't delay the project; noncritical activities that may be delayed; and critical activities that must be executed on time.

Gantt charts can be used in managing projects of all sizes and types.

These may include building infrastructures like dams, bridges, and highways. They may also include software development and other technologies. Project management tools, such as Microsoft Visio, Project, SharePoint, and Excel, or specialized software, such as Gantto or Matchware, can help in designing Gantt charts.

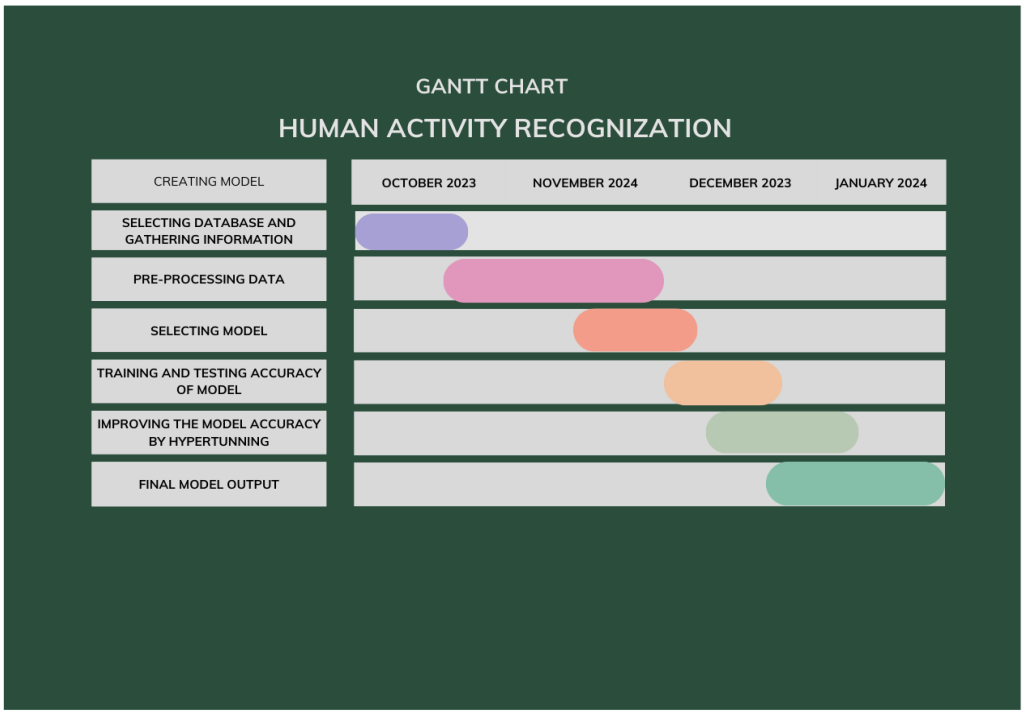
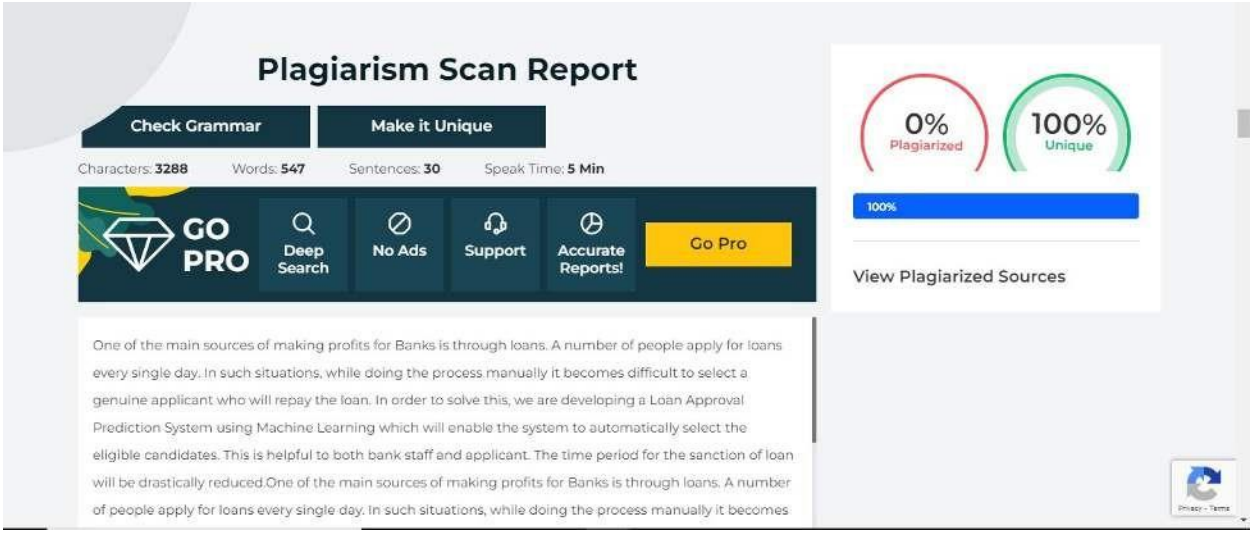
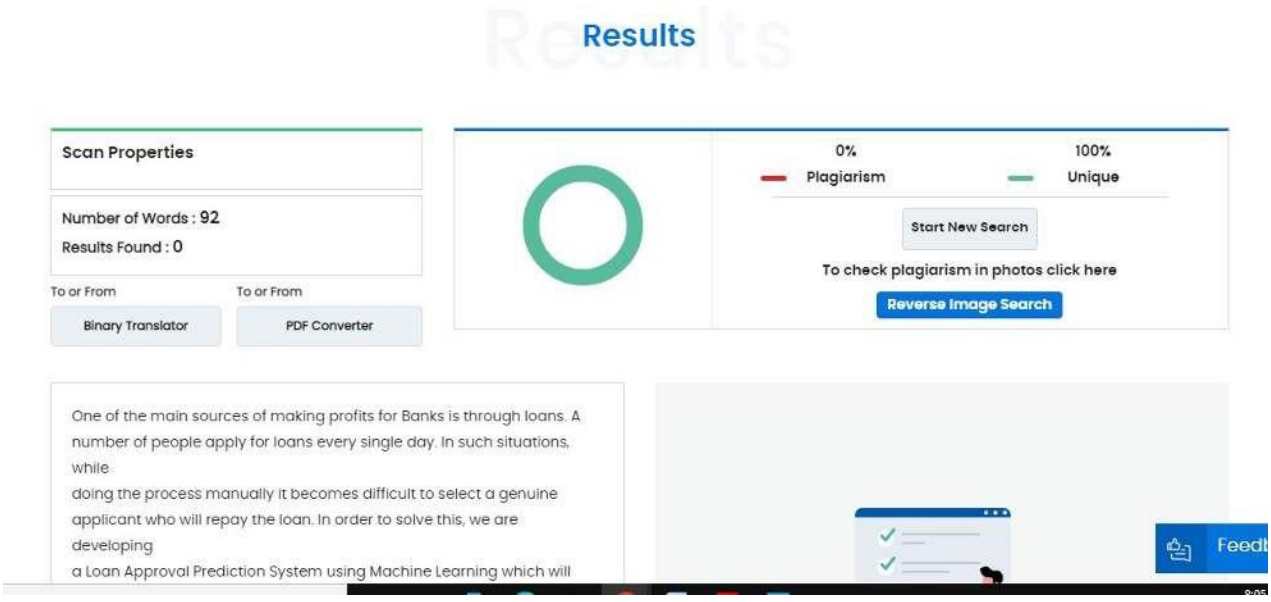


Figure 15 Gantt Chart

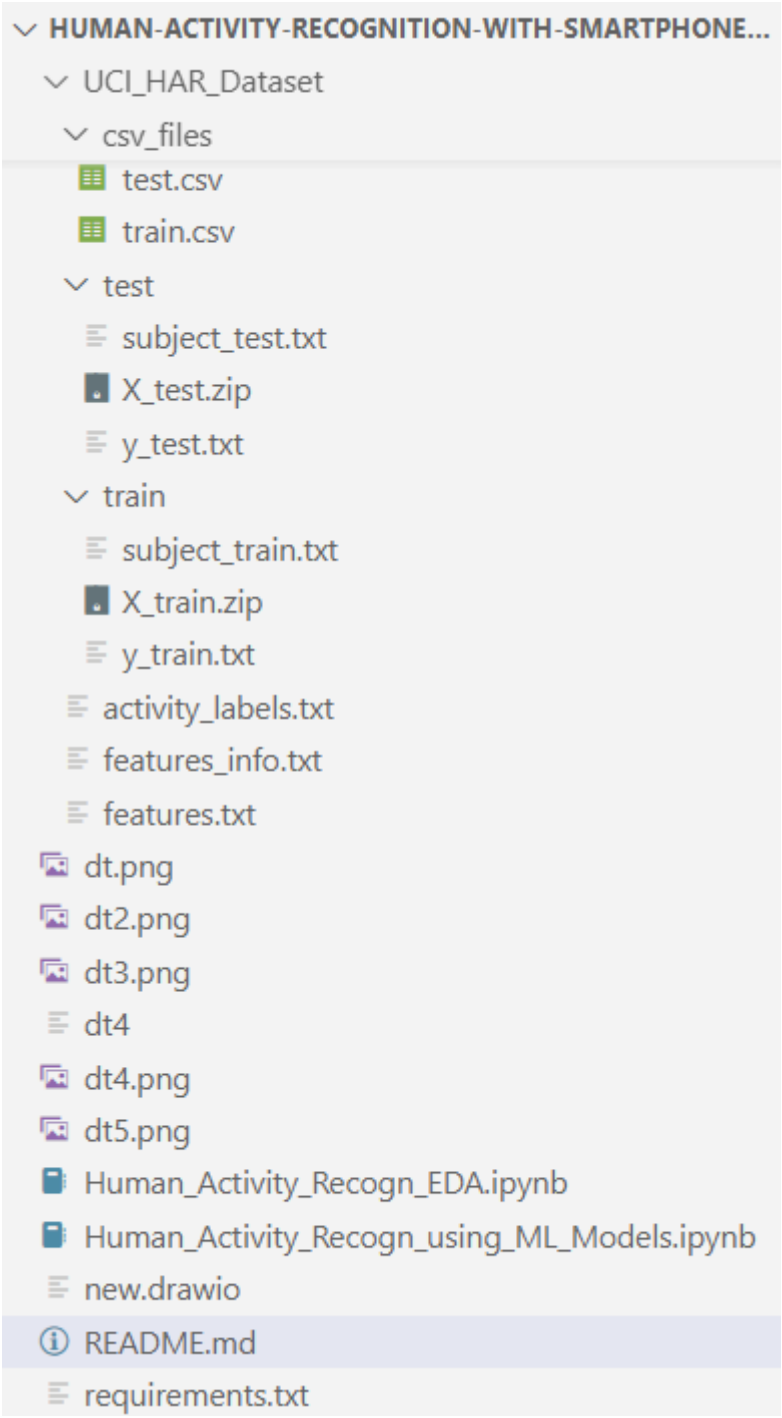
PLAGIARISM REPORT





Machine Learning Model

Jupyter Notebook: Home page containing all the files related to the project. Here, 'Human_Activity_Recogn_using_ML_Models.ipynb'is the file that contains the training and testing part of our project.



