HUMAN ACTIVITY RECOGNITION USING SMARTPHONES.

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELORS OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

WITH SPECIALIZATION IN DATA SCIENCE

By

MANJUSHA SINGH YADAV

12107693

Supervisor

VED PRAKASH CHAUBEY

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Month - April, Year -2024

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B.Tech Dissertation/dissertation proposal entitled "HUMAN ACTIVITY RECOGNITION USING SMARTPHONES**"**, submitted by **Manjusha Singh Yadav** at **Lovely Professional University, Phagwara, India** is a bonafide record of his / her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

(Mr. Ved Prakash Chaubey)

Date:

Counter Signed by:

1) **Neutral Examiners:**

External Examiner

Signature: ________________

Name: ___________________

Affiliation: _______________

Date: ____________________

Internal Examiner

Signature: ________________

Name: ___________________

Date: ____________________

STUDENT DECLARATION

I hereby declare that the research work reported in the dissertation/dissertation proposal entitled "HUMAN ACTIVITY RECOGNITION USING SMARTPHONES" in partial fulfilment of the requirement for the award of Degree for Bachelors of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Ved Prakash Chaubey. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

Manjusha Singh Yadav

R.No_ 12107693

ACKNOWLEDGEMENT

First and foremost, I extend our heartfelt appreciation to our supervisor Mr. Ved Prakash Chaubey, whose guidance, insightful feedback, and unwavering support have been invaluable throughout this endeavoring. Their expertise and encouragement have truly enriched our learning experience.

I am also grateful to Lovely Professional University for providing us with the necessary resources, infrastructure, and conducive environment for conducting this research.
Last but not least, I express our gratitude to our friends and family for their patience, understanding, and unwavering support throughout this journey.

Thank you to everyone who has played a part, directly or indirectly, in the completion of this project. Your support has been instrumental in our success.

TABLE OF CONTENTS

ABSTRACT

Human activity recognition is gaining importance, not only in the view of security and surveillance but also due to psychological interests in understanding the behavioral patterns of humans. This report is a study on various existing techniques that have been brought together to form a working pipeline to study human activity in social gatherings. Humans are first detected with Deformable part models and tracked as a feature point in 2.5D co-ordinate system using Lucas-Kanade algorithm. Linear cyclic pursuit model is then employed to predict short-term trajectory and understand behavior.

OBJECTIVE

The primary objective of HAR is to develop algorithms and systems capable of accurately recognizing and categorizing human activities in real-time or from recorded data. These activities can include everyday actions such as walking, running, sitting, standing, climbing stairs, cycling, and more. By analyzing patterns and characteristics in sensor data or input signals, HAR systems can infer the ongoing activity of an individual and provide valuable insights for various applications.

Human Activity Recognition has numerous practical applications across different domains, including healthcare monitoring, fitness tracking, elderly care, behavior analysis, sports performance analysis, gesture recognition, security surveillance, and smart environments. Advancements in sensor technology, machine learning algorithms, and data analytics have contributed to the development of increasingly accurate and reliable HAR systems, opening up new opportunities for improving human-computer interaction and enhancing quality of life.

In summary, Human Activity Recognition is a multidisciplinary field that combines expertise from signal processing, machine learning, sensor technology, and domain-specific knowledge to develop intelligent systems capable of understanding and interpreting human activities from sensor data or input signals.

INTRODUCTION

1. Background:

Human Activity Recognition (HAR) is a field of study within the broader domain of artificial intelligence and machine learning that focuses on the automatic identification and classification of human activities based on sensor data or other types of input signals. HAR plays a crucial role in various applications, ranging from healthcare and fitness monitoring to smart homes, security systems, and human-computer interaction.

2. Industrial Application

Ergonomics: HAR can track worker movements to identify repetitive or strenuous postures that could lead to injuries. By analyzing activity data, companies can optimize workflows and recommend breaks to prevent musculoskeletal disorders.

Lone Worker Monitoring: In high-risk environments, smartphones can detect falls or inactivity, triggering alerts to emergency services if a lone worker is injured or incapacitated.

Picking and Packing Optimization: Warehouse workers equipped with smartphones can be tracked to optimize picking routes and improve efficiency in order fulfillment.

Asset Tracking: By combining activity recognition with location data, companies can track the movement of equipment and materials within a facility, streamlining logistics and preventing theft.

Predictive Maintenance: HAR data from factory workers can be used to identify changes in equipment operation patterns, potentially indicating malfunctions. This allows for preventive maintenance, reducing downtime and production losses.

Quality Control: Smartphone-based HAR can monitor worker activity on assembly lines, detecting deviations from standard procedures that might result in defective products

THEORETICAL BACKGROUND

What is Machine Learning?

Machine learning is a subfield of artificial intelligence (AI) that uses algorithms trained on data sets to create self-learning models that are capable of predicting outcomes and classifying information without human intervention. Machine learning is used today for a wide range of commercial purposes, including suggesting products to consumers based on their past purchases, predicting stock market fluctuations, and translating text from one language to another. In common usage, the terms "machine learning" and "artificial intelligence" are often used interchangeably with one another due to the prevalence of machine learning for AI purposes in the world today. But, the two terms are meaningfully distinct. While AI refers to the general attempt to create machines capable of human-like cognitive abilities, machine learning specifically refers to the use of algorithms and data sets to do so.

1)Supervised Learning Models: Supervised learning is a type of machine learning where the model learns a mapping between input features and output labels based on labeled training data. In supervised learning, the algorithm learns from a dataset that contains input-output pairs, where the inputs are the features or attributes, and the outputs are the corresponding labels or target variables. The goal is to learn a mapping function from the input variables to the output variable.

Regression Models: Predicts continuous values based on input features. Examples include Linear Regression, Polynomial Regression, and Support Vector Regression.

Classification Models: Predicts class labels or discrete outcomes. Examples include Logistic Regression, Decision Trees, Random Forests, and Support Vector Machines.

2) Unsupervised Learning Models: Clustering Models: Groups similar data points together based on some similarity metric. Examples include K-Means Clustering.

3)Semi-Supervised Learning Models: Combines Supervised and Unsupervised Learning: Uses both labeled and unlabeled data for training. Examples include Self-training and Co-training algorithms.

HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE:

2) Python 3.x version
3) NumPy
4) Windows operating system(32 bit or 64 bit)
5) Various sensors equipped with smartphones

SOFTWARE:

1. Python Programming Language
2. Python and ML libraries

METHODOLOGY

About the Dataset

The Human Activity Recognition database was built from the recordings of 30 study participants performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors. The objective is to classify activities into one of the six activities performed.

Description of experiment

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING UPSTAIRS, WALKING DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth lowpass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

Libraries Used

1. NumPy (`import numpy as np`):

- Role: NumPy is a fundamental package for numerical computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
- Usage in HAR: NumPy is essential for handling sensor data, which is typically represented as multi-dimensional arrays. It allows for efficient manipulation, computation, and transformation of the data.

2. Pandas (`import pandas as pd`):

- Role: Pandas is a powerful data manipulation library built on top of NumPy. It provides data structures and functions to efficiently manipulate and analyze structured data.
- Usage in HAR: Pandas is used for loading, preprocessing, and analyzing the dataset. It allows for tasks such as reading data from various file formats, cleaning missing values, and transforming data for further analysis.

3. Matplotlib (`import matplotlib.pyplot as plt`):

- Role: Matplotlib is a plotting library for Python. It provides a wide variety of plots and customization options for visualizing data.
- Usage in HAR: Matplotlib is used to create visualizations such as line plots, scatter plots, histograms, and heatmaps to explore the dataset, visualize sensor data, and display model performance metrics.

4. Seaborn (`import seaborn as sns`):

- Role: Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics.
- Usage in HAR: Seaborn is often used to create more visually appealing and informative plots compared to Matplotlib. It offers additional statistical

plotting functions and themes that can enhance the visualization of sensor data and analysis results.

