International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 07 | July 2024 www.irjet.net p-ISSN: 2395-0072

A Comparative Study on Human Activity Recognition Using Smartphone Dataset through Machine Learning Approach

Yashraj Mishra[1], Ankita Jaiswal[2], Dr. Goldi Soni[3]

[1]Student, Amity University Chhattisgarh
[2]Student, Amity University Chhattisgarh
[3]Assistant Professor, Amity University Chhattisgarh

Abstract - *Human Activity Recognition (HAR) using smartphone data leverages built-in sensors to detect and classify users' activities. The proliferation of smartphones with various sensors has opened new avenues for recognizing human activities, with significant noteworthy applications in remote healthcare and activity tracking for the elderly or disabled, and fitness monitoring. Recent studies focus on applications that identify daily track activities and calculate calories burned in real-time. These applications capture labeled triaxial acceleration data from the smartphone's accelerometer, which is then preprocessed and analyzed. Features are extracted using various methods, and machine learning algorithms classify the activities. The most effective models are integrated into smartphone applications, enabling real-time activity recognition and health monitoring. This project contributes to developing assistive technologies that improve quality of life and promote a healthier lifestyle through classification techniques. Six types of human activities: standing, sitting, lying down, walking, walking upstairs, and walking downstairs —were identified using data from the University of California Machine Learning Repository. Data from the Samsung Galaxy S II's gyroscope and accelerometer were divided into training and testing sets in a 70:30 ratio. Principal Component Analysis reduced dimensionality, and machine learning methods such as Random Forest, Support Vector Machine, Artificial Neural Network, and K-Nearest Neighbor classified activities. Random simulation and confusion matrices compared the accuracy and performance of various models.*

Key words: Exploratory Data Analysis, Machine Learning Algorithms, Data Modelling, Hyperparameter and cross validation.

1.INTRODUCTION

Human Activity Recognition (HAR) using smartphone data is an emerging field that leverages machine learning algorithms to classify various physical activities based on information collected from a smartphone's sensors. This study involved 28 individuals aged between 19 and 48, who participated in a series of experiments. Each participant had a smartphone (Samsung Galaxy S II) attached to their waist while carrying out six different activities: walking, walking upstairs, walking downstairs, sitting, standing, and lying down.

1.1 Background

The smartphone's integrated accelerometer and gyroscope recorded 3-axial angular velocity and 3-axial linear acceleration at a steady rate of 50Hz. These recordings were essential for capturing the dynamic movements associated with each activity. To ensure accuracy in labelling the data, the trials were captured on camera, enabling manual verification of the activities performed.

The generated dataset was divided randomly into two subsets: training data consisting of 70% of the volunteers and test data from the remaining 30%. This division ensured that the machine learning models could be trained and validated effectively.

1.1.1 Pre-processing and Feature Extraction

Raw accelerometer and gyroscope signals underwent pre-processing to enhance data quality. Noise filters removed irrelevant data, and signals were sampled using 2.56-second sliding windows with 50% overlap, yielding 128 readings per window. A Butterworth low-pass filter with a 0.3 Hz cut off was applied to distinguish between gravitational and body motion components. Features were extracted from each window, including time and frequency domain measures like mean, standard deviation, energy, and entropy, to support accurate machine learning classification.

1.1.2 Dataset and Machine Learning Application

The UCI Machine Learning Repository, in collaboration with Team Kaggle, provided a dataset consisting of two main files: a training set with 7,352 rows and 564 columns, and a test set with 2,947 rows and 564 columns. Each row represents a sliding window of sensor data, and each column corresponds to a specific feature derived from the raw signals. The goal was to develop models to learn from the training data and accurately predict activity labels on the test data. Various algorithms, including decision trees, support vector

machines, and neural networks, were utilized. Human Activity Recognition (HAR) using smartphone data is an effective method for classifying physical activities through machine learning, with applications in health monitoring, fitness tracking, and context-aware computing. The dataset provides a valuable resource for developing and testing these models.

1.2 OVERVIEW

Human Activity Recognition (HAR) is a field of study focused on identifying physical activities performed by individuals based on data collected from sensors. With the proliferation of smartphones equipped with various sensors such as accelerometers and gyroscopes, HAR has gained significant attention due to its applications in health monitoring, sports analytics, and personal fitness tracking. One notable dataset that facilitates research in this area is available on Kaggle: the "Human Activity Recognition Using Smartphones Dataset by UCI Machine Learning."

1.2.1 Dataset Description

The Kaggle dataset used for Human Activity Recognition includes data collected from 30 participants who performed six different activities while wearing the smartphone on their waist. The six activities are:

- Walking
- Walking upstairs
- Walking downstairs
- Sitting
- Standing
- Laying

The smartphone's built-in accelerometer and gyroscope recorded 3-axis linear acceleration and 3-axis angular velocity consistently at a rate of 50Hz.The dataset contains pre-processed, labelled data split into training and test sets, making it suitable for machine learning applications.

1.2.2 Data Structure

The dataset consists of:

- Features: A 561-feature vector with time and frequency domain variables.
- Activity Labels: The type of activity performed.
- Subjects: Identifiers for the participants.

The features include both raw signal data and derived data, such as means and standard deviations of the signals over fixed width sliding windows.

1.2.3 Pre-processing

Pre-processing steps involve:

- Data Cleaning: Ensuring there are no missing values.
- Feature Scaling: Normalizing the data to ensure uniformity.
- Feature Selection: Selecting the most relevant features to reduce dimensionality and computational cost.

1.2.4 Machine Learning Models

Various machine learning models can be applied to the dataset to classify the activities, including Logistic Regression, Decision Trees, Kernel SVM and Random forest with hyperparameter tunning and cross validation.

1.2.5 Evaluation Metrics

Model performance is typically evaluated using metrics such as:

- Accuracy: The proportion of correctly classified activities.
- Precision and Recall: Metrics to evaluate the quality of positive predictions.
- F1 Score: The harmonic means of precision and recall, providing a balance between the two.

1.2.6 Applications

HAR using smartphone data has numerous applications, including:

- Health Monitoring: Tracking daily activities to identify patterns in patients with chronic conditions.
- Fitness Tracking: Providing insights into the intensity and type of workouts.
- Elderly Care: Monitoring the movements of elderly individuals to detect falls or inactivity.

2. PROBLEM DEFINATION

Identifying human activities from smartphone sensor data is complex due to its large dimensions. Our system reduces these dimensions and efficiently employs machine learning to classify six activities using accelerometer and gyroscope data. Robust pre-processing and feature extraction ensure generalization across subjects. Validated on a Kaggle dataset from 28 individuals, the model demonstrates strong performance for real-world applicability.