- **Human_Activity_Recogn_EDA.ipynb**

Exploring the Data

Let's add code to load necessary Python libraries and load data

```
# Importing necessary libraries

import numpy as np
import pandas as pd
```

Python

```
# Let's get the features from file features.txt

features = list()
with open("UCI_HAR_Dataset/features.txt") as f:
    features = [line.split()[1] for line in f.readlines()]

print("Number of Features: {}".format(len(features)))
```

Python

```
Number of Features: 561
```

```
features
```

Python

```
['tBodyAcc-mean()-X',
 'tBodyAcc-mean()-Y',
 'tBodyAcc-mean()-Z',
 'tBodyAcc-std()-X',
 'tBodyAcc-std()-Y',
```

Obtaining the training data

```
# Reading the train data from text file

X_train = pd.read_csv("UCI_HAR_Dataset/train/X_train.txt", delim_whitespace=True, header=None, names=features)
```

Python

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\parsers.py:678: UserWarning: Duplicate names specified. This will raise an error in the futu
  return _read(filepath_or_buffer, kwds)
```

```
# Displaying first 5 rows

X_train.head(5)
```

Python

n),gravityMean)	angle(tBodyGyroMean,gravityMean)	angle(tBodyGyroJerkMean,gravityMean)	angle(X,gravityMean)	angle(Y,gravityMean)	angle(Z,gravityMean)
0.030400	-0.464761	-0.018446	-0.841247	0.179941	-0.058627
-0.007435	-0.732626	0.703511	-0.844788	0.180289	-0.054317
0.177899	0.100699	0.808529	-0.848933	0.180637	-0.049118
-0.012892	0.640011	-0.485366	-0.848649	0.181935	-0.047663
0.122542	0.693578	-0.615971	-0.847865	0.185151	-0.043892

```
# Adding subject column to X_train from text file

X_train['Subject'] = pd.read_csv("UCI_HAR_Dataset/train/subject_train.txt", header=None, squeeze=True)
```

Python

```
# Displaying first 5 rows

X_train.head(5)
```

Python

	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	...	fBodyBodyGyroJerkMag-skewness()	fBody
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.298676	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.595051	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.390748	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.117290	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.351471	

5 rows × 562 columns

```
# Reading labels from text file

y_train = pd.read_csv("UCI_HAR_Dataset/train/y_train.txt", names=['Activity'], squeeze=True)
```

Python


```
# Reading labels from text file

y_train = pd.read_csv("UCI_HAR_Dataset/train/y_train.txt", names=['Activity'], squeeze=True)
```

Python


```
# Displaying first 5 rows

y_train.head(5)
```

Python

```
0    5
1    5
2    5
3    5
4    5
Name: Activity, dtype: int64
```

```
y_train_label = y_train.map({1:'Walking', 2:'Walking_Upstair', 3:'Walking_Downstair', \
                             4:'Sitting', 5:'Standing', 6:'Laying'})
```

Python

```
# Displaying first 5 rows

y_train_label.head(5)
```

Python

```
0    Standing
1    Standing
2    Standing
3    Standing
4    Standing
Name: Activity, dtype: object
```

```
# Putting all columns in a single Dataframe

train = X_train
train['Activity'] = y_train
train['Activity_Name'] = y_train_label
train.sample()
```

12] Python

	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	...	angle(tBodyAccMean,gravity)
1765	0.276214	-0.021644	-0.12604	-0.999125	-0.991823	-0.976067	-0.999571	-0.99091	-0.973345	-0.942342	...	-0.018172

1 rows × 564 columns

```
# Getting the shape of train dataframe

train.shape
```

13] Python

(7352, 564)

Obtaining the test data

```
# Reading the test data from a text file

X_test = pd.read_csv('UCI_HAR_Dataset/test/X_test.txt', delim_whitespace=True, header=None, names=features)
```

4] Python

```
# Displaying first 5 rows

X_test.head(5)
```

5] Python

	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	...	fBodyBodyGyroJerkMag-meanFreq()	fBody
0	0.257178	-0.023285	-0.014654	-0.938404	-0.920091	-0.667683	-0.952501	-0.925249	-0.674302	-0.894088	...	0.071645	
1	0.286027	-0.013163	-0.119083	-0.975415	-0.967458	-0.944958	-0.986799	-0.968401	-0.945823	-0.894088	...	-0.401189	
2	0.275485	-0.026050	-0.118152	-0.993819	-0.969926	-0.962748	-0.994403	-0.970735	-0.963483	-0.939260	...	0.062891	
3	0.270298	-0.032614	-0.117520	-0.994743	-0.973268	-0.967091	-0.995274	-0.974471	-0.968897	-0.938610	...	0.116695	
4	0.274833	-0.027848	-0.129527	-0.993852	-0.967445	-0.978295	-0.994111	-0.965953	-0.977346	-0.938610	...	-0.121711	

5 rows × 561 columns

```
# Adding subject column to test data

X_test['Subject'] = pd.read_csv('UCI_HAR_Dataset/test/subject_test.txt', header=None, squeeze=True)
```

5] Python

```
# Displaying first 5 rows

X_test.head(5)
```
[17] Python

	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	...	fBodyBodyGyroJerkMag-skewness()	fBody
0	0.257178	-0.023285	-0.014654	-0.938404	-0.920091	-0.667683	-0.952501	-0.925249	-0.674302	-0.894088	...	-0.330370	
1	0.286027	-0.013163	-0.119083	-0.975415	-0.967458	-0.944958	-0.986799	-0.968401	-0.945823	-0.894088	...	-0.121845	
2	0.275485	-0.026050	-0.118152	-0.993819	-0.969926	-0.962748	-0.994403	-0.970735	-0.963483	-0.939260	...	-0.190422	
3	0.270298	-0.032614	-0.117520	-0.994743	-0.973268	-0.967091	-0.995274	-0.974471	-0.968897	-0.938610	...	-0.344418	
4	0.274833	-0.027848	-0.129527	-0.993852	-0.967445	-0.978295	-0.994111	-0.965953	-0.977346	-0.938610	...	-0.534685	

5 rows × 562 columns

```
y_test = pd.read_csv('UCI_HAR_Dataset/test/y_test.txt', names=['Activity'], squeeze=True)
```
[18] Python

```
# Displaying first 5 rows

y_test.head(5)
```
[19] Python

```
0    5
1    5
2    5
3    5
4    5
Name: Activity, dtype: int64
```

```
# Mapping labels values to corresponding label text

y_test_label = y_train.map({1:'Walking', 2:'Walking_Upstair', 3:'Walking_Downstair', \
                            4:'Sitting', 5:'Standing', 6:'Laying'})
```
20] Python

```
# Displaying first 5 rows

y_test_label.head(5)
```
21] Python

```
0    Standing
1    Standing
2    Standing
3    Standing
4    Standing
Name: Activity, dtype: object
```

```
# Putting all columns together in a dataframe

test = X_test
test['Activity'] = y_test
test['Activity_Name'] = y_test_label
test.sample()
```
22] Python

	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	...	angle(tBodyAccMean,gravity)
791	0.044667	-0.100836	0.112583	-0.65924	-0.151193	0.242568	-0.654091	-0.052666	0.427958	-0.830019	...	0.151458

1 rows × 564 columns

```python
# Getting the shape of the test dataframe

test.shape
```

```
(2947, 564)
```

Data Cleaning

Checking train and test data for duplicate entries

```python
print('Number of duplicate entries in train data {}'.format(sum(train.duplicated())))
```

```
Number of duplicate entries in train data 0
```

```python
print('Number of duplicate entries in test data {}'.format(sum(test.duplicated())))
```

```
Number of duplicate entries in test data 0
```

Checking for Nan / Null values

```python
print('Number of NaN/Null values in train data {}'.format(train.isnull().values.sum()))
```

```
Number of NaN/Null values in train data 0
```

```python
print('Number of NaN/Null values in test data {}'.format(test.isnull().values.sum()))
```

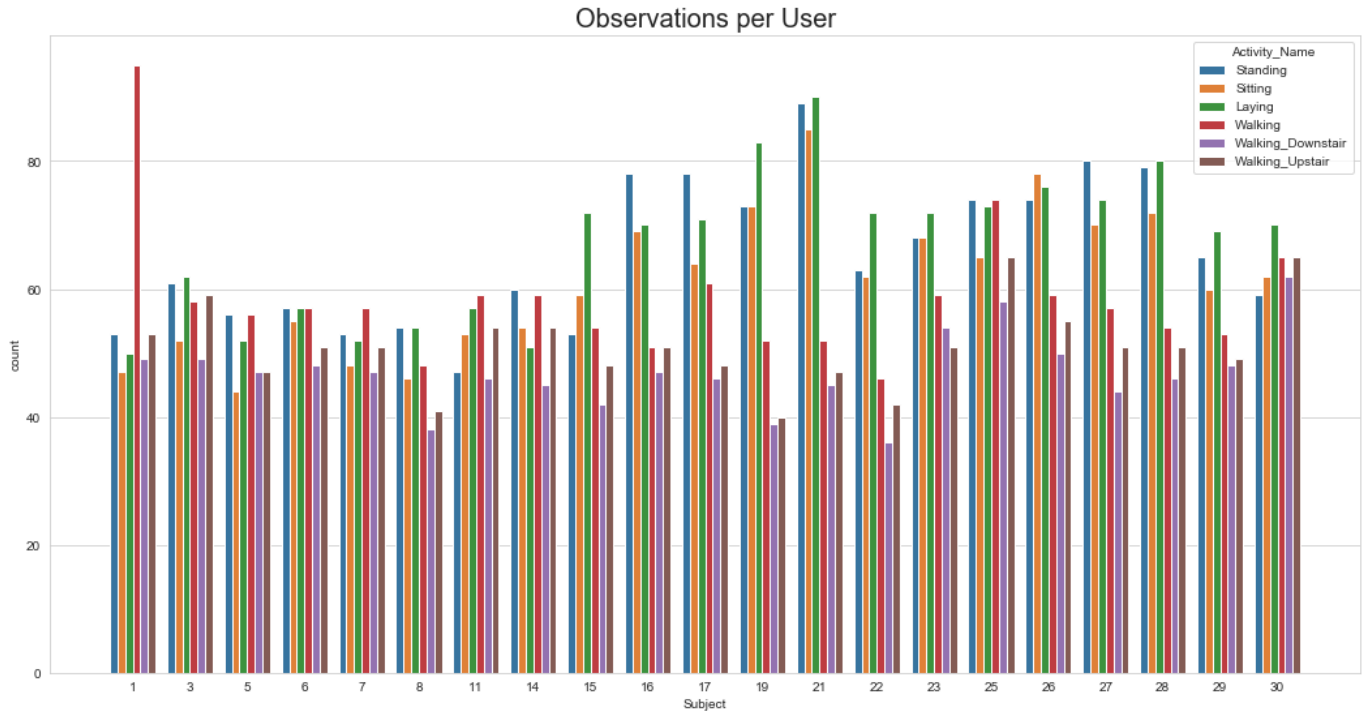
```
Number of NaN/Null values in test data 0
```

Checking for data imbalanced

```python
# Importing libraries

import matplotlib.pyplot as plt
import seaborn as sns
```

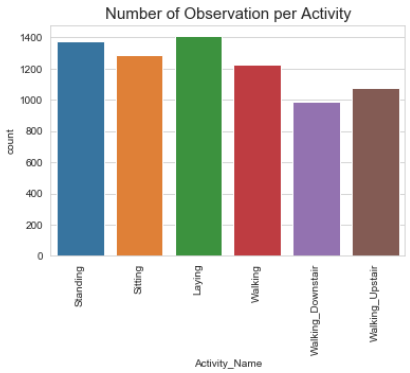
```python
# Plotting the data with respect to each individual OR Subject

sns.set_style('whitegrid')
plt.figure(figsize=(18,9))
plt.title('Observations per User', fontsize=20)
sns.countplot(x='Subject', hue='Activity_Name', data=train)
plt.plot()
```