5. Collections (`from collections import Counter`):

- Role: The `Counter` class from the `collections` module is used for counting occurrences of elements in a collection.
- Usage in HAR: In HAR projects, `Counter` can be used to count the occurrences of different activities or classes in the dataset, providing insights into the class distribution and imbalance.

6. scikit-learn (`from sklearn.*`):

- Role: scikit-learn is a machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It includes a wide range of machine learning algorithms and utility functions for model training, evaluation, and preprocessing.
- Usage in HAR: scikit-learn is used for building and training machine learning models for activity recognition. It provides algorithms such as logistic regression, support vector machines, decision trees, and random forests, which can be used for classification tasks in HAR projects. Additionally, scikit-learn provides tools for hyperparameter tuning, dimensionality reduction, and model evaluation.

Loading data

Loading data is a crucial step in any data analysis or machine learning project, including Human Activity Recognition (HAR).

Data preprocessing

1. Handling Missing Values:
- Identify missing values in the dataset using isnull() or info() functions.
- Decide how to handle missing values, either by removing rows with missing

values (dropna()), filling missing values with a specific value (fillna()), or using more sophisticated imputation techniques.

2. Removing Duplicates:

• Identify and remove duplicate rows from the dataset using the drop_duplicates() function.

3. Data Transformation:

• Convert data types of columns if necessary (e.g., converting categorical variables to numerical representation).

• Normalize or scale numerical features if needed to ensure that all features are on a similar scale.

4. Feature Engineering:

• Create new features from existing ones if necessary to capture additional information relevant to the problem.

5. Handling Outliers:

• Identify and handle outliers in the dataset, either by removing them or transforming them to mitigate their impact on the analysis.

6. Correcting Errors:

• Identify and correct any errors or inconsistencies in the data (e.g., typos, incorrect values).

Exploratory Data Analysis

1. **Analyzing tBodyAccMag-mean Feature:**

   - Start by examining the distribution of the tBodyAccMag-mean feature, which represents the mean magnitude of body acceleration.

   - Use descriptive statistics (e.g., mean, median, standard deviation) to summarize the feature's distribution.
     Visualize the distribution using histograms or box plots to understand its spread and identify any outliers.
     Compare the distribution of this feature across different activity classes to see if there are noticeable differences or patterns.

2. **Analyzing Angle between X-axis and gravityMean Feature and Analyzing Angle between Y-axis and gravityMean Feature:**

   - Follow a similar approach as above to analyze these features, which represent the angles between the X-axis/Y-axis and gravity.
   -
     Examine their distributions, descriptive statistics, and relationships

with activity classes.

3. Visualizing Data Using PCA:

- Perform Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while preserving most of its variance.
- Visualize the data in a lower-dimensional space (e.g., 2D or 3D) using the principal components as axes.
- Color-code the data points based on activity labels to observe if there are any clusters or separations in the reduced space.

4. Visualizing Data Using t-SNE:

- Apply t-distributed Stochastic Neighbor Embedding (t-SNE) to visualize the dataset in a lower-dimensional space, emphasizing local relationships.
- Similar to PCA, visualize the data points in a 2D or 3D space**,** but this time using t-SNE coordinates.
- Again, color-code the data points based on activity labels to identify clusters or patterns.

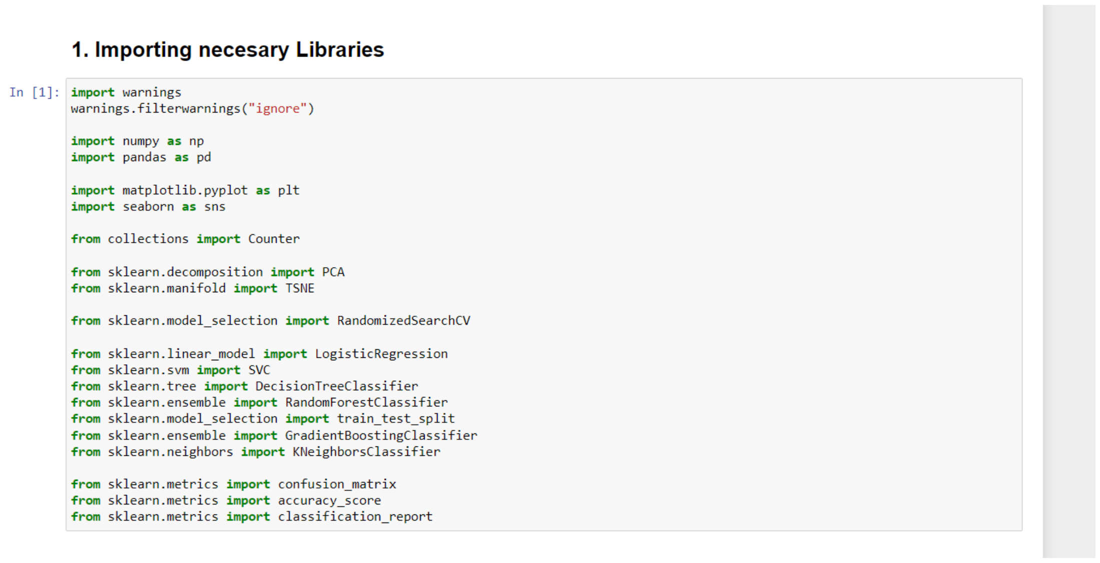
MODELS USED IN PROJECT:

Logistic Regression:
A simple linear model used for binary or multiclass classification tasks. It's interpretable and computationally efficient, making it a good baseline model.
Kernel Support Vector Machine (SVM):
SVMs are effective for classification tasks, especially when dealing with high-dimensional data. They can handle both linear and non-linear decision boundaries using different kernel functions.
Decision Trees:
Decision trees are non-parametric models that partition the feature space into regions and make predictions based on majority voting or averaging within each region. They are easy to interpret and can capture non-linear relationships in the data.
Random Forest:
Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions through averaging or voting. It improves generalization and robustness compared to individual decision trees.
Gradient Boosting Machines (GBM):
GBM is another ensemble learning technique that builds decision trees sequentially, where each tree corrects the errors of its predecessor. It typically achieves higher accuracy than Random Forest but may require more computational resources.
K-Nearest Neighbors (KNN):
KNN is a non-parametric lazy learning algorithm that classifies a data point based on the majority class of its nearest neighbors. It's simple and effective but may suffer from high computational costs for large datasets.
Artificial Neural Networks (ANN):
ANNs are a class of deep learning models inspired by the structure and function of the human brain. They consist of interconnected nodes organized in layers and are capable of learning complex patterns from data.
Convolutional Neural Networks (CNN):
CNNs are a type of ANN specifically designed for processing grid-like data, such as images or sensor data. They use convolutional layers to automatically extract relevant features from the input data, making them well-suited for HAR tasks involving sensor data.
Long Short-Term Memory (LSTM):
LSTMs are a type of recurrent neural network (RNN) designed to capture temporal dependencies in sequential data. They are effective for modeling time-series data and are commonly used in HAR for capturing temporal

patterns in sensor data sequences.
Gated Recurrent Unit (GRU):
GRUs are another variant of RNNs that are similar to LSTMs but have a simpler architecture. They are computationally efficient and can be used for modeling sequential data in HAR tasks.