2.1 Objective and Scope

The Human Activity Recognition database was formed by recording 30 participants engaging in daily activities

while carrying a waist-mounted smartphone equipped with inertial sensors. The goal is to classify six activities: standing, lying down, walking, walking upstairs, walking downstairs, and sitting. The project aims to train a machine learning model using this data to classify the participants' activities accurately..

3. LITERATURE SURVEY

Table-1, Literature Survey of Different Papers or Research in HAR

S.NO.	TITLE OF PAPER	AUTHOR NAME	TECHNOLOGY USED	ADVANTAGES	LIMITATIONS
01	A Public domain dataset for human activity recognition using smartphones [4].	Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz	• Utilizes built-in accelerometers and gyroscopes. • Measures acceleration and rotational motion. • Collects data during user activities. • Aids in developing algorithm-based activity identification.	• Provides valuable resource without specialized equipment. • Captures sensor readings from everyday phone use. • Offers common ground for algorithm comparison and benchmarking.	Real-world Application Recognition • Sensor quality varies between phone models. • User positioning affects performance.
02	Smartphone Based Data Mining for Human Activity Recognition [5].	Girija Chetty, Matthew White and Farnaz Akther.	Sensors Overview • Accelerometers track movement. • Gyroscopes measure rotation. • Magnetometers sense direction. • Techniques include feature extraction and classification algorithms. • Machine learning models often used for pattern classification.	Smartphone Wearables Solution • Eliminates need for dedicated wearables. • Data collection passively via everyday phone usage. • Multiple sensors provide comprehensive movement patterns. • Large-scale deployment possible due to smartphone ubiquity.	Impact of Sensor Quality on Data Collection • Factors include sensor quality, phone placement, and activity complexity. • Data collection requires user consent to avoid privacy violations. • Continuous sensor use can drain battery life.
03	Activity Classification with Smartphone Data [6].	Matt Brown, Trey Deitch, and Lucas O'Conor	Sensor Data Processing Overview • Utilizes accelerometers, gyroscopes, and magnetometers to measure acceleration, track rotation, and sense direction. • Sensor readings are captured at regular intervals during various activities. • Raw sensor data is pre-processed for analysis. • Relevant data points extracted from pre-processed data. • Machine learning models trained on labelled data to learn patterns between features and specific activities. • Models classify new data into activity categories.	Smartphone Wearables: • Eliminates need for dedicated wearables. • Data collection passively through everyday phone usage. • Multiple sensors provide comprehensive movement patterns. • Widely applicable in healthcare, personalized marketing, and context-aware computing.	Impact of Sensor Quality and Phone Placement • Sensor quality, phone placement, and activity complexity affect data collection. • User consent is crucial to avoid privacy violations. • Tracking activities and data storage and usage are crucial. • Continuous sensor use can drain battery life. • Not all relevant information captured for complex activities.

4. SYSTEM ARCHITECTURE

4.1 System Architecture

To design the Human Activity Recognition System using Smartphone Data, uses the Data Analysis and Machine learning methods. So, the system architecture has the following steps to be implemented Fig-1:

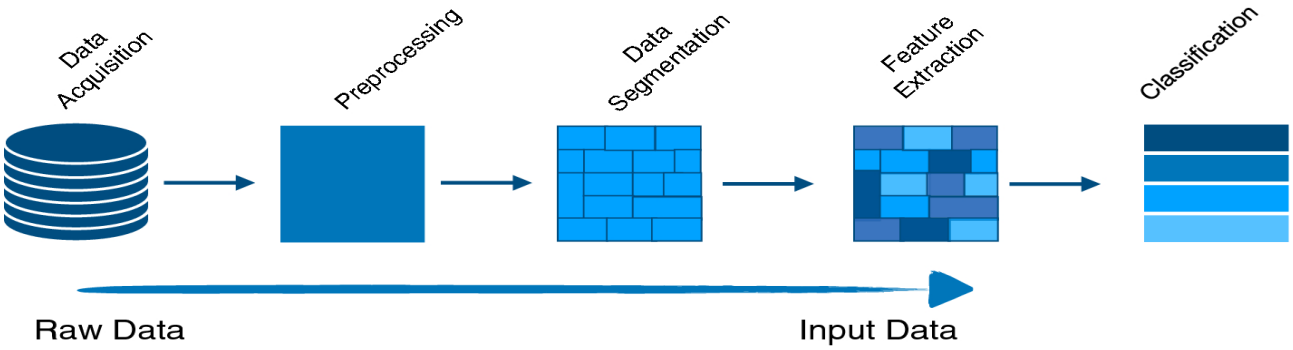

Fig -1: Data Movement

DATA ACQUISITION: In Human Activity Recognition (HAR) using smartphone data, data acquisition is the foundation for building effective models. Here's a breakdown of the key aspects:

Sensors: Smartphones come equipped with various sensors that capture information about user movement and phone orientation.

The primary sensors used in HAR include Accelerometer, Gyroscope and Magnetometer.

Data Collection Techniques:

- App-based collection: A dedicated app is developed to capture sensor data while users perform specific activities. Users might be instructed to follow pre-defined routines or perform activities freely while the app gathers data in the background.
- Continuous collection: Sensor data is collected passively throughout the day during regular phone usage. This approach allows capturing a wider range of everyday activities but requires careful selection of relevant data segments for analysis.

Data Labelling:

- For supervised machine learning algorithms, data needs to be labelled with the corresponding activity (e.g., walking, standing, sleeping).
- In app-based collection, users might manually label data segments after performing activities.
- For continuous collection, external sources like video recordings or user annotations can be used to label specific activity periods within the collected data.

Challenges in Data Acquisition:

- Privacy Concerns: Balancing data collection with user privacy is crucial. Users need to be aware of what data is being collected, how it's used, and have the option to opt-out.
- User Burden: App-based collection with frequent labelling can be tedious for users. Continuous collection needs to be optimized to minimize battery drain.
- Data Quality: Sensor quality can vary between phones, and factors like phone placement can impact data accuracy.

These samples are Provided by UCI (University of California Irvine) and Kaggle Team with use of Gyro sensors.

4.2 Data Preprocessing

Data preprocessing is a crucial step in Human Activity Recognition (HAR) using smartphone data. It prepares the raw sensor data for effective analysis and improves the performance of machine learning models used for activity classification. Here's a breakdown of the key data preprocessing techniques:

Cleaning:

- Missing Values: Identify and handle missing data points. This might involve removing data entries with missing values, imputation (filling in missing values with statistical methods), or interpolation (estimating missing values based on surrounding data).
- Outlier Removal: Extreme sensor readings that deviate significantly from the norm can be outliers. These outliers can be removed or mitigated through techniques like capping.