```
[]
```




```python
# Let's check number of observations per label

plt.title('Number of Observation per Activity', fontsize=15)
sns.countplot(train.Activity_Name)
plt.xticks(rotation=90)
plt.show()
```

Python



Observation

Data seems to be balanced as all observations per subject is alomst the same

Changing the name of columns / features

```
# Copying train data column names to new column dataframe

columns = train.columns
```

31] Python

```
# Initial name of column

columns
```

32] Python

```
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',
       ...
       'angle(tBodyAccMean,gravity)', 'angle(tBodyAccJerkMean),gravityMean)',
       'angle(tBodyGyroMean,gravityMean)',
       'angle(tBodyGyroJerkMean,gravityMean)', 'angle(X,gravityMean)',
       'angle(Y,gravityMean)', 'angle(Z,gravityMean)', 'Subject', 'Activity',
       'Activity_Name'],
      dtype='object', length=564)
```

```
# Removing (), - and , from column name

columns = columns.str.replace('[()]', '')
columns = columns.str.replace('[-]', '')
columns = columns.str.replace('[,]', '')

train.columns = columns
test.columns = columns
```

33] Python

```
# Displaying new names

train.columns
```

34] Python

```
Index(['tBodyAccmeanX', 'tBodyAccmeanY', 'tBodyAccmeanZ', 'tBodyAccstdX',
       'tBodyAccstdY', 'tBodyAccstdZ', 'tBodyAccmadX', 'tBodyAccmadY',
       'tBodyAccmadZ', 'tBodyAccmaxX',
       ...
       'angletBodyAccMeangravity', 'angletBodyAccJerkMeangravityMean',
       'angletBodyGyroMeangravityMean', 'angletBodyGyroJerkMeangravityMean',
       'angleXgravityMean', 'angleYgravityMean', 'angleZgravityMean',
       'Subject', 'Activity', 'Activity_Name'],
      dtype='object', length=564)
```

Saving the train and test dataframe to a CSV file

```
train.to_csv('UCI_HAR_Dataset/csv_files/train.csv', index=False)
test.to_csv('UCI_HAR_Dataset/csv_files/test.csv', index=False)
```

Saving the train and test dataframe to a CSV file

```
[35]
train.to_csv('UCI_HAR_Dataset/csv_files/train.csv', index=False)
test.to_csv('UCI_HAR_Dataset/csv_files/test.csv', index=False)
```
Python

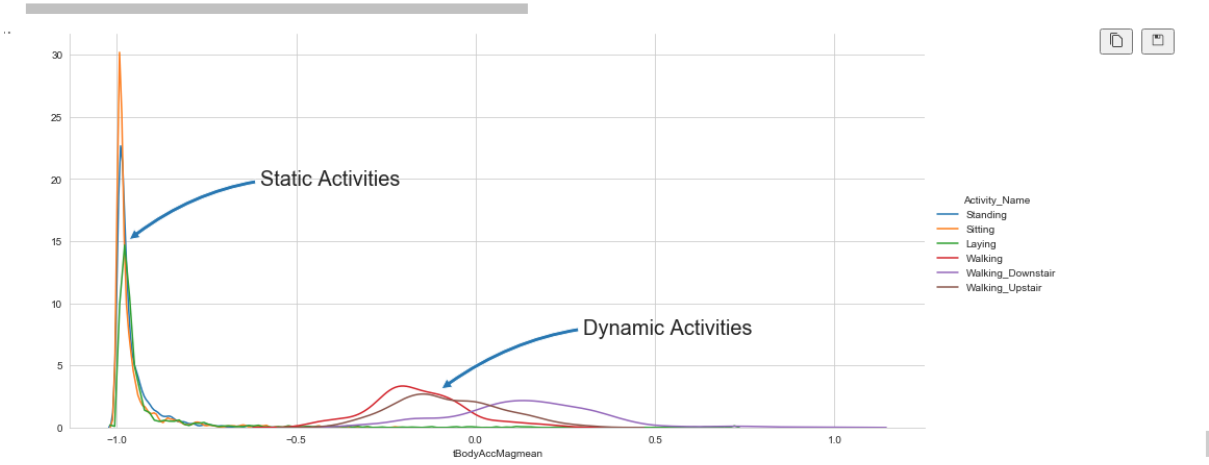
Exploratory Data Analysis

There are two types of activities in oue dataset

Static Activities - Sitting, Standing and Laying down
Dynamic Activities - Walking, Walking-upstairs, Walking-downstairs

```
[36]
# Let's see how we can differentiate the same using plots

facetgrid = sns.FacetGrid(data=train, hue='Activity_Name', size=6, aspect=2)
facetgrid.map(sns.distplot, 'tBodyAccMagmean', hist=False).add_legend()

plt.annotate("Static Activities", xy=(-0.97,15), xytext=(-0.6, 20), size=20,\
             va='center', ha='left',\
             arrowprops=dict(arrowstyle="simple",connectionstyle="arc3,rad=0.2"))

plt.annotate("Dynamic Activities", xy=(-0.1,3), xytext=(0.3, 8), size=20,\
             va='center', ha='left',\
             arrowprops=dict(arrowstyle="simple",connectionstyle="arc3,rad=0.2"))

plt.show()
```
Python

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; please update
  warnings.warn(msg, UserWarning)
```

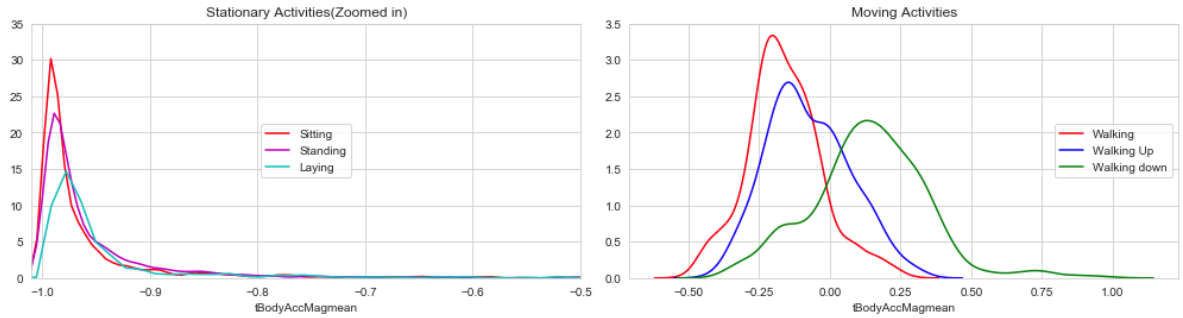



```
[37]
# Taking datapoint of each activity to a different Dataframe

df1 = train[train['Activity'] == 1]
df2 = train[train['Activity'] == 2]
df3 = train[train['Activity'] == 3]
df4 = train[train['Activity'] == 4]
df5 = train[train['Activity'] == 5]
df6 = train[train['Activity'] == 6]
```
Python

```
# Plotting each dataframe

plt.figure(figsize=(14,7))
plt.subplot(2,2,1)
plt.title('Stationary Activities(Zoomed in)')
sns.distplot(df4['tBodyAccMagmean'],color = 'r',hist = False, label = 'Sitting')
sns.distplot(df5['tBodyAccMagmean'],color = 'm',hist = False,label = 'Standing')
sns.distplot(df6['tBodyAccMagmean'],color = 'c',hist = False, label = 'Laying')
plt.axis([-1.01, -0.5, 0, 35])
plt.legend(loc='center')

plt.subplot(2,2,2)
plt.title('Moving Activities')
sns.distplot(df1['tBodyAccMagmean'],color = 'red',hist = False, label = 'Walking')
sns.distplot(df2['tBodyAccMagmean'],color = 'blue',hist = False,label = 'Walking Up')
sns.distplot(df3['tBodyAccMagmean'],color = 'green',hist = False, label = 'Walking down')
plt.legend(loc='center right')

plt.tight_layout()
plt.show()
```

8]

Python

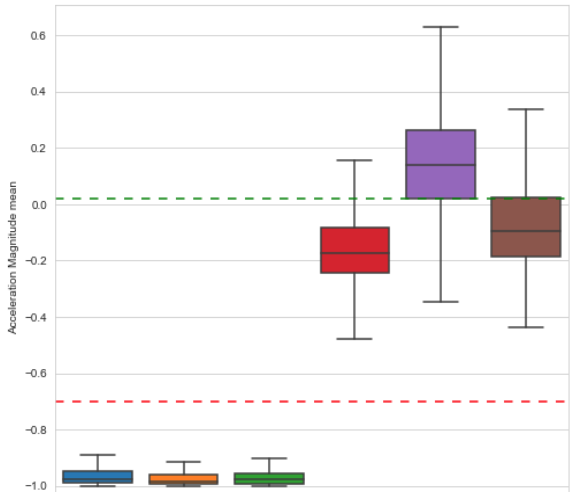


Magnitude of accelearation clearly separates static and dynamic activities

```
# Let's plot a boxplot to check the same

plt.figure(figsize=(8,8))
sns.boxplot(x='Activity_Name', y='tBodyAccMagmean',data=train, showfliers=False, saturation=1)
plt.ylabel('Acceleration Magnitude mean')
plt.xticks(rotation=90)
plt.axhline(y=-0.7, xmin=0.0, xmax=1.0, dashes=(5,5), c='r')
plt.axhline(y=0.02, xmin=0.0, xmax=1.0, dashes=(5,5), c='g')
plt.show()
```

9]

Python



Observations

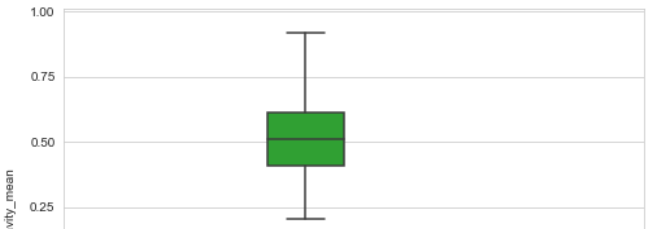
- If Acceleration Magnitude mean is greater than 0.8 then we can surely say that the activitites are either Walking, Walking Downstair OR Walking Upstair
- If Acceleration Magnitude mean is less than 0.8 then we can say that the activitites are either Standing, Sitting OR Laying
- If Acceleration Magnitude mean is greater than 0.02 then we can say that the activity is walking downstair
- Using this knowledge we can classify at least 60-70% of data. We can use if else statement to do the classification

Let's use Angle between X-axis and Gravity Mean

```
# Let's plot a boxplot to check the same

plt.figure(figsize=(8,8))
sns.boxplot(x='Activity_Name', y='angleXgravityMean',data=train, showfliers=False, saturation=1)
plt.ylabel('Angle between X-axis and Gravity_mean')
plt.xticks(rotation=90)
plt.axhline(y=0.0, xmin=0.0, xmax=1.0, dashes=(5,5), c='g')
plt.show()
```

]

Python



Observations

- If Angle between X-axis and Gravity_mean is greater than 0.0 then we can surely say that the activity is Laying

Let's apply t-SNE to better visualize the data

```
# Let's import necessary libraries

import numpy as np
import matplotlib.pyplot as plt

import seaborn as sns
from sklearn.manifold import TSNE
```

1]