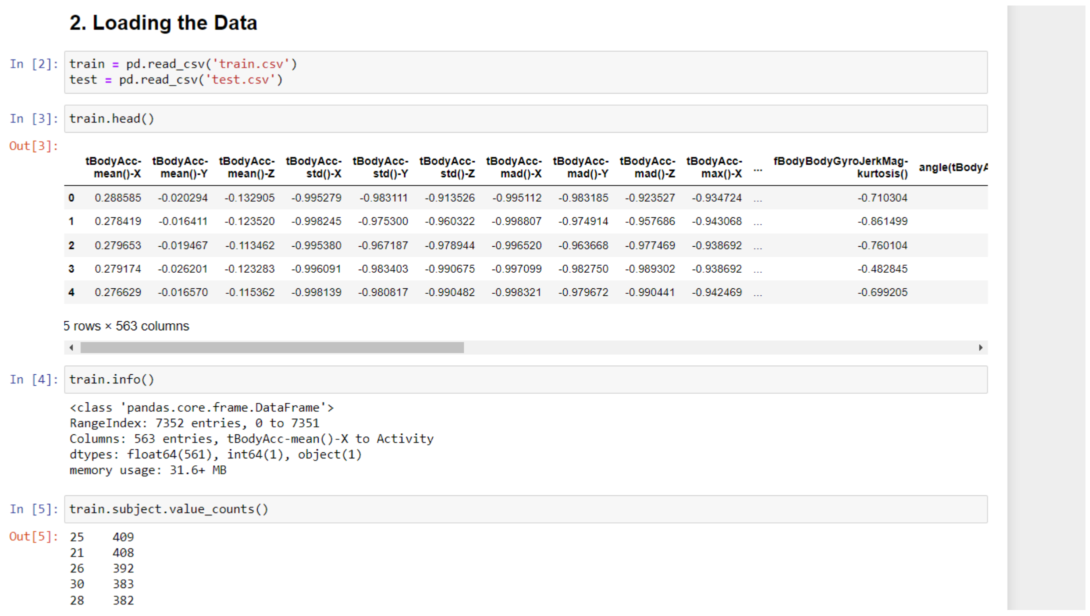
RESULT

Experimental Results:



1. Importing necesary Libraries

In [1]:
```
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from collections import Counter

from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

from sklearn.model_selection import RandomizedSearchCV

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```



2. Loading the Data

In [2]:
```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [3]:
```
train.head()
```

Out[3]:

	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-kurtosis()	angle(tBodyA
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.482845	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205	

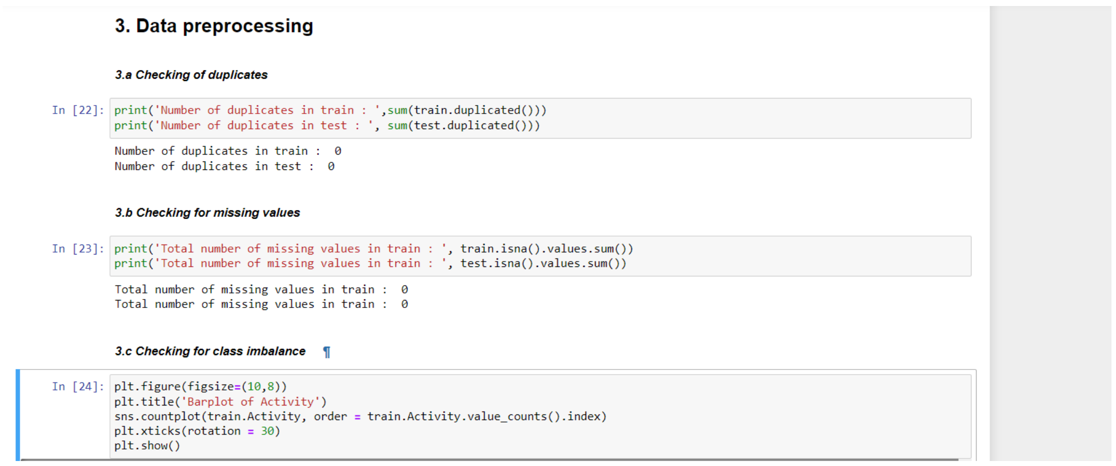
5 rows × 563 columns

In [4]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7352 entries, 0 to 7351
Columns: 563 entries, tBodyAcc-mean()-X to Activity
dtypes: float64(561), int64(1), object(1)
memory usage: 31.6+ MB
```

In [5]:
```
train.subject.value_counts()
```

Out[5]:
```
25    409
21    408
26    392
30    383
28    382
```



3. Data preprocessing

3.a Checking of duplicates

```
In [22]: print('Number of duplicates in train : ',sum(train.duplicated()))
         print('Number of duplicates in test : ', sum(test.duplicated()))
```

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

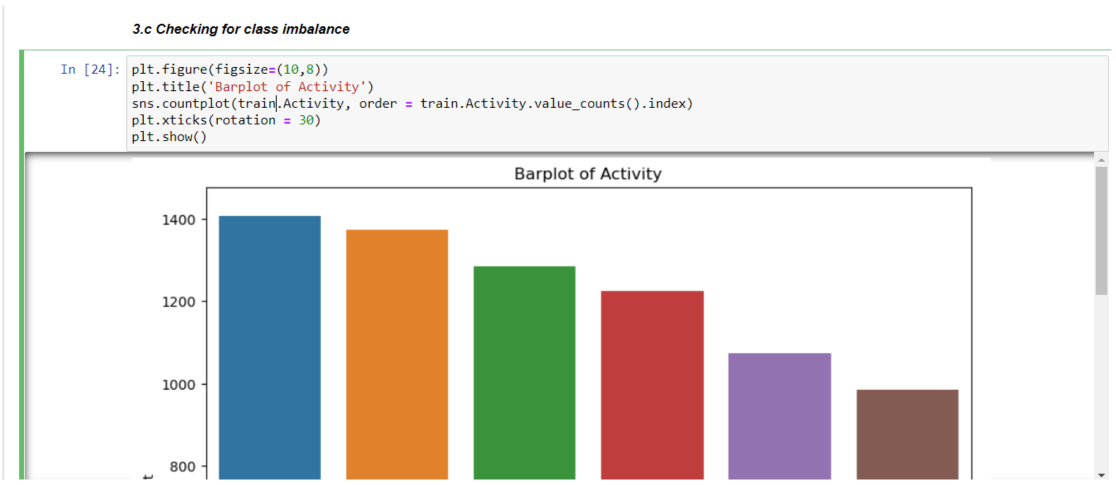
3.b Checking for missing values

```
In [23]: print('Total number of missing values in train : ', train.isna().values.sum())
         print('Total number of missing values in train : ', test.isna().values.sum())
```

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

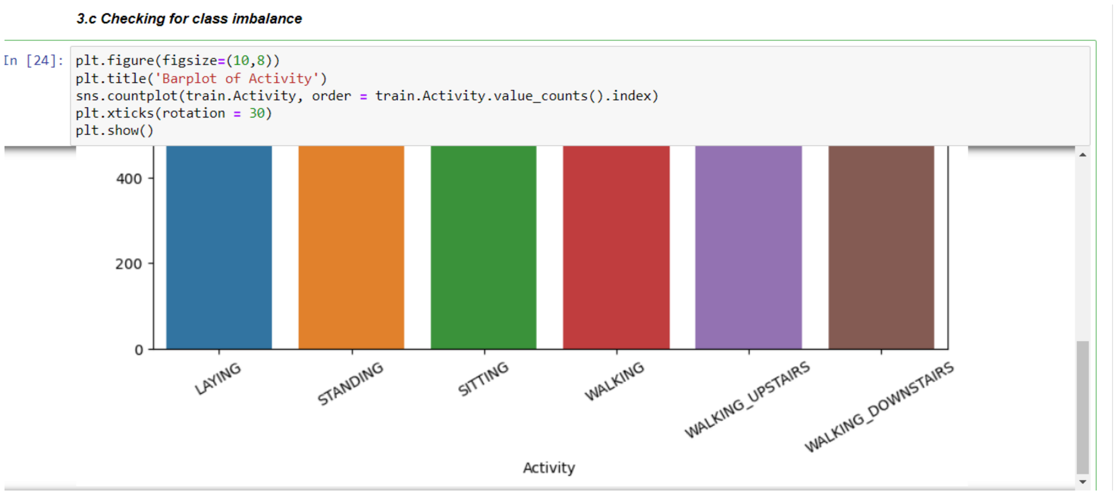
3.c Checking for class imbalance ¶

```
In [24]: plt.figure(figsize=(10,8))
         plt.title('Barplot of Activity')
         sns.countplot(train.Activity, order = train.Activity.value_counts().index)
         plt.xticks(rotation = 30)
         plt.show()
```



3.c Checking for class imbalance

```
In [24]: plt.figure(figsize=(10,8))
         plt.title('Barplot of Activity')
         sns.countplot(train.Activity, order = train.Activity.value_counts().index)
         plt.xticks(rotation = 30)
         plt.show()
```



3.c Checking for class imbalance

```
In [24]: plt.figure(figsize=(10,8))
         plt.title('Barplot of Activity')
         sns.countplot(train.Activity, order = train.Activity.value_counts().index)
         plt.xticks(rotation = 30)
         plt.show()
```

4. Exploratory Data Analysis

what features are there?