Normalization:

Sensor data can be measured in different units (e.g., g-force for accelerometers). Normalization techniques like min-max scaling or z-score normalization ensure that all features are on a similar scale, allowing the machine learning model to focus on the relative patterns within the data.

4.3 Data Segmentation

Raw sensor data is a continuous stream. Segmentation involves dividing the data into smaller segments (windows) that represent specific instances of activities. Window size and overlap between windows are crucial parameters. A larger window might capture more context but lose granularity, while a smaller window might miss important details. Overlap helps capture transitions between activities.

4.4 Feature Extraction

Extracting relevant features from the segmented data is vital. Common features include:

- Statistical features*:* Mean, standard deviation, variance, skewness, kurtosis - capture basic properties of the movement within the window.
- Frequency domain features: Extracted using techniques like Fast Fourier Transform (FFT) to identify dominant frequencies in the data, which can be helpful for differentiating activities like walking and running.

4.5 Classifications

Supervised machine learning algorithms are commonly used for activity classification in HAR. During training, the model learns the patterns that differentiate between activities based on the features extracted from the pre-processed data.

Common Classification Algorithms:

- Logistic Regression (LR): Simple Classifier.
- *K-Nearest Neighbours (KNN):* A simple yet effective algorithm that classifies new data points based on their similarity to labelled data points in the training set.
- Support Vector Machines (SVM) construct a hyperplane that optimally distinguishes data points representing different activities within a high-dimensional feature space.
- Decision Trees*:* Learns a tree-like structure where each branch represents a decision based on a feature, leading to a final classification label (activity).
- Random Forests: An ensemble method that combines predictions from multiple decision trees, improving overall accuracy and robustness.

5. DESIGN AND IMPLEMENTATION

5.1 Designing Workflow and Implementation:

In the below Figure-2, it shows the overall strategy to implement the workflow process of the Human Activity Recognition Model which classifies for seven steps:

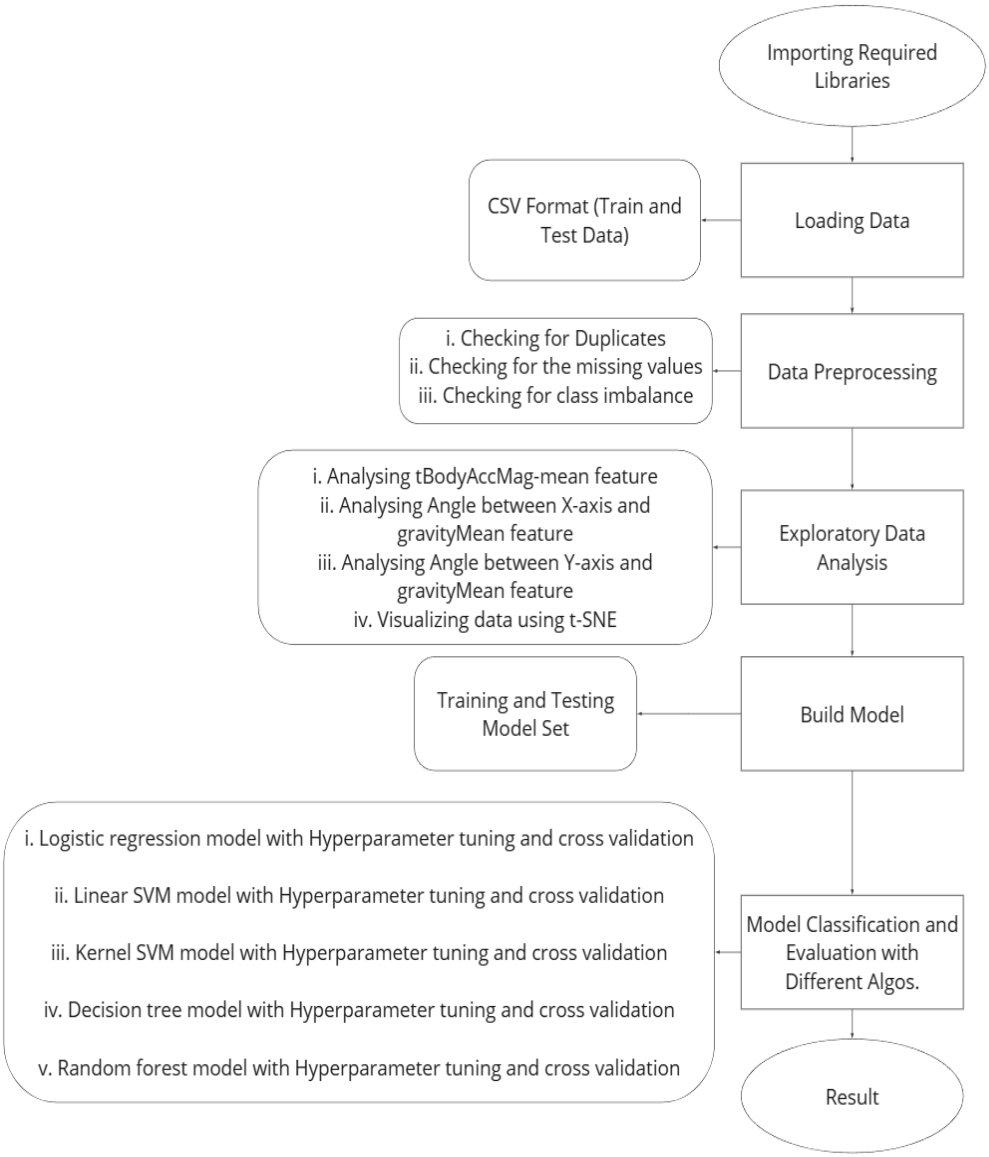

Figure -2, Workflow

5.1.1. Importing Required Libraries:

Import the required Libraries that are useful for the building model for example:

Table-2, shows the required libraries to be import and why?