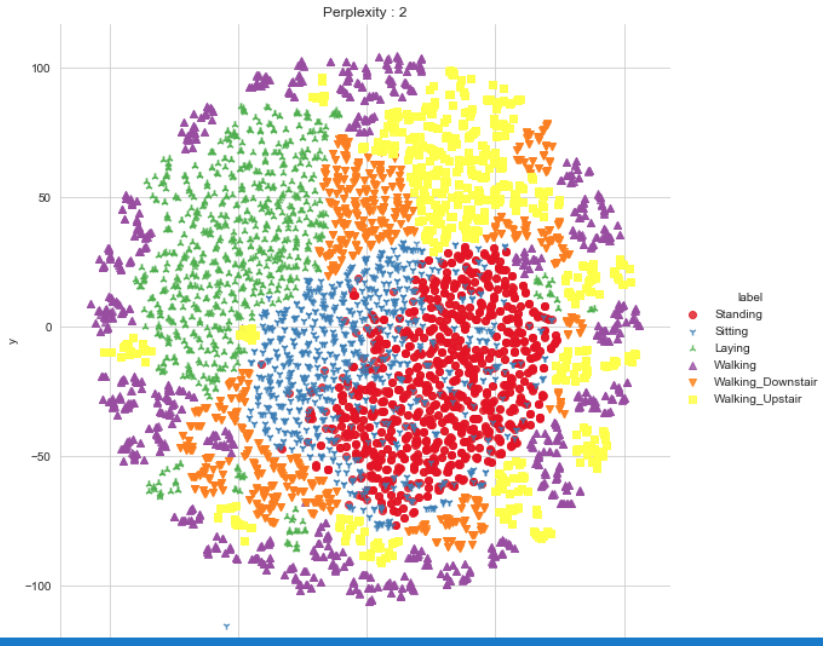
Python

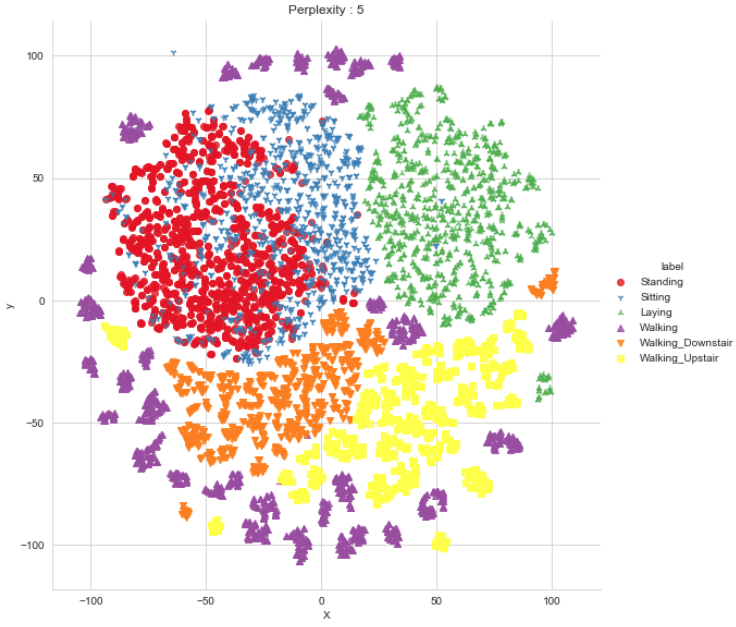
```
# Defining a function that takes input and plots the t-SNE

def perform_tsne(X_data, y_data, perplexities):
    for index, perplexity in enumerate(perplexities):
        # perform t-SNE
        X_reduced = TSNE(perplexity=perplexity).fit_transform(X_data)

        # Getting required data for Seaborn plot
        print('Creating plot for this t-SNE Visualization..')
        df = pd.DataFrame({'X':X_reduced[:,0], 'y':X_reduced[:,1], 'label':y_data})

        # Plotting the data
        sns.lmplot(data=df, x='X', y='y', hue='label', fit_reg=False, size=8,\
                   palette="Set1",markers=['o', '1','2','^','v','s'])
        plt.title("Perplexity : {}".format(perplexity))
        plt.show()
```

```
X_tsne = train.drop(['Subject', 'Activity','Activity_Name'], axis=1)
y_tsne = train['Activity_Name']
perform_tsne(X_data = X_tsne,y_data=y_tsne, perplexities =[2,5,10,20,50])
```

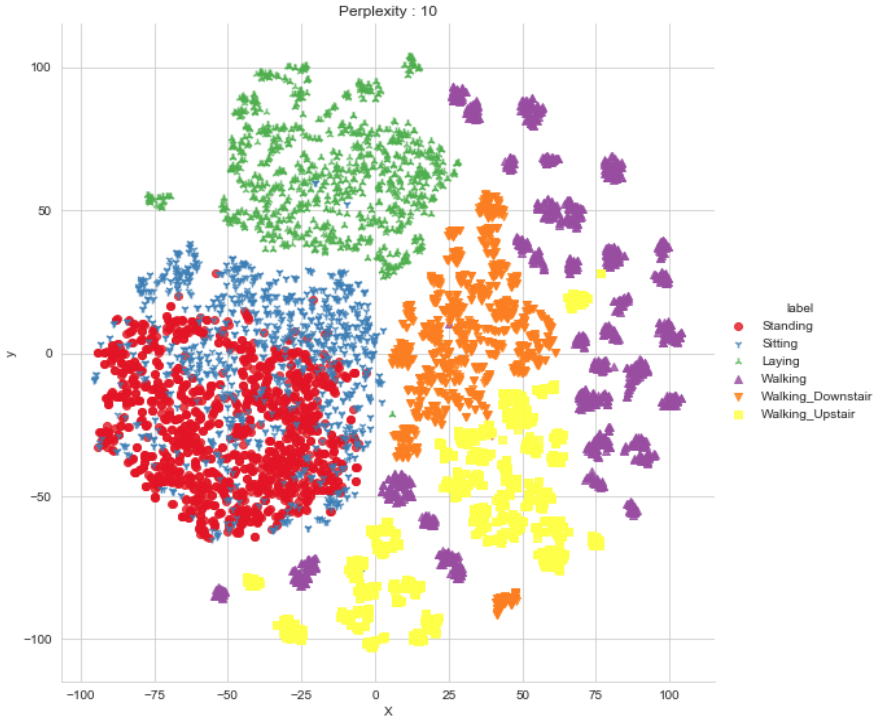
[43] Python

```
Creating plot for this t-SNE Visualization..
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\regression.py:546: UserWarning: The `size` paramter has been renamed to `height`; please updat
  warnings.warn(msg, UserWarning)
```

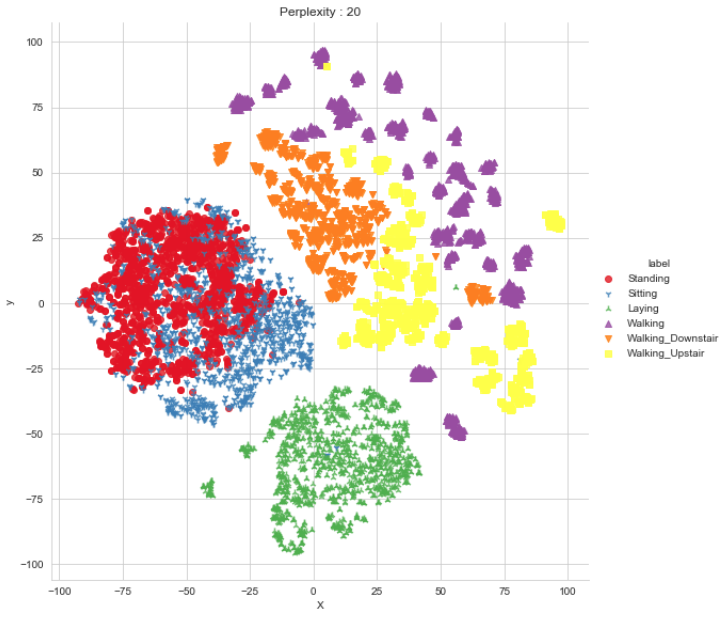




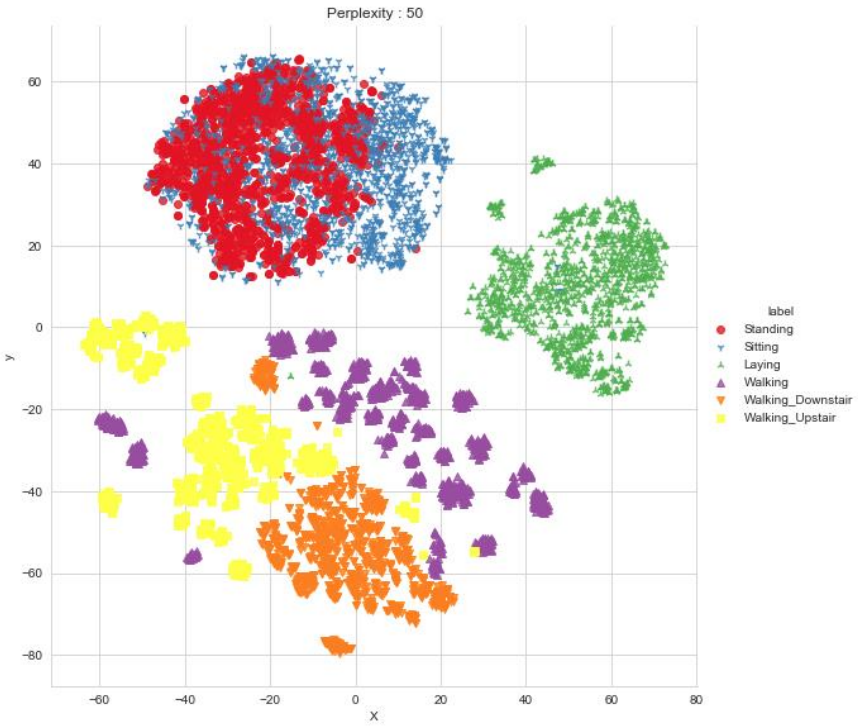

```
Done
Creating plot for this t-SNE Visualization..
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\regression.py:546: UserWarning: The `size` paramter has been renamed to `height`; please updat
  warnings.warn(msg, UserWarning)
```




```
Done
Creating plot for this t-SNE Visualization..
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\regression.py:546: UserWarning: The `size` paramter has been renamed to `height`; please updat
  warnings.warn(msg, UserWarning)
```




```
Done
Creating plot for this t-SNE Visualization..
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\regression.py:546: UserWarning: The `size` paramter has been renamed to `height`; please updat
  warnings.warn(msg, UserWarning)
```



- **Human_Activity_Recogn_using_ML_Models.ipynb**

Project :

Human Activity Recognition : Predictions using ML Models

```
# Importing necessary libraries

import numpy as np
import pandas as pd
```

1] ✓ 3.2s Python

```
C:\Users\Nabil Shah\AppData\Local\Temp\ipykernel_14804\1072870537.py:4: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

```
# Reading data from CSV file

train = pd.read_csv('./UCI_HAR_Dataset/csv_files/train.csv')
test = pd.read_csv('./UCI_HAR_Dataset/csv_files/test.csv')
```

2] ✓ 1.7s Python