```
In [25]: train.head()
```

Out[25]:

	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-kurtosis()	angle(tBody
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.482845	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205	

5 rows × 563 columns

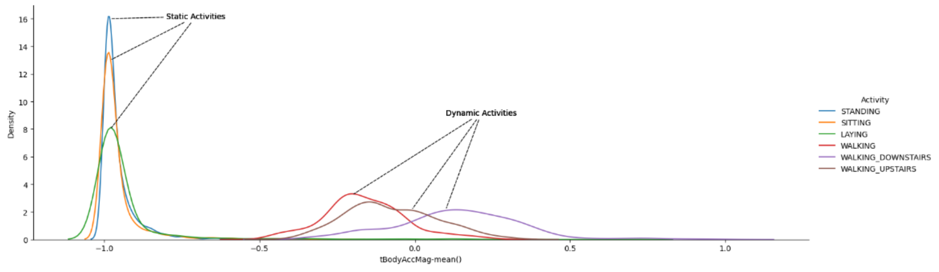
```
In [26]: pd.DataFrame.from_dict(Counter([col.split('-')[0].split('(')[0] for col in train.columns]),
                                orient = "index").rename(columns = {0:'count'}).sort_values('count', ascending=False)
```

Out[26]:

	count
fBodyAcc	79
fBodyGyro	79
fBodyAccJerk	79
tGravityAcc	40
tBodyAcc	40
tBodyGyroJerk	40

4.a Analysing tBodyAccMag-mean feature

```
In [27]: facetgrid = sns.FacetGrid(train, hue = 'Activity', height = 5, aspect = 3)
         facetgrid.map(sns.distplot, 'tBodyAccMag-mean()', hist = False).add_legend()

         plt.annotate("Static Activities", xy = (-.98, 8), xytext = (-.8, 16), arrowprops={'arrowstyle': '-', 'ls': 'dashed'})
         plt.annotate("Static Activities", xy = (-.98, 13), xytext = (-.8, 16), arrowprops={'arrowstyle': '-', 'ls': 'dashed'})
         plt.annotate("Static Activities", xy = (-.98, 16), xytext = (-.8, 16), arrowprops={'arrowstyle': '-', 'ls': 'dashed'})

         plt.annotate("Dynamic Activities", xy=(-0.2,3.25), xytext=(0.1, 9),arrowprops={'arrowstyle': '-', 'ls': 'dashed'})
         plt.annotate("Dynamic Activities", xy=(0.1,2.18), xytext=(0.1, 9),arrowprops={'arrowstyle': '-', 'ls': 'dashed'})
         plt.annotate("Dynamic Activities", xy=(-0.01,2.15), xytext=(0.1, 9),arrowprops={'arrowstyle': '-', 'ls': 'dashed'})

         plt.show()
```

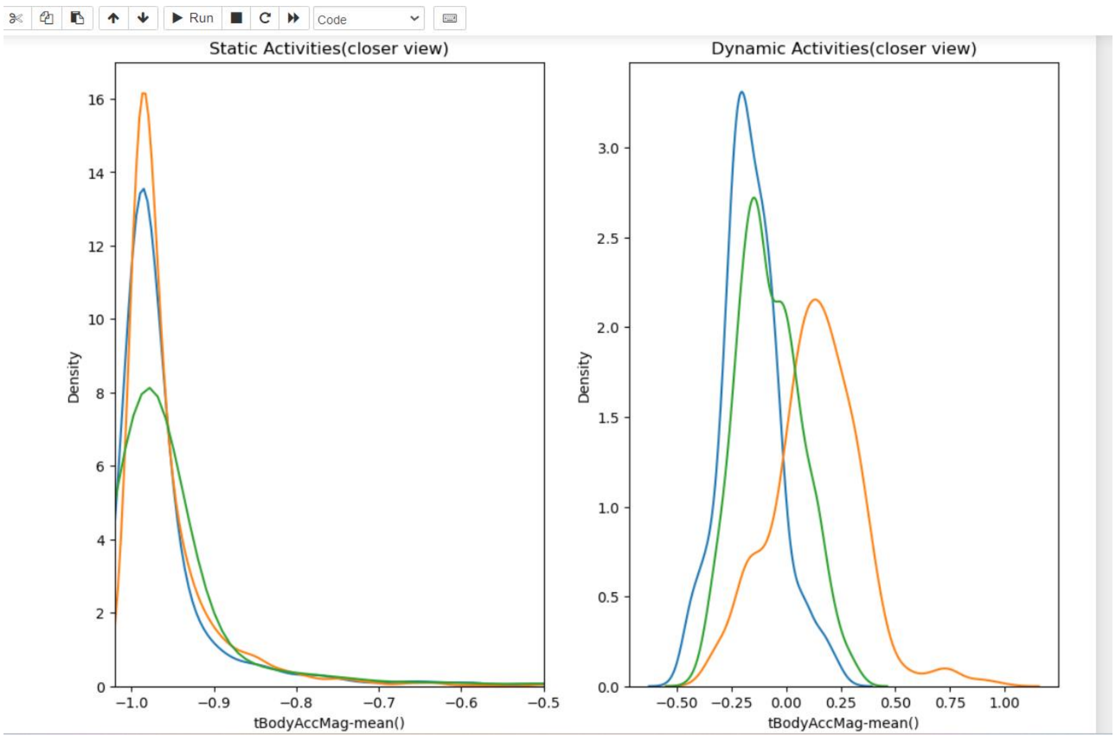


Using the above density plot we can easily come with a condition to seperate static activities from dynamic activities.