Import Required	Why to Import?
NumPy	Dealing with Arrays
Pandas	Dealing with Data Frame, mostly, used in importing csv dataset.
Seaborn and Matplotlib	Matplotlib is the foundation for creating various data visualizations in Python. It offers a wide range of plot types and customization options. Seaborn builds on top of Matplotlib, providing a higher-level interface specifically designed for creating statistical graphics. It simplifies creating aesthetically pleasing and informative visualizations with features like dataset-oriented

	plotting and automatic statistical aggregation.
Data Analysis	
Counter	The Counter class from the collections library in Python provides an efficient way to count the occurrences of elements in a collection (like a list or string). It creates a dictionary-like object where elements are keys and their counts are values.
PCA	The PCA class from sklearn.decomposition allows you to perform Principal Component Analysis (PCA) in Python. PCA reduces the dimensionality of your data by finding a new set of features (principal components) that capture most of the variance, making it useful for data compression and visualization.
TNSE	You'd use from sklearn.manifold import TSNE in Python to visualize high-dimensional data. Unlike PCA (which is linear), TSNE(t-Distributed Stochastic Neighbor Embedding) excels at uncovering non-linear relationships between data points, allowing you to explore complex datasets in lower dimensions for better visualization.
Model	
RandomSizedSearchCV	Regular grid search can be computationally expensive. RandomizedSearchCV from sklearn.model_selection offers a more efficient alternative for hyperparameter tuning. It randomly samples parameter combinations from defined distributions, reducing computation cost while still providing good estimates for optimal hyperparameters.
Machine Learning Model	
LogisticRegression	The LogisticRegression class from sklearn.linear_model lets you build machine learning models for classification tasks in Python. It works well for problems with binary or multiple class labels, using a logistic function to estimate the probability of an instance belonging to a particular class.
SVC	The SVC class from sklearn.svm implements Support Vector Machines (SVMs) for classification in Python. SVMs excel at finding hyperplanes that best separate data points belonging to different classes, offering good performance in high-dimensional spaces and with limited training data.
DecsionTreeClassifier	The DecisionTreeClassifier from sklearn.tree lets you create machine learning models based on decision trees in Python. These trees make predictions by recursively splitting the data based on features, offering a clear and interpretable model for classification tasks.
RandomForestClassifier	The RandomForestClassifier from sklearn.ensemble is used for building robust classification models in Python. It combines multiple decision trees, reducing the risk of overfitting and improving model performance compared to a single decision tree.
Metrics Evaluation	
Confusion_matrix	• It helps evaluate your classification model's performance by providing a table that shows how many instances were correctly and incorrectly classified. • This breakdown (true positives, false negatives, etc.) allows you to identify areas for improvement in your model.
Accuracy_score	• It calculates the accuracy of your classification model, which is the proportion of predictions correctly made. • This simple metric provides a quick overall understanding of how well your model performs.
Classification_report	• It generates a report with metrics like precision, recall, F1-score for each class, giving insights beyond just overall accuracy. • This helps identify potential issues like class imbalance or uneven model performance across different categories.

5.1.2. Loading Data

Data provided by Kaggle, and the UCI Machine Learning department involved 30 volunteers aged 19-48 performing six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) while wearing a waist-mounted Samsung Galaxy S II smartphone. The smartphone's accelerometer and gyroscope captured 3-axial linear acceleration and angular velocity at 50Hz. Experiments were video recorded for manual labelling. The dataset was split into 70% for training and 30% for testing. Pre-processing involved noise filtering and sampling in 2.56-second sliding windows with 50% overlap. Using a Butterworth low-pass filter, body acceleration and gravity were separated, and features were extracted from both time and frequency domains. And finally provided in Kaggle Website: https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones in csv file. [10].

CSV (Comma-Separated Values) is a type of excel format where the UCI provided two dataset which is "train.csv" and "test.csv". The resulting dataset, hosted by UCI Machine Learning in collaboration with Team Kaggle, includes training (7352 samples, 563 features) and testing (2947 samples, 563 features) sets. Let's explore the age of people participated in this experiment is from 19-48 in total 30 so, in training dataset total contribution is:

```
subject
25    409
21    408
26    392
30    383
28    382
27    376
23    372
17    368
16    366
19    360
1     347
29    344
3     341
15    328
6     325
14    323
22    321
11    316
7     308
5     302
8     281
Name: count, dtype: int64
```

5.1.3. Data Preprocessing

In HAR model, used three steps:

i. Checking for Null Values

In dataset provided by UCI and Kaggle there is no data or no rows/columns which is NaN.

```
Total number of missing values in train :  0
Total number of missing values in train :  0
```

Figure-3, Total Number of Missing Values in Dataset

ii. Checking for Duplicates

In dataset provided by UCI and Kaggle there is no data which is duplicated.

```
Number of duplicates in train :  0
Number of duplicates in test :  0
```

Figure-4, Number of Duplicates in Dataset

iii. Checking for Imbalance

In dataset provided by UCI and Kaggle, there is almost same number of observations across all the six activities, so this data does not have class imbalance problem.

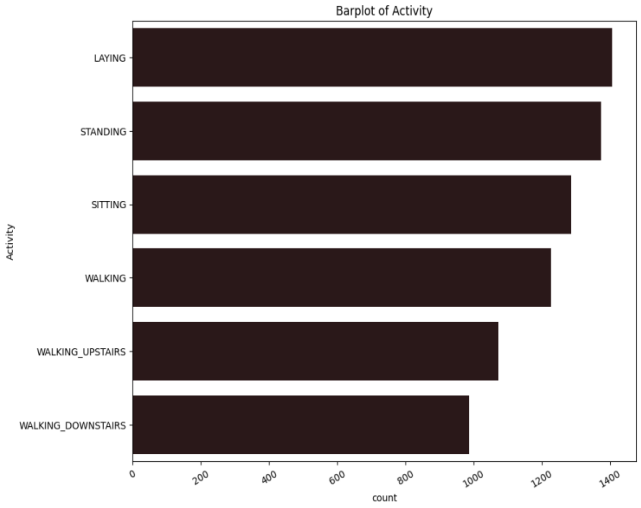

Figure-5, Imbalance Graph

5.1.4. Exploratory Data Analysis

In HAR, I have focused on four EDA parts:

i. Mean Feature- Analysis tBodyAccMag-mean

There are primarily 'acceleration' and 'gyroscope' features, with a few 'gravity' features also present. It's remarkable how many features there are given the limited number of sensors used.

Based on the typical nature of activities, we can broadly categorize them into two groups:

- *Static and dynamic activities:*

SITTING, STANDING, LAYING are considered static activities with minimal motion. WALKING, WALKING_DOWNSTAIRS, WALKING_UPSTAIRS are considered dynamic activities involving significant motion.