```
# Checking the shape of train and test

print(train.shape)
print(test.shape)
```

3] ✓ 0.0s Python

```python
# Checking the shape of train and test

print(train.shape)
print(test.shape)
```

0.0s Python

```
(7352, 564)
(2947, 564)
```

```python
# Displaying first 5 rows of training data
train.head(5)
```

0.0s Python

	tBodyAccmeanX	tBodyAccmeanY	tBodyAccmeanZ	tBodyAccstdX	tBodyAccstdY	tBodyAccstdZ	tBodyAccmadX	tBodyAccmadY	tBodyAccmadZ	tBodyA
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-

5 rows × 564 columns

```python
# Displaying first 5 rows of testing data

test.head(5)
```

0.0s Python

odyGyroMeangravityMean	angletBodyGyroJerkMeangravityMean	angleXgravityMean	angleYgravityMean	angleZgravityMean	Subject	Activity	Activity_Name
-0.825886	0.271151	-0.720009	0.276801	-0.057978	2	5	Standing

```python
# Getting X_train and y_train from train data

X_train = train.drop(['Subject', 'Activity', 'Activity_Name'], axis=1)
y_train = train.Activity
```

0.0s Python

```python
test["Activity"].value_counts()
test["Activity_Name"].value_counts()
```

0.0s Python

```
Activity_Name
Walking             543
Standing            530
Laying              516
Sitting             489
Walking_Upstair     458
Walking_Downstair   411
Name: count, dtype: int64
```

```python
# Getting X_test and y_test from test data

X_test = test.drop(['Subject', 'Activity', 'Activity_Name'], axis=1)
y_test = test.Activity
```

0.0s Python

```python
# Displaying the shape of training and testing data

print('X_train and y_train : ({},{})'.format(X_train.shape, y_train.shape))
print('X_test  and y_test  : ({},{})'.format(X_test.shape, y_test.shape))
```

0.0s Python

```
# Displaying the shape of training and testing data

print('X_train and y_train : ({},{})'.format(X_train.shape, y_train.shape))
print('X_test  and y_test  : ({},{})'.format(X_test.shape, y_test.shape))
```

[15] ✓ 0.0s Python

```
X_train and y_train : ((7352, 561),(7352,))
X_test  and y_test  : ((2947, 561),(2947,))
```

```
# Let's use Linear discriminant analysis to find features that classifies the label well

# Importing libraries

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
```

[16] ✓ 4.5s Python

```
lda = LDA()
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
```

[17] ✓ 1.3s Python

```
# Displaying the shape of training and testing data

print('X_train and y_train : ({},{})'.format(X_train.shape, y_train.shape))
print('X_test  and y_test  : ({},{})'.format(X_test.shape, y_test.shape))
```

[18] ✓ 0.0s Python

```
X_train and y_train : ((7352, 5),(7352,))
X_test  and y_test  : ((2947, 5),(2947,))
```

Let's define some generic functions to create ML models

Function to plot Confusion Matrix

```
# Importing necessary libraries

import itertools
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

%matplotlib inline
```

[19] ✓ 1.3s Python

✓ # Creating a function to print text in Bold and in given color ···

```
# Function to plot Confusion Matrix

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion Matrix',
                          cmap = plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
```

```
from datetime import datetime
def perform_model(model, X_train, y_train, X_test, y_test, class_labels, cm_normalize=True, \
                  print_cm=True, cm_cmap=plt.cm.Greens):


    # Let's create an empty dictionary to be returned by the function
    results = dict()

    # Let's calculate & print the total training time

    train_start_time = datetime.now()
    model.fit(X_train, y_train)
    train_end_time = datetime.now()
    results['training_time'] =  train_end_time - train_start_time
    printmd('Training_time(HH:MM:SS.ms) - {}'.format(results['training_time']), color='blue')

    # Let's calculate & print the test time

    test_start_time = datetime.now()
    y_pred = model.predict(X_test)
    test_end_time = datetime.now()
    results['testing_time'] = test_end_time - test_start_time
    printmd('testing time(HH:MM:SS:ms) - {}'.format(results['testing_time']), color='blue')
    results['predicted'] = y_pred

    # Let's calculate the Accuracy of Model

    accuracy = metrics.accuracy_score(y_true=y_test, y_pred=y_pred)
    results['accuracy'] = accuracy
    printmd('**Accuracy:**', color='blue')
    print('{}'.format(accuracy))

    # Let's get the Confusion Matrix

    cm = metrics.confusion_matrix(y_test, y_pred)

    # Plotting Confusion Matrix
```

```
def print_grid_search_attributes(model):

    # Let's print the best estimator that gave highest score

    printmd('**Best Estimator:**', color='blue')
    print('{}\n'.format(model.best_estimator_))


    # Let's print the best parameters that gave best results

    printmd('**Best parameters:**', color='blue')
    print('{}\n'.format(model.best_params_))


    #  Let's print the number of cross validation splits

    printmd('**Number of CrossValidation sets:**', color='blue')
    print('{}\n'.format(model.n_splits_))


    # Let's print the Best score of the best estimator

    printmd('**Best Score:**', color='blue')
    print('{}\n'.format(model.best_score_))
```

✓ 0.0s Python

Applying various Machine learning model with Grid-Search

1. Logistic Regression

```
# Importing necessary libraries

from sklearn import linear_model
from sklearn import metrics

from sklearn.model_selection import GridSearchCV
```

[24] ✓ 0.0s Python

```
# Creating a list labels to be added to plots

labels=['Laying', 'Sitting','Standing','Walking','Walking_Downstairs','Walking_Upstairs']
```

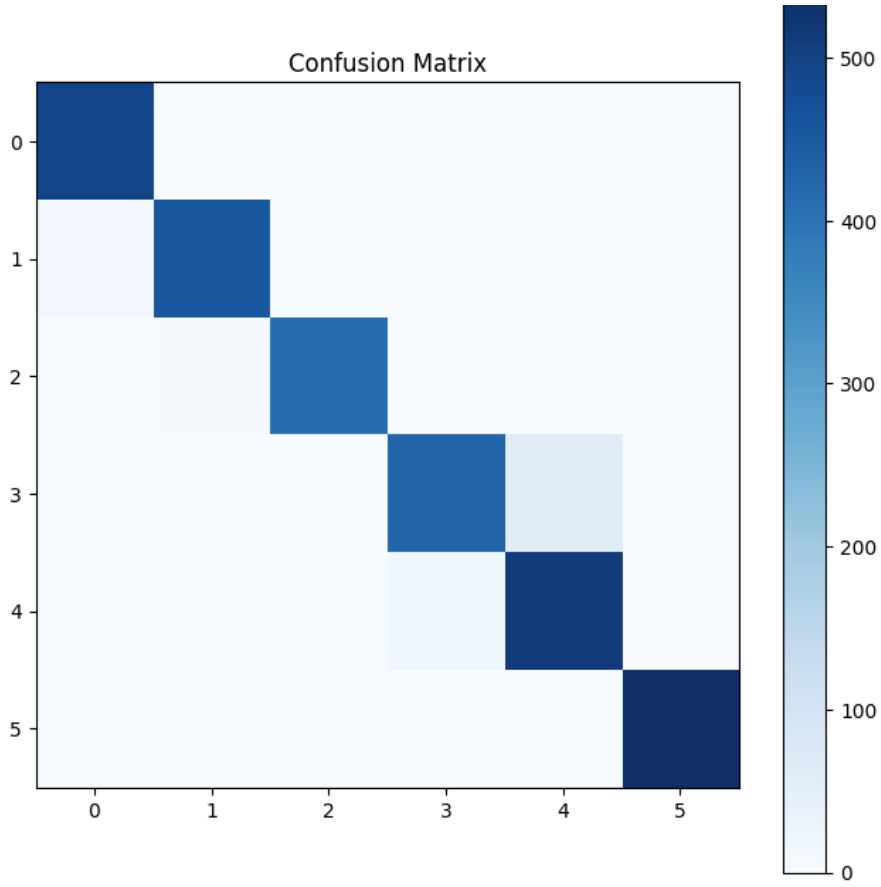
[70] Python

```
from sklearn.model_selection import GridSearchCV
from sklearn import linear_model
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt

# Let's define the parameters to be tuned
parameters = {'C': [20, 25, 30, 35, 40], 'penalty': ['l2','l1']}

# Let's initiate the model
log_reg = linear_model.LogisticRegression()
log_reg_grid = GridSearchCV(log_reg, param_grid=parameters, verbose=1, n_jobs=-1)
log_reg_grid_results = log_reg_grid.fit(X_train, y_train)
```

```
# Let's initiate the model
log_reg = linear_model.LogisticRegression()
log_reg_grid = GridSearchCV(log_reg, param_grid=parameters, verbose=1, n_jobs=-1)
log_reg_grid_results = log_reg_grid.fit(X_train, y_train)

# Printing the best attributes of the model
print("Best Parameters: ", log_reg_grid_results.best_params_)
print("Best Estimator: ", log_reg_grid_results.best_estimator_)
print("Best Score: ", log_reg_grid_results.best_score_)

# Perform predictions on the test set
y_pred = log_reg_grid_results.predict(X_test)

# Print classification report and confusion matrix
print("Classification Report:\n", classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# Plot confusion matrix
plt.figure(figsize=(8, 8))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
plt.show()
```

Pyt

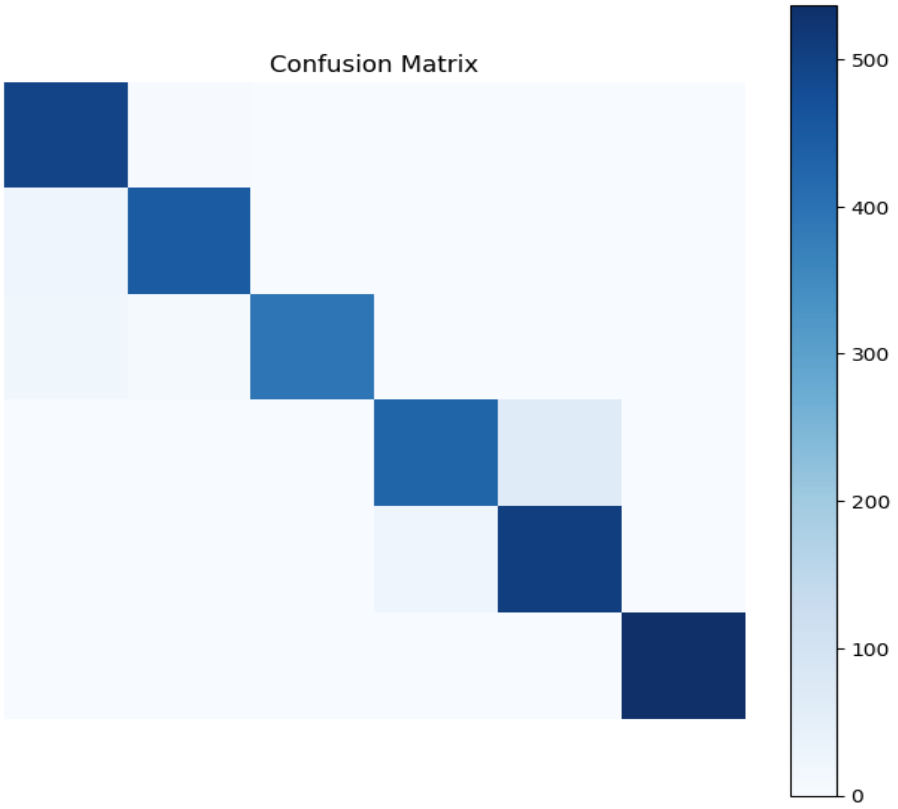
```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Best Parameters:  {'C': 20, 'penalty': 'l2'}
Best Estimator:  LogisticRegression(C=20)
Best Score:  0.9838137784005513
Classification Report:
              precision    recall  f1-score   support