```
if(tBodyAccMag-mean()<=-0.5):
    Activity = "static"
else:
    Activity = "dynamic"
```

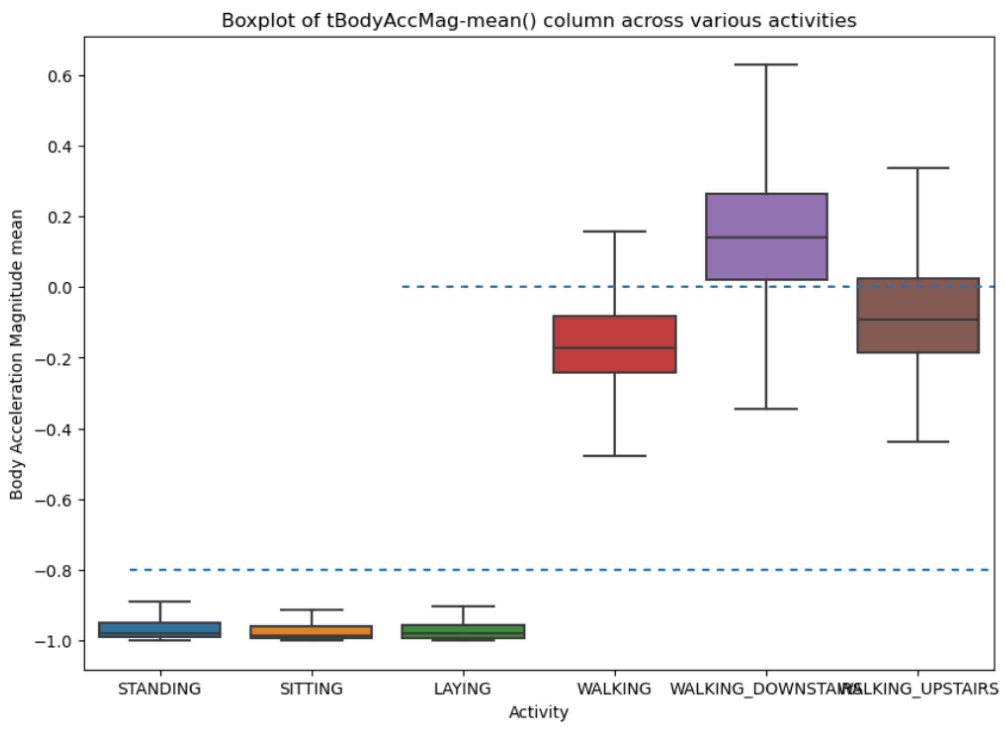
Let's have a more closer view on the PDFs of each activity under static and dynamic categorization.

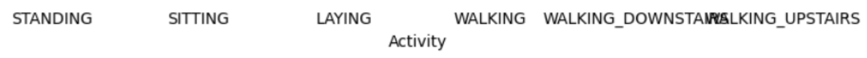
```
In [28]: plt.figure(figsize=(12,8))
         plt.subplot(1,2,1)
         plt.title("Static Activities(closer view)")
         sns.distplot(train[train["Activity"]=="SITTING"]['tBodyAccMag-mean()'],hist = False, label = 'Sitting')
         sns.distplot(train[train["Activity"]=="STANDING"]['tBodyAccMag-mean()'],hist = False,label = 'Standing')
         sns.distplot(train[train["Activity"]=="LAYING"]['tBodyAccMag-mean()'],hist = False, label = 'Laying')
         plt.axis([-1.02, -0.5, 0, 17])

         plt.subplot(1,2,2)
         plt.title("Dynamic Activities(closer view)")
         sns.distplot(train[train["Activity"]=="WALKING"]['tBodyAccMag-mean()'],hist = False, label = 'WALKING')
         sns.distplot(train[train["Activity"]=="WALKING_DOWNSTAIRS"]['tBodyAccMag-mean()'],hist = False,label = 'WALKING_DOWNSTAIRS')
         sns.distplot(train[train["Activity"]=="WALKING_UPSTAIRS"]['tBodyAccMag-mean()'],hist = False, label = 'WALKING_UPSTAIRS')
         plt.show()
```



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

```
In [29]: plt.figure(figsize=(10,7))
         sns.boxplot(x = "Activity", y="tBodyAccMag-mean()", data = train, showfliers = False)
         plt.ylabel('Body Acceleration Magnitude mean')
         plt.title("Boxplot of tBodyAccMag-mean() column across various activities")
         plt.axhline(y = -0.8, xmin = 0.05, dashes = (3,3))
         plt.axhline(y= 0.0, xmin = 0.35, dashes=(3,3))
         plt.show()
```





Using boxplot again we can come with conditions to seperate static activities from dynamic activities.

```
if(tBodyAccMag-mean()<=-0.8):
    Activity = "static"
if(tBodyAccMag-mean()>=-0.6):
    Activity = "dynamic"
```

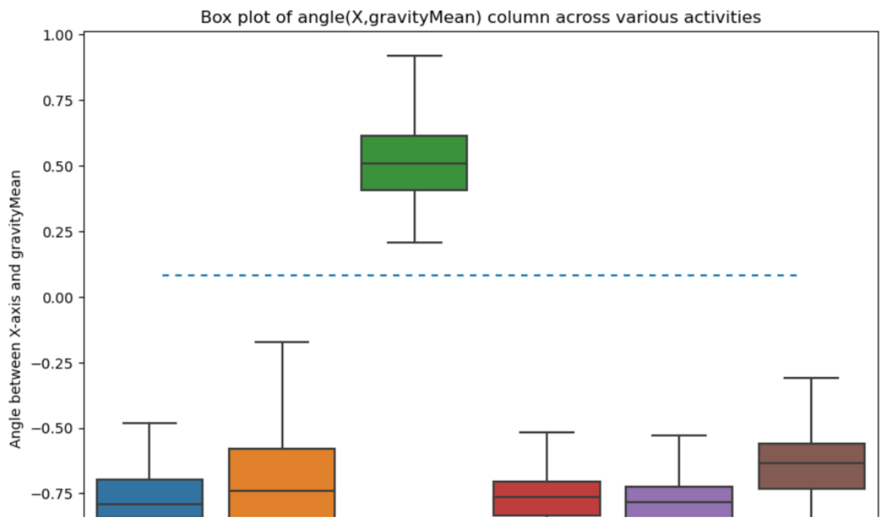
Also, we can easily seperate WALKING_DOWNSTAIRS activity from others using boxplot.

```
if(tBodyAccMag-mean()>0.02):
    Activity = "WALKING_DOWNSTAIRS"
else:
    Activity = "others"
```

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.

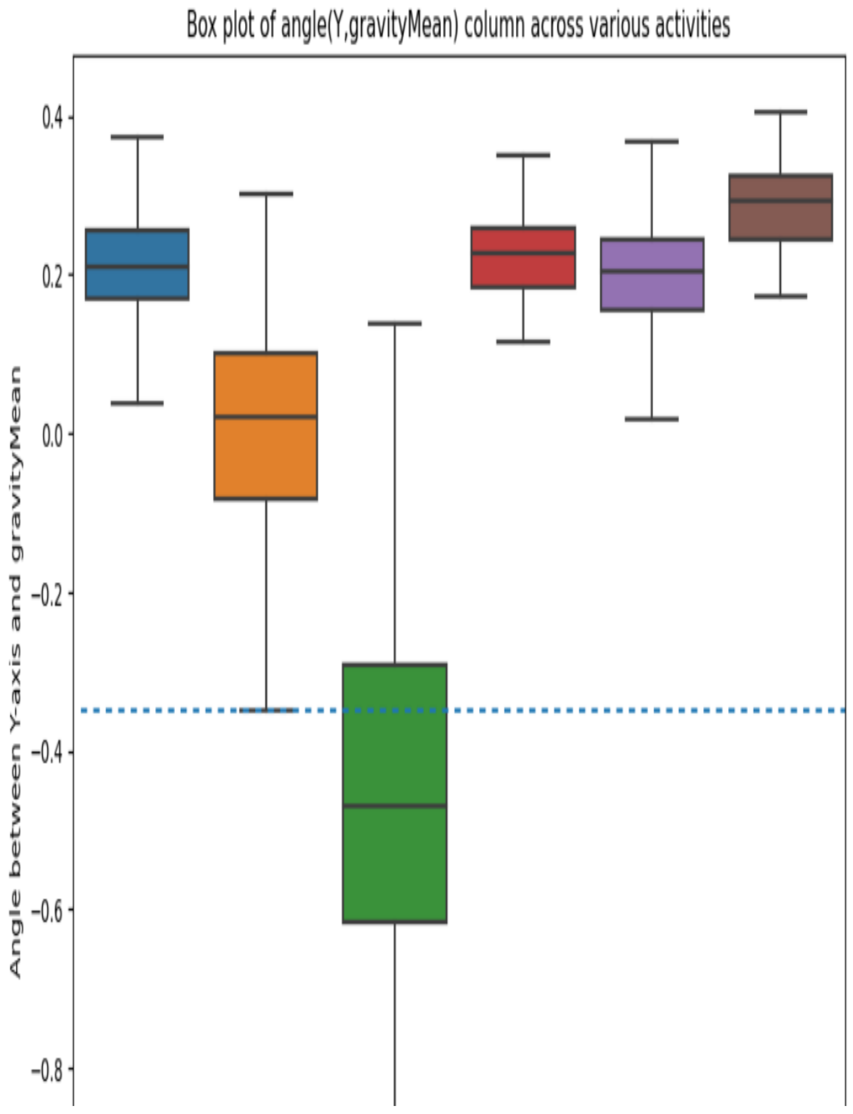
4.b Analysing Angle between X-axis and gravityMean feature