Let's use the tBodyAccMag-mean() feature to distinguish between these two broader sets of activities. If we aim to develop a simple classification model using one variable at a time, the probability density function (PDF) is invaluable for assessing the importance of continuous variables. Let's plot density plot:

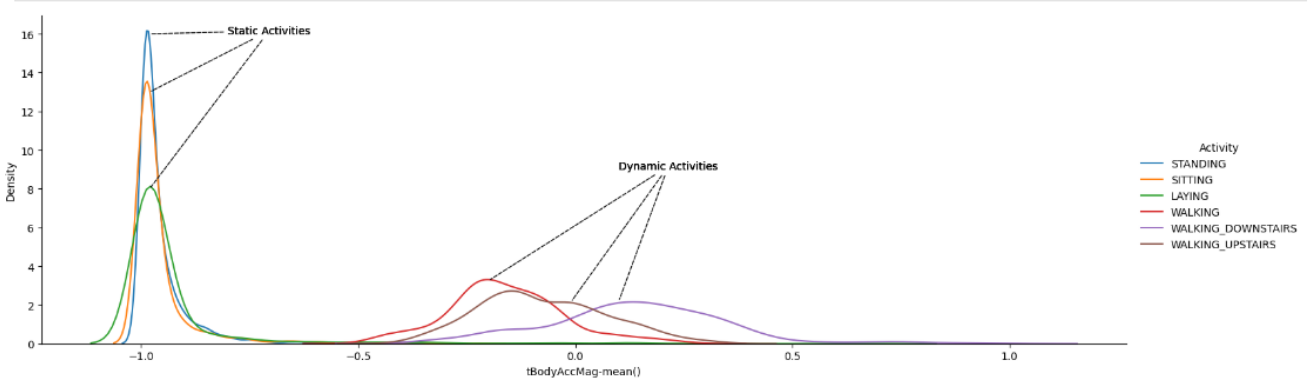

Figure-6, Activities Density Plot _ Static and Dynamic Distributions

Using the above density plot we can easily come with a condition to separate static activities from dynamic activities. Therefore, with plot we get to the condition that, if tBodyAccMag-mean() is lesser than -0.5 then activities are static in nature or else they are dynamic.

- Let's have a closer view on the PDFs of each activity under static and dynamic categorization.

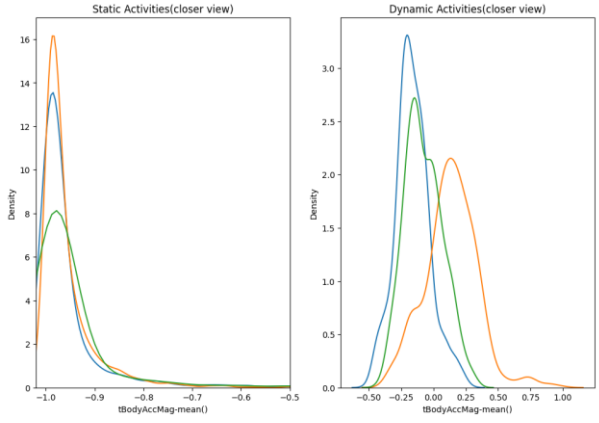

Figure-7, Closer Activities Plots

- The insights obtained through the density plots can be represented using Box plots. Let's plot the boxplot of Body Accelartion Magnitude mean(tBodyAccMag-mean()) across all the six categories.

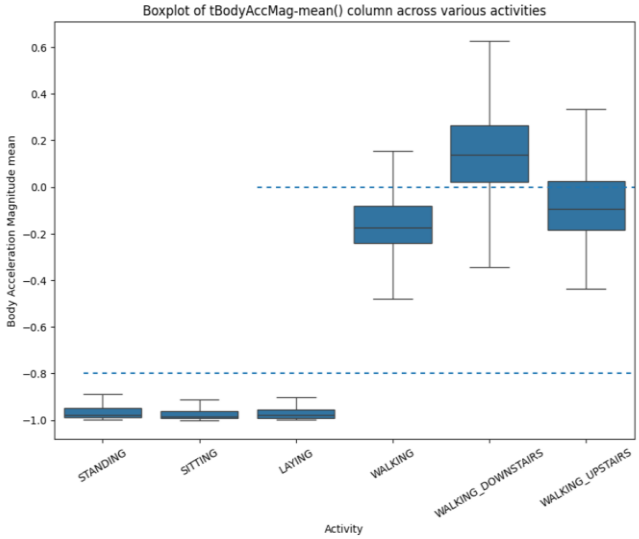

Figure-8, Box Plot for Activities

Using boxplot again we can come with conditions to separate static activities from dynamic activities. Therefore, form the plot we come to a condition that if tBodyAccMag-mean() is less than -0.8 then box plot represents the activities as static and if tBodyAccMag-mean() is greater than -0.6 then it represents as dynamic activities.

Also, we can easily separate WALKING_DOWNSTAIRS activity from others using boxplot like, if the condition tBodyAccMag-mean() is greater than 0.02 then person performing activity is Walking Downstairs if not then plots would specify other activities.

But still 25% of WALKING_DOWNSTAIRS observations are below 0.02 which are misclassified as others, so this condition makes an error of 25% in classification.

ii. Analysing Angle between X-axis and gravityMean feature

Using Box Plot Plotting angle between X-axis and gravity-mean features:

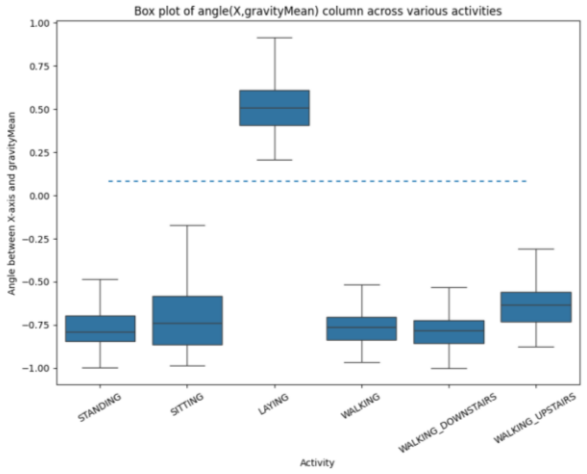

Figure-9, Angle Between X-axis and Gravity Mean

From the boxplot we can observe that angle(X,gravityMean) perfectly seperates LAYING from other activities. Condition is that if angle of X and gravity Mean is greater than 0.01 then it specifies as Laying activity if not just mention more on case other activities.

iii. Analysing Angle between Y-axis and gravityMean feature

Plotting the Box Plot:

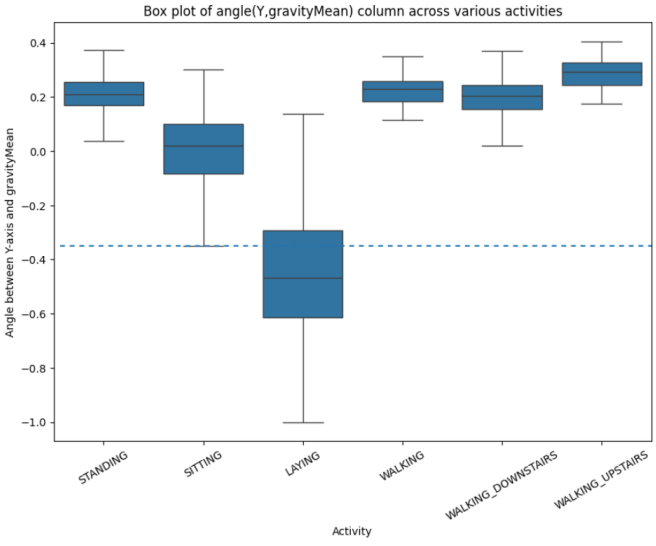

Figure-10, Box Plot of Angle Between Y-axis and Gravity Mean

iv. Visualizing data using PCA (Principal Component Analysis) and t-SNE (TSNE: t-distributed Stochastic Neighbor Embedding)

- *PCA:*

Using PCA, data can be transformed from a highly dimensional space to a lower dimensional space while retaining significant information. With the training data consisting of 561 unique features, PCA allows us to visualize it in a 2D space effectively.