           1       0.97      0.99      0.98       496
           2       0.97      0.97      0.97       471
           3       1.00      0.98      0.99       420
           4       0.96      0.88      0.91       491
```



2. Decision Tree

```
# Importing libraries

from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt

# Let's define the parameters to be tuned
parameters = {'max_depth': np.arange(4, 10, 1)}

# Let's initiate the model
dtree = DecisionTreeClassifier()
dtree_grid = GridSearchCV(dtree, param_grid=parameters, verbose=1, n_jobs=-1)
dtree_grid_results = dtree_grid.fit(X_train, y_train)

# Printing the best attributes of the model
print("Best Parameters: ", dtree_grid_results.best_params_)
print("Best Estimator: ", dtree_grid_results.best_estimator_)
print("Best Score: ", dtree_grid_results.best_score_)

# Perform predictions on the test set
y_pred = dtree_grid_results.predict(X_test)

# Print classification report and confusion matrix
print("Classification Report:\n", classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

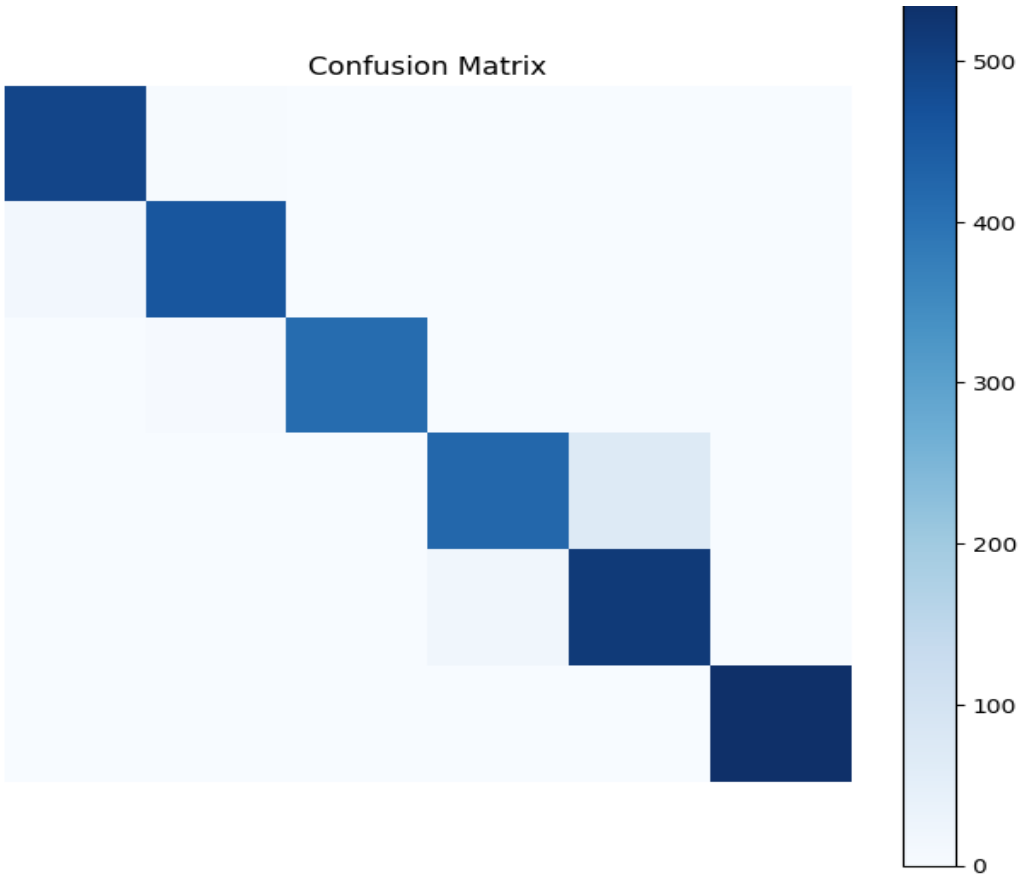


3. Random Forest Classifier

```
# Importing libraries

from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt

# Let's define the parameters to be tuned
parameters = {'n_estimators': np.arange(10, 201, 20), 'max_depth': np.arange(4, 15, 2)}

# Let's initiate the model
rfc = RandomForestClassifier()
rfc_grid = GridSearchCV(rfc, param_grid=parameters, n_jobs=-1)
rfc_grid_results = rfc_grid.fit(X_train, y_train)

# Printing the best attributes of the model
print("Best Parameters: ", rfc_grid_results.best_params_)
print("Best Estimator: ", rfc_grid_results.best_estimator_)
print("Best Score: ", rfc_grid_results.best_score_)

# Perform predictions on the test set
y_pred = rfc_grid_results.predict(X_test)

# Print classification report and confusion matrix
print("Classification Report:\n", classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Best Parameters:  {'max_depth': 8, 'n_estimators': 50}
Best Estimator:  RandomForestClassifier(max_depth=8, n_estimators=50)
Best Score:  0.9857187252875317
Classification Report:
              precision    recall  f1-score   support

           1       0.97      0.99      0.98       496
           2       0.98      0.97      0.98       471
           3       1.00      0.98      0.99       420
           4       0.96      0.86      0.91       491
           5       0.88      0.97      0.93       532
           6       1.00      1.00      1.00       537

    accuracy                           0.96      2947
   macro avg       0.97      0.96      0.96      2947
weighted avg       0.96      0.96      0.96      2947

Confusion Matrix:
 [[492   4   0   0   0   0]
 [ 14 457   0   0   0   0]
 [  3   4 412   1   0   0]
 [  0   1   0 422  68   0]
 [  0   0   0  15 517   0]
 [  0   0   0   0   0 537]]
```



Let's compare all the models together

```
# Logistic Regression
print('\n                    Accuracy     Error')
print('                  ----------   --------')
print('Logistic Regression : {:.04}%      {:.04}%'.format(log_reg_grid_results.best_score_ * 100,
                                                          100 - (log_reg_grid_results.best_score_ * 100)))
# Decision Tree
print('Decision Tree       : {:.04}%      {:.04}% '.format(dtree_grid_results.best_score_ * 100,
                                                           100 - (dtree_grid_results.best_score_ * 100)))

# Random Forest
print('Random Forest       : {:.04}%       {:.04}% '.format(rfc_grid_results.best_score_ * 100,
                                                            100 - (rfc_grid_results.best_score_ * 100)))
```

25] ✓ 0.0s Python

..

```
                    Accuracy     Error
                    ----------   --------
Logistic Regression : 98.38%      1.619%
Decision Tree       : 98.3%      1.7%
Random Forest       : 98.56%       1.442%
```

```
# Load the test dataset
test_data = pd.read_csv('./UCI_HAR_Dataset/csv_files/test.csv')

# Preprocess the test data (drop unnecessary columns and apply LDA transformation)
X_test = test_data.drop(['Subject', 'Activity', 'Activity_Name'], axis=1)
X_test_transformed = lda.transform(X_test)

# Predict output labels using Logistic Regression
log_reg_pred = log_reg_grid_results.predict(X_test_transformed)

# Predict output labels using Decision Tree
dtree_pred = dtree_grid_results.predict(X_test_transformed)

# Predict output labels using Random Forest
rfc_pred = rfc_grid_results.predict(X_test_transformed)

# Mapping activity labels to activity names
activity_mapping = {
    1: 'WALKING',
    2: 'WALKING_UPSTAIRS',
    3: 'WALKING_DOWNSTAIRS',
    4: 'SITTING',
    5: 'STANDING',
    6: 'LAYING'
}

# Transform predicted labels into activity names for Logistic Regression
log_reg_pred_activity = [activity_mapping[label] for label in log_reg_pred]

# Transform predicted labels into activity names for Decision Tree
dtree_pred_activity = [activity_mapping[label] for label in dtree_pred]

# Transform predicted labels into activity names for Random Forest
rfc_pred_activity = [activity_mapping[label] for label in rfc_pred]
```

```
# Now you have the predicted activity names for each model:
# log_reg_pred_activity, dtree_pred_activity, rfc_pred_activity
print("log" , log_reg_pred_activity)
print("dec" , dtree_pred_activity)
print("rfc",rfc_pred_activity)
```

[26] ✓ 0.4s Python

```
log ['STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STAN
dec ['STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STAN
rfc ['STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STANDING', 'STAN
```

Conclusion

The above table shows that Logistic Regression, Decision Tree & Random Forest has highest Accuracy with lowest Error value. We can use any of these three models for future predictions

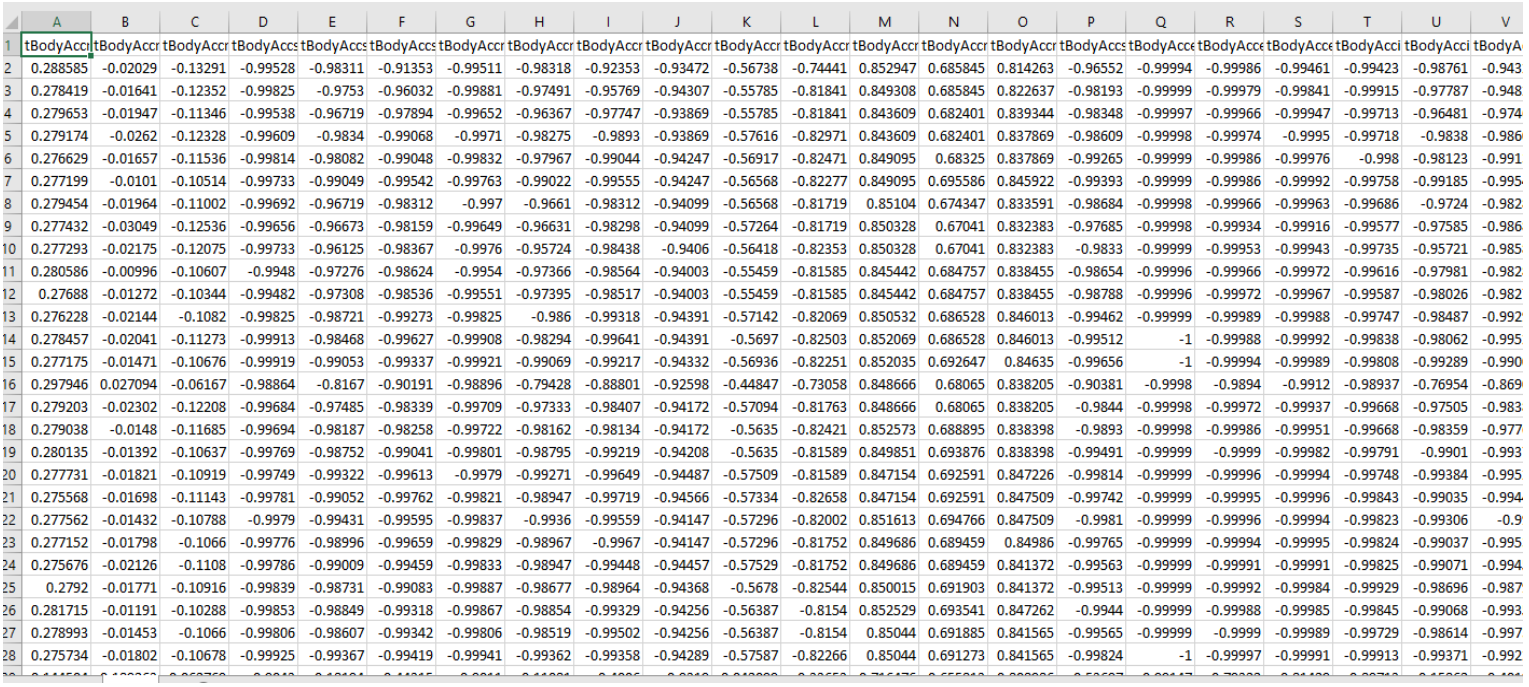
Requirement.txt

requirements.txt

```
argon2-cffi-bindings==21.2.0
arrow==1.2.3
asttokens==2.2.1
astunparse==1.6.3
attrs==22.2.0
backcall==0.2.0
beautifulsoup4==4.11.2
bleach==6.0.0
blinker==1.7.0
cachetools==5.3.2
certifi==2024.2.2
cffi==1.15.1
charset-normalizer==3.3.2
click==8.1.7
colorama==0.4.6
comm==0.1.2
contourpy==1.2.0
cycler==0.12.1
debugpy==1.6.6
decorator==5.1.1
defusedxml==0.7.1
executing==1.2.0
fastjsonschema==2.16.3
filelock==3.13.1
Flask==3.0.2
flatbuffers==23.5.26
fonttools==4.49.0
fqdn==1.5.1
fsspec==2024.2.0
gast==0.5.4
google-auth==2.28.1
google-auth-oauthlib==1.2.0
google-pasta==0.2.0
grpcio==1.62.0
h5py==3.10.0
idna==3.4
ipykernel==6.21.2
ipython==8.11.0
ipython-genutils==0.2.0
```