```
In [30]: plt.figure(figsize=(10,7))
         sns.boxplot(x='Activity', y='angle(X,gravityMean)', data=train, showfliers=False)
         plt.axhline(y=0.08, xmin=0.1, xmax=0.9,dashes=(3,3))
         plt.ylabel("Angle between X-axis and gravityMean")
         plt.title('Box plot of angle(X,gravityMean) column across various activities')
         plt.xticks(rotation = 30)
         plt.show()
```



4.c Analysing Angle between Y-axis and gravityMean feature

```
In [31]: plt.figure(figsize=(10,7))
         sns.boxplot(x='Activity', y='angle(Y,gravityMean)', data = train, showfliers=False)
         plt.ylabel("Angle between Y-axis and gravityMean")
         plt.title('Box plot of angle(Y,gravityMean) column across various activities')
         plt.xticks(rotation = 90)
         plt.axhline(y=-0.35, xmin=0.01, dashes=(3,3))
         plt.show()
```



4.d Visualizing data using PCA

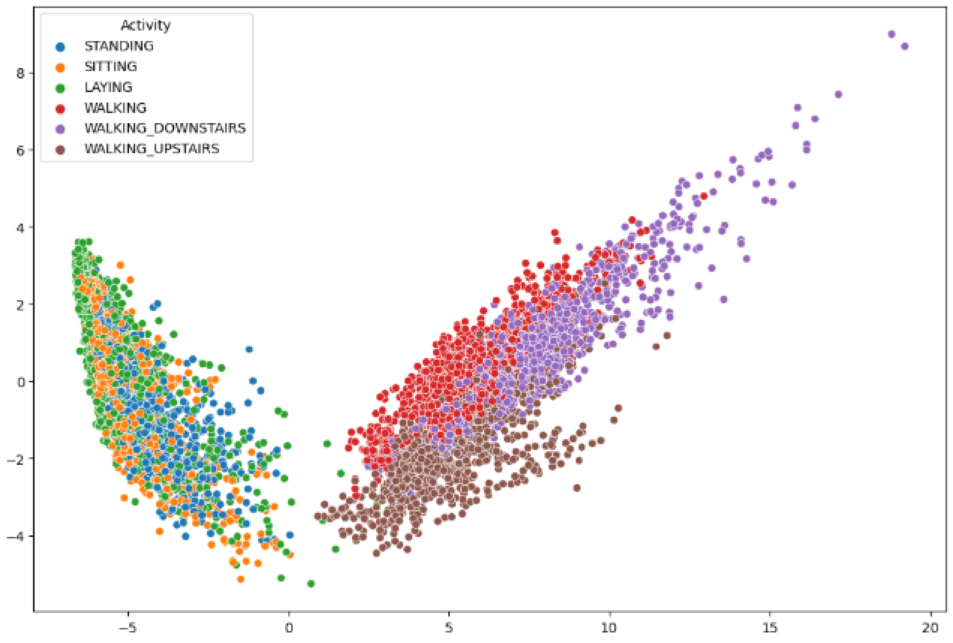
Using PCA data can be visualized from a extremely high dimensional space to a low dimensional space and still it retains lots of actual information. Given training data has 561 unqiue features, using PCA let's visualize it to a 2D space.

In [32]:
```
x_for_pca = train.drop(['subject', 'Activity'], axis = 1)
pca = PCA(n_components=2, random_state=0).fit_transform(x_for_pca)
```

In [33]:
```
pca
```

Out[33]:
```
array([[-5.5202803 , -0.29027701],
       [-5.53534954, -0.08253011],
       [-5.47498801,  0.28738703],
       ...,
       [ 5.85750527, -3.08184312],
       [ 5.42109482, -3.42643002],
       [ 5.49797027, -2.78992867]])
```

In [34]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(x = pca[:, 0], y = pca[:, 1], hue = train['Activity'])
plt.show()
```



Using the two new components obtained through PCA we can visualize and seperate all the six activities in a 2D space.

4.d Visualizing data using t-SNE

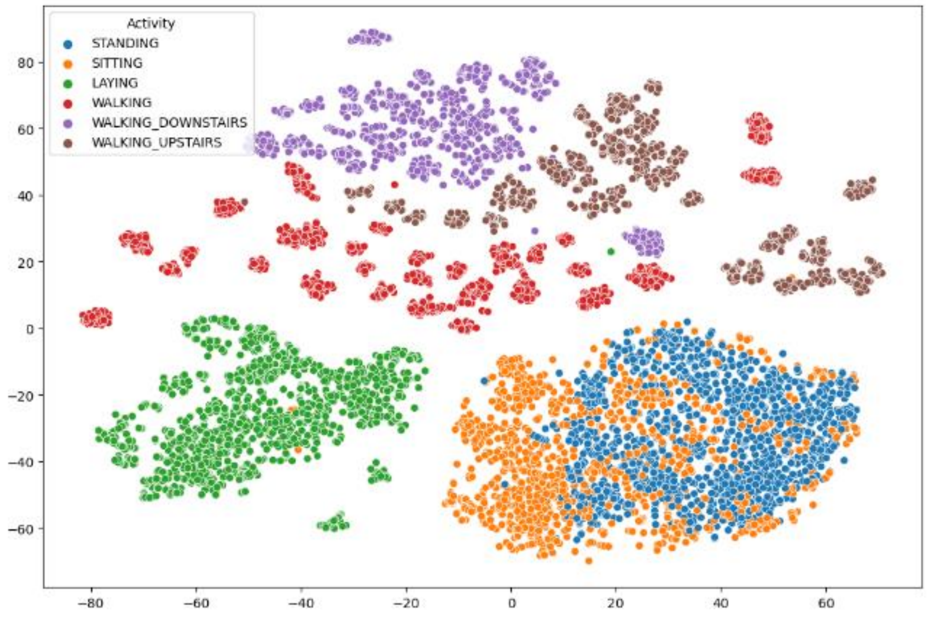
Using t-SNE data can be visualized from a extremely high dimensional space to a low dimensional space and still it retains lots of actual information. Given training data has 561 unqiue features, using t-SNE let's visualize it to a 2D space.

```
In [35]: x_for_tsne = train.drop(['subject', 'Activity'], axis = 1)
         tsne = TSNE(n_components=2, random_state=0, n_iter=1000).fit_transform(x_for_tsne)
```

```
In [36]: tsne
```

```
Out[36]: array([[ 63.4906   , -39.7206   ],
                [ 13.5708065, -39.372986 ],
                [ 16.88333  , -37.352337 ],
                ...,
                [ 60.73749  ,  14.081628 ],
                [ 60.48153  ,  14.081899 ],
                [ 59.754375 ,  14.62647  ]], dtype=float32)
```

```
In [37]: plt.figure(figsize=(12,8))
         sns.scatterplot(x = tsne[:, 0], y = tsne[:, 1], hue = train['Activity'])
         plt.show()
```