From the below graph, Using the two new components obtained through PCA we can visualize and separate all the six activities in a 2D space.

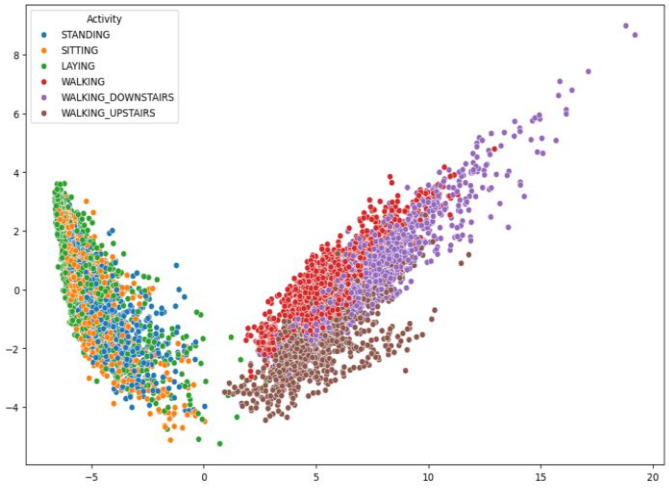

Figure-11, Visualizing Using PCA

- *t-SNE*

Uncovers non-linear relationships: Unlike linear dimensionality reduction techniques like PCA, TSNE excels at revealing non-linear structures in your data. This allows you to explore complex datasets where data points might not lie in easily separable linear clusters.

Better visualization in lower dimensions: By projecting high-dimensional data onto a lower-dimensional space (often 2 or 3 dimensions), TSNE enables you to visualize complex relationships that might be difficult to grasp in the original high-dimensional space. This facilitates easier data exploration and interpretation.

Using t-SNE, data can be visualized from a high-dimensional space to a lower-dimensional space while preserving substantial information. Given that the training data comprises 561 unique features, t-SNE enables us to visualize it effectively in a 2D space.

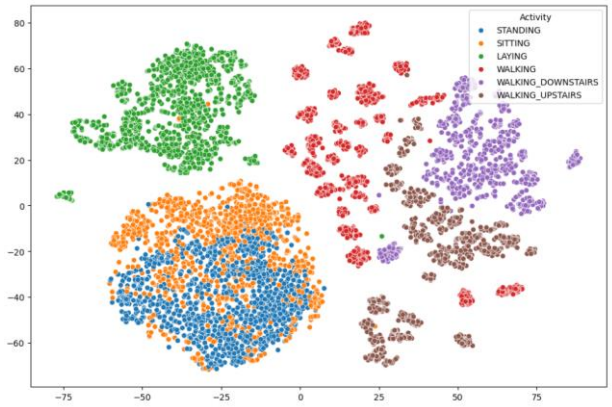

Figure-12, Visualizing using TNSE.

5.1.5 Build Model

Building Model refers to define training and testing model. Splitting training and testing data is crucial in machine learning for two key reasons:

- *Prevents Overfitting:* Imagine training a model on all your data. It might perfectly memorize the specific details of that data, leading to great performance on that specific set. However, when presented with new, unseen data, the model might fail because it was too focused on the training data's quirks instead of learning generalizable patterns. Splitting the data ensures a dedicated testing set that the model hasn't "seen" during training. Evaluating on this unseen data gives a more realistic picture of how well the model generalizes to new information.
- *Unbiased Evaluation:* If you use the same data for both training and testing, you're essentially testing the model on data it already "knows." This leads to an inflated sense of accuracy and doesn't reflect how the model performs on truly new data. Splitting the data ensures an unbiased evaluation where the model is assessed on data it wasn't tailored to fit during training.

Here is the below demonstration of implementation of Building Training and Testing Dataset, also with number of splatted rows.

```
Shape Of Training Data Set :  (7352, 561)
Shape Of Testing Data Set : (999, 561)
Shape Of Train Label : (7352,)
Shape Of Test Label : (999,)
```

Figure-13, Build Model

5.1.6. Model Classification and Evaluation with Different Algorithms

Key Points: Here, I have used Hyperparameter Tunning and Cross Validation. Requirement of Doing:

- *Improved Generalizability:* Hyperparameter tuning helps find the best settings for your model, but it can lead to overfitting if not done carefully. Cross-validation provides a robust estimate of how well the tuned model performs on unseen data, ensuring it generalizes well to new situations.
- *Reduced Overfitting:* Overfitting occurs when a model memorizes the training data too well, losing its ability to adapt to new data. Cross-validation helps prevent this by evaluating the model on different data splits, forcing it to learn generalizable patterns.
- *Optimal Hyperparameter Selection:* Hyperparameter tuning alone can lead to picking settings that work well on a specific training set but might not generalize. Cross-validation allows you to compare performance across different hyperparameter combinations, guiding you towards the best overall settings.
- *Unbiased Performance Evaluation:* Without cross-validation, it's easy to overestimate your model's performance on unseen data. Cross-validation provides a more realistic picture by averaging performance across different data splits, giving a more unbiased evaluation.
- *Efficiency and Informed Decisions:* By combining hyperparameter tuning with cross-validation, you can efficiently explore different settings and identify the best configuration for your model, saving time and resources in the long run.