Datasets at a glance:

Train dataset:



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyA
0.288585	-0.02029	-0.13291	-0.99528	-0.98311	-0.91353	-0.99511	-0.98318	-0.92353	-0.93472	-0.56738	-0.74441	0.852947	0.685845	0.814263	-0.96552	-0.99994	-0.99986	-0.99461	-0.99423	-0.98761	-0.943
0.278419	-0.01641	-0.12352	-0.99825	-0.9753	-0.96032	-0.99881	-0.97491	-0.95769	-0.94307	-0.55785	-0.81841	0.849308	0.685845	0.822637	-0.98193	-0.99999	-0.99979	-0.99841	-0.99915	-0.97787	-0.948
0.279653	-0.01947	-0.11346	-0.99538	-0.96719	-0.97894	-0.99652	-0.96367	-0.97747	-0.93869	-0.55785	-0.81841	0.843609	0.682401	0.839344	-0.98348	-0.99997	-0.99966	-0.99947	-0.99713	-0.96481	-0.974
0.279174	-0.0262	-0.12328	-0.99609	-0.9834	-0.99068	-0.9971	-0.98275	-0.9893	-0.93869	-0.57616	-0.82971	0.843609	0.682401	0.837869	-0.98609	-0.99998	-0.99974	-0.9995	-0.99718	-0.9838	-0.986
0.276629	-0.01657	-0.11536	-0.99814	-0.98082	-0.99048	-0.99832	-0.97967	-0.99044	-0.94247	-0.56917	-0.82471	0.849095	0.68325	0.837869	-0.99265	-0.99999	-0.99986	-0.99976	-0.998	-0.98123	-0.991
0.277199	-0.0101	-0.10514	-0.99733	-0.99049	-0.99542	-0.99763	-0.99022	-0.99555	-0.94247	-0.56568	-0.82277	0.849095	0.695586	0.845922	-0.99393	-0.99999	-0.99986	-0.99992	-0.99758	-0.99185	-0.995
0.279454	-0.01964	-0.11002	-0.99692	-0.96719	-0.98312	-0.997	-0.9661	-0.98312	-0.94099	-0.56568	-0.81719	0.85104	0.674347	0.833591	-0.98684	-0.99998	-0.99966	-0.99963	-0.99686	-0.9724	-0.982
0.277432	-0.03049	-0.12536	-0.99656	-0.96673	-0.98159	-0.99649	-0.96631	-0.98298	-0.94099	-0.57264	-0.81719	0.850328	0.67041	0.832383	-0.97685	-0.99998	-0.99934	-0.99916	-0.99577	-0.97585	-0.986
0.277293	-0.02175	-0.12075	-0.99733	-0.96125	-0.98367	-0.9976	-0.95724	-0.98438	-0.9406	-0.56418	-0.82353	0.850328	0.67041	0.832383	-0.9833	-0.99999	-0.99953	-0.99943	-0.99735	-0.95721	-0.985
0.280586	-0.00996	-0.10607	-0.9948	-0.97276	-0.98624	-0.9954	-0.97366	-0.98564	-0.94003	-0.55459	-0.81585	0.845442	0.684757	0.838455	-0.98654	-0.99996	-0.99966	-0.99972	-0.99616	-0.97981	-0.982
0.27688	-0.01272	-0.10344	-0.99482	-0.97308	-0.98536	-0.99551	-0.97395	-0.98517	-0.94003	-0.55459	-0.81585	0.845442	0.684757	0.838455	-0.98788	-0.99996	-0.99972	-0.99967	-0.99587	-0.98026	-0.982
0.276228	-0.02144	-0.1082	-0.99825	-0.98721	-0.99273	-0.99825	-0.986	-0.99318	-0.94391	-0.57142	-0.82069	0.850532	0.686528	0.846013	-0.99462	-0.99999	-0.99989	-0.99988	-0.99747	-0.98487	-0.992
0.278457	-0.02041	-0.11273	-0.99913	-0.98468	-0.99627	-0.99908	-0.98294	-0.99641	-0.94391	-0.5697	-0.82503	0.852069	0.686528	0.846013	-0.99512	-1	-0.99988	-0.99992	-0.99838	-0.98062	-0.995
0.277175	-0.01471	-0.10676	-0.99919	-0.99053	-0.99337	-0.99921	-0.99069	-0.99217	-0.94332	-0.56936	-0.82251	0.852035	0.692647	0.84635	-0.99656	-1	-0.99994	-0.99989	-0.99808	-0.99289	-0.990
0.297946	0.027094	-0.06167	-0.98864	-0.8167	-0.90191	-0.98896	-0.79428	-0.88801	-0.92598	-0.44847	-0.73058	0.848666	0.68065	0.838205	-0.90381	-0.9998	-0.9894	-0.9912	-0.98937	-0.76954	-0.869
0.279203	-0.02302	-0.12208	-0.99684	-0.97485	-0.98339	-0.99709	-0.97333	-0.98407	-0.94172	-0.57094	-0.81763	0.848666	0.68065	0.838205	-0.9844	-0.99998	-0.99972	-0.99937	-0.99668	-0.97505	-0.983
0.279038	-0.0148	-0.11685	-0.99694	-0.98187	-0.98258	-0.99722	-0.98162	-0.98134	-0.94172	-0.5635	-0.82421	0.852573	0.688895	0.838398	-0.9893	-0.99998	-0.99986	-0.99951	-0.99668	-0.98359	-0.977
0.280135	-0.01392	-0.10637	-0.99769	-0.98752	-0.99041	-0.99801	-0.98795	-0.99219	-0.94208	-0.5635	-0.81589	0.849851	0.693876	0.838398	-0.99491	-0.99999	-0.9999	-0.99982	-0.99791	-0.9901	-0.993
0.277731	-0.01821	-0.10919	-0.99749	-0.99322	-0.99613	-0.9979	-0.99271	-0.99649	-0.94487	-0.57509	-0.81589	0.847154	0.692591	0.847226	-0.99814	-0.99999	-0.99996	-0.99994	-0.99748	-0.99384	-0.995
0.275568	-0.01698	-0.11143	-0.99781	-0.99052	-0.99762	-0.99821	-0.98947	-0.99719	-0.94566	-0.57334	-0.82658	0.847154	0.692591	0.847509	-0.99742	-0.99999	-0.99995	-0.99996	-0.99843	-0.99035	-0.994
0.277562	-0.01432	-0.10788	-0.9979	-0.99431	-0.99595	-0.99837	-0.9936	-0.99559	-0.94147	-0.57296	-0.82002	0.851613	0.694766	0.847509	-0.9981	-0.99999	-0.99996	-0.99994	-0.99823	-0.99306	-0.9
0.277152	-0.01798	-0.1066	-0.99776	-0.98996	-0.99659	-0.99829	-0.98967	-0.9967	-0.94147	-0.57296	-0.81752	0.849686	0.689459	0.84986	-0.99765	-0.99999	-0.99994	-0.99995	-0.99824	-0.99037	-0.995
0.275676	-0.02126	-0.1108	-0.99786	-0.99009	-0.99459	-0.99833	-0.98947	-0.99448	-0.94457	-0.57529	-0.81752	0.849686	0.689459	0.841372	-0.99563	-0.99999	-0.99991	-0.99991	-0.99825	-0.99071	-0.994
0.2792	-0.01771	-0.10916	-0.99839	-0.98731	-0.99083	-0.99887	-0.98677	-0.98964	-0.94368	-0.5678	-0.82544	0.850015	0.691903	0.841372	-0.99513	-0.99999	-0.99992	-0.99984	-0.99929	-0.98696	-0.987
0.281715	-0.01191	-0.10288	-0.99853	-0.98849	-0.99318	-0.99867	-0.98854	-0.99329	-0.94256	-0.56387	-0.8154	0.852529	0.693541	0.847262	-0.9944	-0.99999	-0.99988	-0.99985	-0.99845	-0.99068	-0.993
0.278993	-0.01453	-0.1066	-0.99806	-0.98607	-0.99342	-0.99806	-0.98519	-0.99502	-0.94256	-0.56387	-0.8154	0.85044	0.691885	0.841565	-0.99565	-0.99999	-0.9999	-0.99989	-0.99729	-0.98614	-0.997
0.275734	-0.01802	-0.10678	-0.99925	-0.99367	-0.99419	-0.99941	-0.99362	-0.99358	-0.94289	-0.57587	-0.82266	0.85044	0.691273	0.841565	-0.99824	-1	-0.99997	-0.99991	-0.99913	-0.99371	-0.992