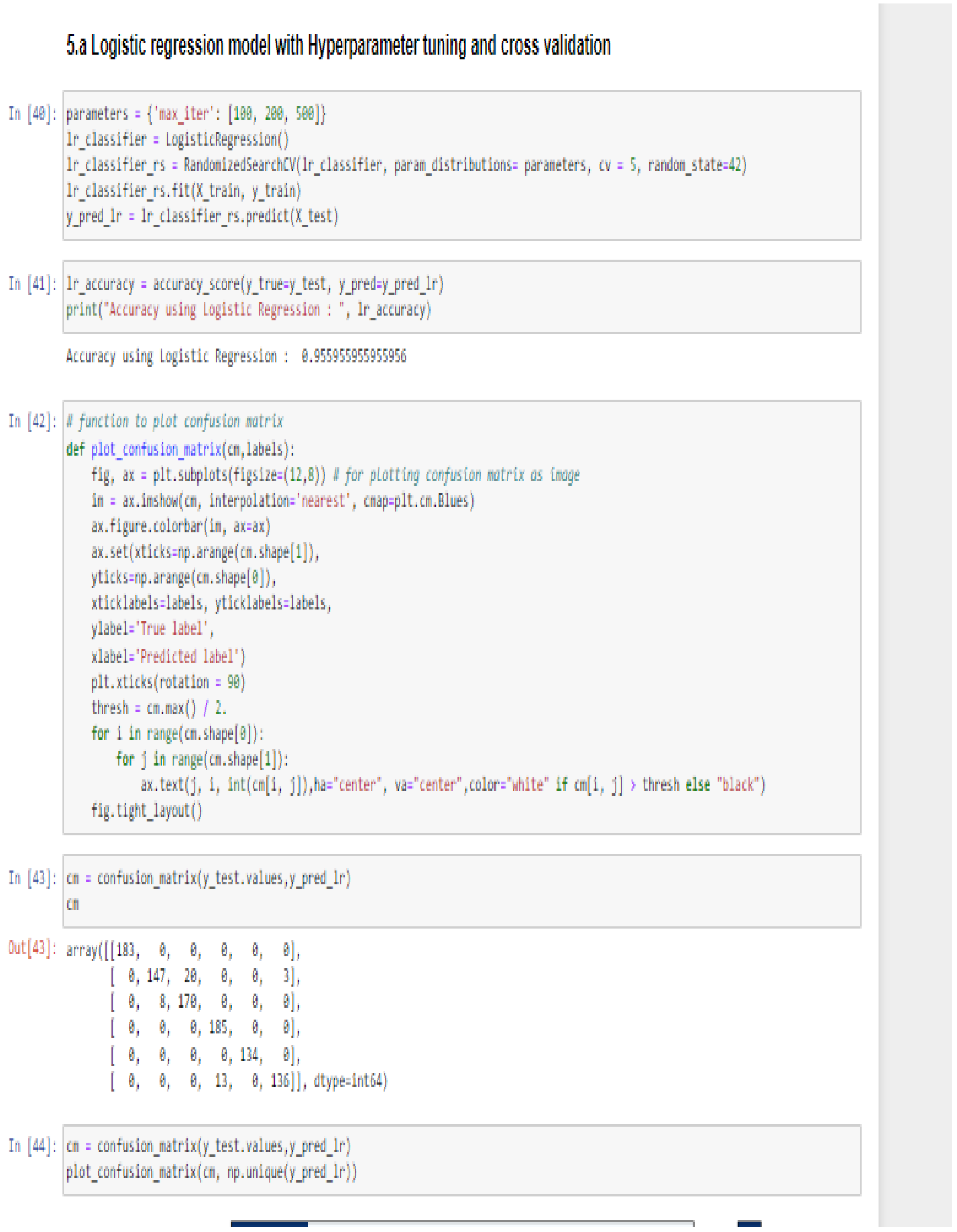


5. ML models

Getting training and test data ready

```
In [38]: X_train = train.drop(['subject', 'Activity'], axis = 1)
         y_train = train.Activity

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

```
In [39]: print('Training data size : ', X_train.shape)
         print('Test data size : ', X_test.shape)

         Training data size :  (7352, 561)
         Test data size :  (999, 561)
```



5.a Logistic regression model with Hyperparameter tuning and cross validation

In [40]:
```
parameters = {'max_iter': [100, 200, 500]}
lr_classifier = LogisticRegression()
lr_classifier_rs = RandomizedSearchCV(lr_classifier, param_distributions= parameters, cv = 5, random_state=42)
lr_classifier_rs.fit(X_train, y_train)
y_pred_lr = lr_classifier_rs.predict(X_test)
```

In [41]:
```
lr_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred_lr)
print("Accuracy using Logistic Regression : ", lr_accuracy)
```

```
Accuracy using Logistic Regression :  0.955955955955956
```

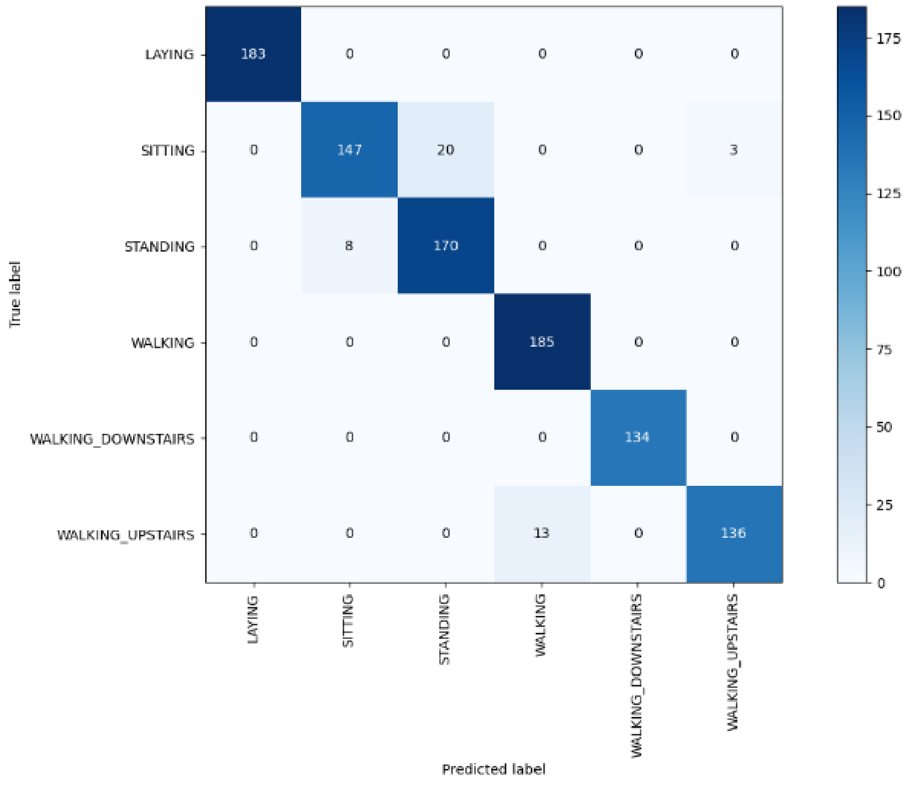
In [42]:
```
# function to plot confusion matrix
def plot_confusion_matrix(cm,labels):
    fig, ax = plt.subplots(figsize=(12,8)) # for plotting confusion matrix as image
    im = ax.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]),
    yticks=np.arange(cm.shape[0]),
    xticklabels=labels, yticklabels=labels,
    ylabel='True label',
    xlabel='Predicted label')
    plt.xticks(rotation = 90)
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, int(cm[i, j]),ha="center", va="center",color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
```

In [43]:
```
cm = confusion_matrix(y_test.values,y_pred_lr)
cm
```

Out[43]:
```
array([[183,   0,   0,   0,   0,   0],
       [  0, 147,  20,   0,   0,   3],
       [  0,   8, 170,   0,   0,   0],
       [  0,   0,   0, 185,   0,   0],
       [  0,   0,   0,   0, 134,   0],
       [  0,   0,   0,  13,   0, 136]], dtype=int64)
```

In [44]:
```
cm = confusion_matrix(y_test.values,y_pred_lr)
plot_confusion_matrix(cm, np.unique(y_pred_lr))
```

```
In [44]: cm = confusion_matrix(y_test.values,y_pred_lr)
         plot_confusion_matrix(cm, np.unique(y_pred_lr))
```



5.b Kernel SVM model with Hyperparameter tuning and cross validation

```
In [*]: parameters = {
            'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
            'C': [100, 50]
        }

        svm_rs = RandomizedSearchCV(SVC(), param_distributions=parameters, cv = 3, random_state=42)
        svm_rs.fit(X_train, y_train)
```