In HAR model, I have used Different machine learning algorithms for modelling:

i. Logistic Regression with Hyperparameter and Cross_Validation

Therefore, when using LR algo:

```
Accuracy using Logistic Regression :  95.4954954954955
Best estimator :  LogisticRegression(max_iter=500)
Best set of parameters :  {'max_iter': 500}
Best score :  93.73035141996976
```

And, the Heatmap which shows the co-relation matrix:

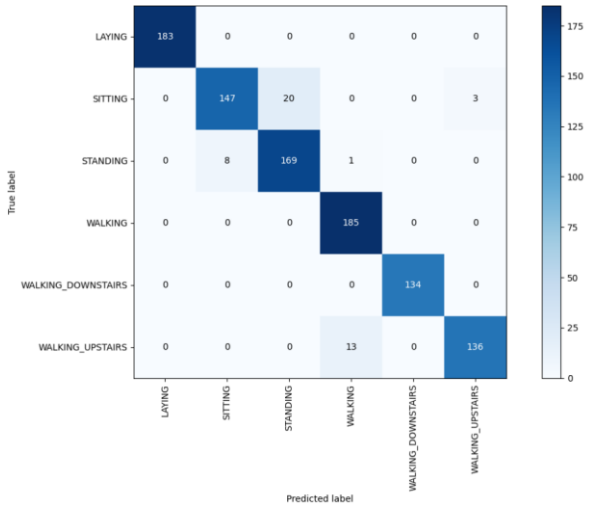

Figure-14, Heatmap of LR

ii. Kernel SVM model with Hyperparameter and Cross_Validation

When Using Kernel SVM,

Accuracy using Kernel SVM : 96.5965965965966
Best estimator : SVC(C=50)
Best set of parameters : {'kernel': 'rbf', 'C': 50}
Best score : 94.64109332023303

And, the Heatmap which shows the co-relation matrix:

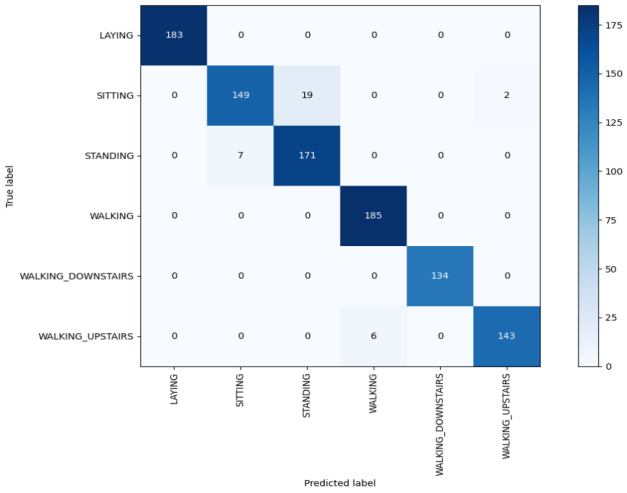

Figure-15, Heatmap of KERNEL SVM

iii. Decision Tree Model with Hyperparameter and Cross_Validation

When using Decision Tree Model,

Accuracy using Decision tree : 84.48448448448448
Best estimator : DecisionTreeClassifier(max_depth=8)
Best set of parameters : {'max_depth': 8}
Best score : 84.97097166534866

And, the Heatmap which shows the co-relation matrix,

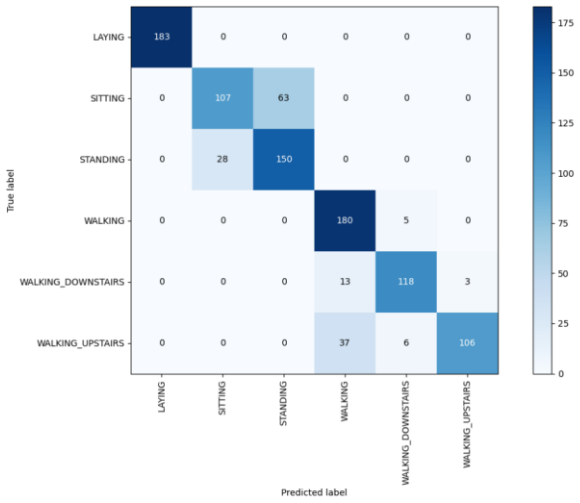

Figure-16, Heatmap of Decision Tree Classifier

iv. Random Forest Model with Hyperparameter and Cross_Validation

When using the Random Forest Model,

Accuracy using Random forest : 90.990990990991
Best estimator : RandomForestClassifier(max_depth=12, n_estimators=70)
Best set of parameters : {'n_estimators': 70, 'max_depth': 12}
Best score : 92.31547792468449

And, the Heatmap which shows the co-relation matrix,

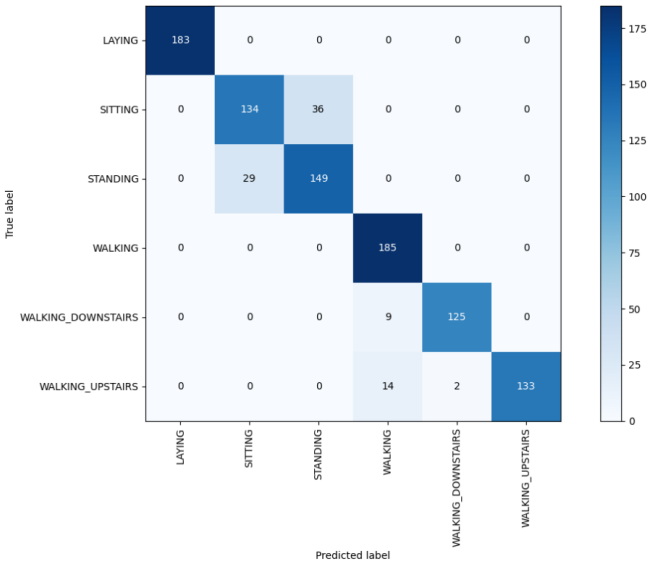

Figure-17, Heatmap of Random Forest Classifier

5.1.7. Outcome

When using the different machine learning algorithm, Kernel SVM found out to have a best estimator's accuracy.

5.2. System Requirements

There are two types of requirement analysis Projected on HAR models:

i. Software Requirements:

Table-3: Software Requirements

Requirement Type	Software or Package	Purpose	Recommended Version
Operating System	Windows, macOS, Linux	Platform to run Jupyter Notebook	Latest Stable
Programming Language	Python	Core language for Jupyter Notebook	3.8 or newer
Notebook Application	Jupyter Notebook	Interface for creating and managing notebooks	Latest Release
Package Manager	pip or conda	Tool to install and manage Python packages	Latest Release
Virtual Environment	conda	Isolates Python environments (Anaconda)	Latest Release
Basic Libraries	NumPy	Fundamental package for scientific computing	Compatible with Python version
	Pandas	Data structures & analysis	Compatible with Python version
	Matplotlib	Plotting library	Compatible with Python version
Advanced Libraries	scikit-learn	Machine learning library	Compatible with Python version
	TensorFlow or PyTorch	Deep learning frameworks	Compatible with Python version
Extensions	JupyterLab	Next-generation web-based UI for Project Jupyter	Latest Release
Version Control	Git	Version control system	Latest Release
Collaboration Tools	GitHub or GitLab	Remote storage and collaboration on notebooks	Latest Release

ii. Hardware Requirements:

Table-4, Hardware Requirements

Usage Level	CPU Recommendation	Memory Recommendation	Storage Recommendation	Additional Notes
Minimum	Single-core processor	1 GB RAM	Few GB	Suitable for basic tasks and learning
Recommended	Dual-core processor	4 GB RAM or more	10 GB on SSD	Good for medium-sized datasets and moderate tasks
Advanced	Quad-core processor or higher	8 GB RAM or more	20 GB on SSD	Best for large datasets and intensive computations
GPU-Enhanced	Compatible NVIDIA GPU	16 GB RAM or more	50 GB on SSD	Necessary for GPU-accelerated computing tasks

6. RESULT AND ANALYSIS

6.1 Analysis

So, Comparing different algorithms evaluation:

Table-5, Analysis of Different Algorithm.