Test dataset:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyAcc	tBodyA
0.257178	-0.02329	-0.01465	-0.9384	-0.92009	-0.66768	-0.9525	-0.92525	-0.6743	-0.89409	-0.55458	-0.46622	0.717208	0.635502	0.789497	-0.87776	-0.99777	-0.99841	-0.93435	-0.97567	-0.94982	-0.830
0.286027	-0.01316	-0.11908	-0.97541	-0.96746	-0.94496	-0.9868	-0.9684	-0.94582	-0.89409	-0.55458	-0.80601	0.768031	0.683698	0.796706	-0.9691	-0.99958	-0.99965	-0.99772	-0.99401	-0.97364	-0.950
0.275485	-0.02605	-0.11815	-0.99382	-0.96993	-0.96275	-0.9944	-0.97073	-0.96348	-0.93926	-0.56851	-0.79912	0.848305	0.667864	0.822442	-0.97678	-0.99995	-0.99957	-0.99872	-0.99346	-0.97427	-0.964
0.270298	-0.03261	-0.11752	-0.99474	-0.97327	-0.96709	-0.99527	-0.97447	-0.9689	-0.93861	-0.56851	-0.79912	0.848305	0.667864	0.822442	-0.97442	-0.99995	-0.99933	-0.99893	-0.99524	-0.97874	-0.969
0.274833	-0.02785	-0.12953	-0.99385	-0.96745	-0.97829	-0.99411	-0.96595	-0.97735	-0.93861	-0.56083	-0.82589	0.849179	0.6707	0.829897	-0.97528	-0.99995	-0.99947	-0.99882	-0.9935	-0.96703	-0.976
0.27922	-0.01862	-0.1139	-0.99446	-0.97042	-0.96532	-0.99459	-0.96948	-0.9659	-0.93786	-0.56083	-0.80115	0.850985	0.67559	0.829897	-0.98098	-0.99996	-0.99972	-0.99893	-0.99486	-0.97634	-0.970
0.279746	-0.01827	-0.104	-0.99582	-0.97635	-0.97772	-0.996	-0.97366	-0.97925	-0.93786	-0.56426	-0.80115	0.851515	0.688119	0.833937	-0.98685	-0.99997	-0.9998	-0.99944	-0.99584	-0.97424	-0.981
0.274601	-0.02504	-0.11683	-0.99559	-0.98207	-0.98526	-0.99534	-0.98148	-0.98461	-0.94126	-0.56887	-0.8213	0.851096	0.686588	0.833937	-0.98566	-0.99997	-0.99976	-0.99959	-0.99464	-0.98362	-0.983
0.272529	-0.02095	-0.11447	-0.99678	-0.97591	-0.9866	-0.99703	-0.97374	-0.98556	-0.94126	-0.56429	-0.8213	0.849753	0.686588	0.840875	-0.98739	-0.99998	-0.99977	-0.99968	-0.99696	-0.97293	-0.982
0.275746	-0.01037	-0.09978	-0.99837	-0.98693	-0.99102	-0.99866	-0.98714	-0.99108	-0.94376	-0.56429	-0.81433	0.849753	0.68923	0.847376	-0.9911	-0.99999	-0.99983	-0.99973	-0.9986	-0.98981	-0.989
0.278596	-0.01523	-0.09891	-0.99879	-0.98194	-0.99138	-0.99883	-0.98001	-0.99141	-0.94376	-0.56702	-0.81433	0.852843	0.689106	0.849271	-0.99199	-0.99999	-0.99986	-0.99972	-0.99856	-0.97976	-0.989
0.279152	-0.02188	-0.10973	-0.99778	-0.99295	-0.98568	-0.99771	-0.99268	-0.98494	-0.94398	-0.57706	-0.81988	0.852317	0.689106	0.840358	-0.99307	-0.99999	-0.99992	-0.99971	-0.99683	-0.99331	-0.98
0.274544	-0.02315	-0.11254	-0.9962	-0.99157	-0.98752	-0.99652	-0.99206	-0.98713	-0.94129	-0.57706	-0.82039	0.847464	0.685826	0.840358	-0.99154	-0.99998	-0.99989	-0.99973	-0.99624	-0.99344	-0.988
0.269066	-0.02769	-0.11018	-0.99688	-0.98644	-0.98848	-0.9975	-0.98739	-0.98949	-0.94129	-0.57905	-0.8223	0.845002	0.673562	0.833509	-0.98802	-0.99997	-0.99971	-0.99978	-0.99725	-0.99046	-0.988
0.275579	-0.01894	-0.09741	-0.99606	-0.96823	-0.9807	-0.99622	-0.96463	-0.98235	-0.94325	-0.56155	-0.8134	0.845002	0.673562	0.833509	-0.98322	-0.99998	-0.99968	-0.99939	-0.99592	-0.96604	-0.982
0.281931	-0.00488	-0.08611	-0.98908	-0.95901	-0.97302	-0.99378	-0.96795	-0.97785	-0.9205	-0.53516	-0.78128	0.83017	0.66759	0.842335	-0.97145	-0.99989	-0.99925	-0.99857	-0.9976	-0.97968	-0.984
0.311078	-0.01943	-0.10187	-0.93669	-0.84019	-0.81683	-0.94134	-0.84883	-0.81261	-0.84158	-0.45723	-0.67024	0.834126	0.64636	0.796272	-0.87854	-0.99751	-0.99453	-0.98273	-0.95614	-0.90024	-0.829
0.262328	-0.02326	-0.12552	-0.98456	-0.91338	-0.91267	-0.98395	-0.9072	-0.90043	-0.93853	-0.55027	-0.7973	0.834126	0.64636	0.796272	-0.93989	-0.99977	-0.99818	-0.99499	-0.9829	-0.91894	-0.874
0.288416	-0.00349	-0.08383	-0.99457	-0.97836	-0.97994	-0.99534	-0.97673	-0.9778	-0.93828	-0.55027	-0.7973	0.847618	0.695902	0.852497	-0.97341	-0.99994	-0.99949	-0.99866	-0.99583	-0.98009	-0.97
0.271166	-0.02597	-0.09492	-0.97012	-0.90188	-0.9653	-0.97739	-0.90897	-0.96839	-0.91115	-0.54791	-0.78694	0.781	0.60356	0.828882	-0.95805	-0.99942	-0.99766	-0.99872	-0.99132	-0.94324	-0.975
0.253095	-0.04419	-0.1393	-0.96892	-0.91038	-0.91886	-0.97539	-0.91803	-0.90989	-0.91115	-0.54791	-0.78694	0.781	0.60356	0.80832	-0.92546	-0.99927	-0.99681	-0.99455	-0.98487	-0.94261	-0.902
0.271339	-0.02995	-0.1186	-0.96093	-0.90504	-0.87127	-0.9734	-0.9057	-0.85633	-0.86437	-0.51815	-0.73775	0.760479	0.644166	0.806879	-0.91498	-0.99907	-0.99763	-0.99079	-0.98643	-0.93059	-0.831
0.299267	0.011799	-0.03165	-0.94875	-0.84418	-0.88071	-0.95841	-0.85373	-0.88484	-0.8384	-0.41586	-0.68033	0.760479	0.641519	0.806879	-0.88019	-0.99838	-0.99329	-0.98343	-0.96597	-0.88048	-0.903
0.293897	0.011151	-0.06928	-0.96241	-0.86334	-0.85996	-0.97218	-0.86901	-0.85615	-0.8384	-0.41586	-0.68033	0.820955	0.641519	0.823948	-0.91046	-0.99908	-0.99449	-0.98722	-0.98484	-0.90225	-0.856
0.275639	-0.01557	-0.12546	-0.98916	-0.96852	-0.96872	-0.99063	-0.96944	-0.96776	-0.93051	-0.54475	-0.81198	0.83551	0.682499	0.823948	-0.97664	-0.99989	-0.99969	-0.99869	-0.99309	-0.977	-0.969
0.272255	-0.02505	-0.13318	-0.99115	-0.96355	-0.97316	-0.9918	-0.96258	-0.97089	-0.93443	-0.55771	-0.82518	0.842611	0.675445	0.825676	-0.97181	-0.99992	-0.9995	-0.99839	-0.99289	-0.96864	-0.968
0.276427	-0.02628	-0.12694	-0.99299	-0.96562	-0.96574	-0.99342	-0.96446	-0.96218	-0.93443	-0.56845	-0.81845	0.84716	0.675445	0.825676	-0.97292	-0.99995	-0.99949	-0.99849	-0.99337	-0.96845	-0.955

Future Enhancements

•The Human Activity Recognition model could benefit from integration with Cloud services to facilitate scalability and efficiency in an Artificial Intelligence environment.

•Consideration for designing a visually appealing Graphical User Interface (GUI) could enhance user interaction and understanding of the Human Activity Recognition system.

•Additional functionalities within the GUI could enhance user engagement and experience by providing real-time feedback and personalized insights into activity patterns.

•Improving model training by leveraging a comprehensive dataset encompassing diverse activity types, sensor readings, and environmental factors could enhance the accuracy and robustness of the Human Activity Recognition system.

•Implementing a feedback mechanism within the system could offer insights into misclassifications or inaccuracies, allowing for continual improvement and refinement of the Human Activity Recognition model.

Conclusion

Human activity recognition (HAR) using smartphone datasets represents a significant advancement in the field of wearable computing and artificial intelligence. The extensive evaluation of the HAR system's strengths and weaknesses underscores its transformative potential in optimizing time and resources for both users and researchers.

Efficient Resource Utilization: By leveraging smartphone sensors to capture data related to human activities, the HAR system maximizes resource utilization. Unlike traditional methods that may require specialized equipment or manual observation, the system harnesses the ubiquitous nature of smartphones, making it accessible to a wide range of users.

Industry Expectations and Requirements: The HAR system not only meets but exceeds industry expectations by providing accurate and reliable activity recognition capabilities. Its ability to classify various activities with high precision and recall ensures that it fulfills fundamental requirements within the realm of activity recognition, thereby enhancing its value proposition for industries ranging from healthcare to sports and fitness.

Seamless Integration: The system's seamless integration capabilities with various platforms enhance its adaptability and utility. Whether integrated into mobile applications, wearable devices, or smart environments, the HAR system seamlessly fits into users' existing technological ecosystems, ensuring a frictionless user experience.

Error Minimization: By automating the activity recognition process, the system minimizes the potential for errors inherent in manual classification methods. Human observers may be prone to inaccuracies or subjective biases, whereas the HAR system relies on objective sensor data and machine learning algorithms to consistently deliver precise results.

Efficiency and User Satisfaction: The HAR system's commendable efficiency is evident in its ability to accurately recognize activities in real-time or near-real-time scenarios. Users benefit from prompt feedback and actionable insights derived from the system's outputs, leading to increased satisfaction and engagement with the technology.

Achieving Objectives: The HAR system effectively addresses the initial objectives set forth in the domain of activity recognition, demonstrating its efficacy in enhancing human-machine interaction, facilitating personalized services, and supporting decision-making processes in various domains.

In summary, human activity recognition using smartphone datasets represents a paradigm shift in how we perceive and interact with technology in our daily lives. By harnessing the power of sensor data and machine learning algorithms, the HAR system offers tangible benefits in terms of efficiency, accuracy, and user satisfaction, ultimately contributing to a more seamless and intelligent technological ecosystem.

Bibliography

1. V. Vankadara, B. S. K. K. Ibrahim, and M. R. Nookala, "Human Activity Recognition Using Smartphone Dataset," International Journal of Computer Applications, vol. 181, no. 19, pp. 33-38, 2018. (Introduction)

2. Banerjee, S. Sengupta, and S. Chakraborty, "A Review on Human Activity Recognition Using Smartphone Dataset," International Journal of Computer Applications, vol. 175, no. 10, pp. 32-38, 2017. (Methodology & Implementation)

3. J. Chen, X. Li, and H. Liu, "Human Activity Recognition Using Machine Learning Techniques on Smartphone Dataset," in Proceedings of the International Conference on Data Mining and Big Data, 2019, pp. 231-236. (Implementation)

4. S. Singh, R. K. Sharma, and S. R. Chaudhary, "Machine Learning-Based Human Activity Recognition Using Smartphone Dataset," in Proceedings of the International Conference on Artificial Intelligence and Big Data, 2020, pp. 143-150. (Introduction & EDA)

5. S. Gupta, A. Kumar, and S. Verma, "Human Activity Recognition Through Smartphone Dataset Using Machine Learning Algorithms," International Journal of Computer Sciences and Engineering, vol. 7, no. 10, pp. 64-69, 2019. (Insightful for project)

6. https://alex.smola.org/drafts/thebook.pdf (Implementation Techniques)

7. https://aitskadapa.ac.in/ebooks/AI&ML/MACHINE%20LEARNING/Machine%20Learning%20(%20etc.)%20(z-lib.org).pdf (Introduction)