```
In [*]: y_pred = svm_rs.predict(X_test)
```

```
In [*]: kernel_svm_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
        print("Accuracy using Kernel SVM : ", kernel_svm_accuracy)
```

```
In [*]: cm = confusion_matrix(y_test.values,y_pred)
        plot_confusion_matrix(cm, np.unique(y_pred))
```

```
In [*]: get_best_randomsearch_results(svm_rs)
```

5.c Decision tree model with Hyperparameter tuning and cross validation

```
In [*]: parameters = {'max_depth': np.arange(2, 10, 2)}

        dt_classifier = DecisionTreeClassifier()
        dt_classifier_rs = RandomizedSearchCV(dt_classifier, param_distributions = parameters, random_state = 42)
        dt_classifier_rs.fit(X_train, y_train)
```

```
In [*]: y_pred = dt_classifier_rs.predict(X_test)
```

```
In [*]: dt_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
        print("Accuracy using Decision tree : ", dt_accuracy)
```

```
In [*]: cm = confusion_matrix(y_test.values,y_pred)
        plot_confusion_matrix(cm, np.unique(y_pred)) # plotting confusion matrix
```

```
In [*]: # getting best random search attributes
        get_best_randomsearch_results(dt_classifier_rs)
```

5.d Random forest model with Hyperparameter tuning and cross validation

In [ ]:
```python
parameters = {
    'n_estimators':np.arange(20, 101, 10),
    'max_depth': np.arange(2, 17, 2)
}
rf_classifier = RandomForestClassifier()
rf_classifier_rs = RandomizedSearchCV(rf_classifier, param_distributions=parameters,random_state = 42)
rf_classifier_rs.fit(X_train, y_train)
```

In [ ]:
```python
# getting best random search attributes
get_best_randomsearch_results(rf_classifier_rs)
```

In [ ]:
```python
y_pred = rf_classifier_rs.predict(X_test)
```

In [ ]:
```python
rf_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy using Random forest : ", rf_accuracy)
```

In [ ]:
```python
cm = confusion_matrix(y_test.values,y_pred)
plot_confusion_matrix(cm, np.unique(y_pred))
```

5.e Gradient Boosting Machines

In [6]:
```python
X = pd.read_csv('train.csv')
y = pd.read_csv('test.csv')

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the GBM classifier
gbm = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, random_state=42)

# Train the GBM classifier on the training data
gbm.fit(X_train, y_train)

# Predict activity labels for the test data
y_pred = gbm.predict(X_test)

# Evaluate the performance of the GBM classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generate a detailed classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

5.f K-Nearest Neighbours

In [ ]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report

# Load the HAR dataset (assuming 'X' contains features and 'y' contains labels)
# Replace 'X' and 'y' with your actual feature matrix and label array
X = pd.read_csv('test.csv')
y = pd.read_csv('train.csv')

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the KNN classifier
k = 5  # Number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)

# Train the KNN classifier on the training data
knn.fit(X_train, y_train)

# Predict activity labels for the test data
y_pred = knn.predict(X_test)

# Evaluate the performance of the KNN classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generate a detailed classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

CONCLUSION

Human Activity Recognition (HAR) has made significant strides in recent years, thanks to advancements in sensor technology, machine learning algorithms, and data processing techniques. HAR systems can accurately classify and interpret human activities based on data from various sensors, such as accelerometers, gyroscopes, and wearable devices.

These systems have found applications in diverse fields, including healthcare, sports, security, and human-computer interaction. They enable real-time monitoring of physical activities, health tracking, gesture recognition, and behavior analysis, among other functionalities.

The effectiveness of HAR systems depends on several factors, including the choice of sensors, feature extraction methods, classification algorithms, and data preprocessing techniques. Continuous research efforts are required to improve the accuracy, robustness, and scalability of HAR systems, especially in challenging environments with complex activities and noisy sensor data.

FUTURE SCOPE

Enhanced Accuracy: Future research will focus on improving the accuracy and reliability of HAR systems by exploring advanced machine learning algorithms, such as deep learning and ensemble methods. These approaches can capture complex temporal dependencies and spatial relationships in sensor data, leading to more robust activity recognition models.

Multimodal Fusion: Integrating data from multiple sensors (multimodal fusion) offers opportunities to enhance HAR systems' performance. Fusion techniques, including sensor-level fusion, feature-level fusion, and decision-level fusion, can leverage complementary information from different sensor modalities to improve activity recognition accuracy and robustness.

Context-awareness: HAR systems will evolve to incorporate contextual information, such as environmental factors, user preferences, and social interactions, to enhance activity recognition accuracy and relevance. Context-aware HAR models can adapt to changing conditions and user behaviors, leading to more personalized and adaptive applications.

Real-time Processing: Advancements in hardware and software technologies will enable HAR systems to perform real-time data processing and analysis, facilitating instant feedback and response in various applications, such as healthcare monitoring, sports coaching, and interactive systems.

Privacy and Security: As HAR systems become more pervasive, ensuring user privacy and data security will be crucial. Future research will focus on developing privacy-preserving HAR techniques that enable accurate activity recognition while minimizing the risks of data exposure and unauthorized access.

Ubiquitous Computing: With the proliferation of Internet of Things (IoT) devices and smart environments, HAR systems will be seamlessly integrated into everyday objects and surroundings, enabling pervasive and unobtrusive activity monitoring and assistance.

REFERENCES

Books:

"Human Activity Recognition: Using Wearable Sensors and Smartphones" by Ling Shao, Jean-Philippe Thiran, Diane Cook, Rosalind W. Picard.

Review Papers:

Lara, O., & Labrador, M. A. (2013). A survey on human activity recognition using wearable sensors. IEEE Communications Surveys & Tutorials, 15(3), 1192-1209.

Journals:

IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)

IEEE Transactions on Mobile Computing (TMC)

D. S. Antoniuk, T. A. Vakaliuk, V. V. Didkivskyi, O. Y. Vizgalov, Necessity of the personal finance management simulation development, Innovative pedagogy 2 (2020) 208–212. doi:10.32843/2663-6085/2020/24-2.41.

G. Kaplan, G. L. Violante, A model of the consumption response to fiscal stimulus payments, Econometrica 82 (2014) 1199–1239. doi:10.3982/ECTA10528.

A. Olafsson, M. Pagel, The Liquid Hand-to-Mouth: Evidence from Personal Finance Management Software, The Review of Financial Studies 31 (2018) 4398– 4446. doi:10. 1093/rfs/hhy055.

D. F. Costa, F. d. M. Carvalho, B. C. d. M. Moreira, Behavioral economics and behavioral finance: A bibliometric analysis of the scientific fields, Journal of Economic Surveys 33 (2019) 3–24. doi:10.1111/joes.12262.

S. Palan, GIMS — Software for asset market experiments, Journal of Behavioral and Experimental Finance 5 (2015) 1–14. doi:10.1016/j.jbef.2015.02.001.

Checklist for Dissertation-III Supervisor

Name: ____________________________ UID: ________ Domain: _____
Registration No: ____________Name of student:__________________

Title of Dissertation:
____

- ☐ Front pages are as per the format.
- ☐ Topic on the PAC form and title page are same.
- ☐ Front page numbers are in roman and for report, it is like 1, 2, 3……
- ☐ TOC, List of Figures, etc. are matching with the actual page numbers in the report.
- ☐ Font, Font Size, Margins, line Spacing, Alignment, etc. are as per the guidelines.
- ☐ Color prints are used for images and implementation snapshots.
- ☐ Captions and citations are provided for all the figures, tables etc. and are numbered and center aligned.
- ☐ All the equations used in the report are numbered.
- ☐ Citations are provided for all the references.
- ☐ **Objectives are clearly defined.**
- ☐ Minimum total number of pages of report is 50.
- ☐ Minimum references in report are 30.

Here by, I declare that I had verified the above mentioned points in the final dissertation report.

Signature of Supervisor with UID