	Model	Accuracy %	Best Score Accuracy %
1	Kernel SVC	96.596597	94.641093
3	Random Forest	90.990991	92.315478
0	Logistic Regression	84.484484	93.730351
2	Decision Trees	84.484484	84.970972

Here, as you can see the Kernel SVC or SVM gives the best accurate value.

6.2 Result

Now, according to above analysis to Build Model Kernel SVC found to showing optimal accuracy and average accuracy (best score accuracy %) of 96.59% and 94.64% respectively.

Therefore, building the HAR model with KERNEL SVM will gives the optimal recognition.

Hence, Visualization: Depending on the application, the results might be visualized. For example, you might see heatmaps overlaid on data frames highlighting where specific activities are detected. This heatmap achieved is much optimized then other algorithm models.

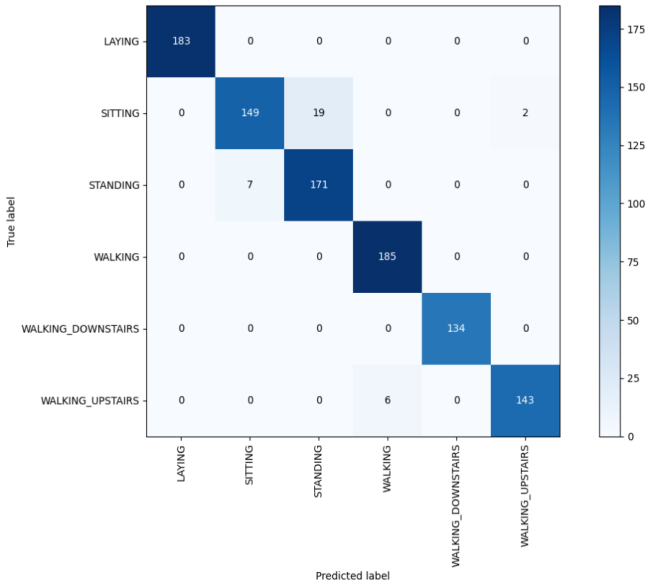

Figure-18, Resultant Heatmap of KERNEL SVM

7. CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

- Studies on Kaggle datasets have demonstrated the feasibility of using smartphone sensor data (accelerometer, gyroscope) for recognizing various activities like walking, standing, sitting, upstairs/downstairs movement.

- Machine learning models like KNN, SVM, Random Forests have achieved promising accuracy in classifying these activities based on extracted features from sensor data.

Kernel SVC shows the Best Score Accuracy.

Accuracy using Kernel SVM: 96.5965965965966

7.2 Future Scope

The field of Human Activity Recognition (HAR) using smartphone data continues to evolve, and there are exciting future research directions. The future of HAR lies in interdisciplinary collaboration, innovative model architectures, and addressing practical challenges to make smartphone-based activity recognition more accurate, efficient, and applicable to diverse scenarios. Here are some potential areas of focus:

i. Multimodal Sensor Fusion:

- Combining data from various smartphone sensors (accelerometer, gyroscope, magnetometer, etc.) with other modalities (such as audio or visual data) can enhance activity recognition accuracy.
- Exploring fusion techniques and designing robust models that leverage multiple sensor inputs is an interesting avenue for research.

ii. Real-Time and Continuous Monitoring:

- HAR models can be extended to provide real-time feedback and alerts.
- Developing systems that continuously monitor activities and adapt to changes in behavior or context is crucial for applications like healthcare and safety.

iii. Transfer Learning and Generalization:

- Investigating transfer learning approaches can help generalize HAR models across different user populations, demographics, and cultural contexts.
- Pretraining on large datasets and fine-tuning on specific tasks can improve model performance.

iv. Edge Computing and Energy Efficiency:

- Deploying lightweight models directly on smartphones (edge devices) can reduce reliance on cloud servers.
- Optimizing energy consumption during data collection and inference is critical for prolonged smartphone usage.

v. Long-Term Behavioral Patterns:

- Understanding long-term behavioral patterns can provide insights into lifestyle changes, mental health, and disease progression.
- Investigating how HAR models can capture and analyze extended time series data is valuable.

vi. Benchmark Datasets and Evaluation Metrics:

- Creating standardized benchmark datasets for smartphone-based HAR is essential.
- Defining consistent evaluation metrics and protocols will facilitate fair comparisons across studies.

8. REFRENCES

[1] (2012). Mobile Phone Access Reaches Three Quarters of Planet's Population. The World Bank. Available: http://www.worldbank.org/en/news/press-release/2012/07/17/mobile-phoneaccess-reaches-three-quarters-planets-population

[2] M. Schirmer, H. Höpfner. Smartphone Hardware Sensors. Available: https://www.uni-weimar.de/kunst-und-gestaltung/wiki/images/Zeitmaschinensmartphonesensors.pdf

[3] O. C. Ann and L. B. Theng, "Human activity recognition: A review," 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014), Batu Ferringhi, 2014, pp. 389-393. doi: 10.1109/ICCSCE.2014.7072750

[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. A Public domain dataset for human activity recognition using smartphones. ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence

[5] Girija Chetty, Matthew White and Farnaz Akther. SmartPhone Based Data Mining For Human Activity Recognition. Information and Communication Technologies (ICICT 2014). Procedia Computer Science 46 (2015) 1181 – 1187

[6] Matt Brown, Trey Deitch, and Lucas O'Conor. Activity Classification with Smartphone Data. Stanford CS 229, Fall 2013

[7] Sensors in Android. Available: http://developer.android.com/guide/topics/sensors/sensors_overview.html [8] Metabolic equivalent. Wikipedia. Available: https://en.wikipedia.org/wiki/Metabolic_equivalent

[9] Jetté, M., Sidney, K. and Blümchen, G. (1990), Metabolic equivalents (METS) in exercise testing, exercise prescription, and evaluation of functional capacity. Clin Cardiol, 13: 555–565. doi: 10.1002/clc.4960130809 [10] MET Values for 800+ Activities. Available: http://golf.procon.org/view.resource.php?resourceID=004786

[10] Dataset Source: Human Activity Recognition with Smartphones, https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones