Human Activity Recognition Using Smartphones

Jyothi Yendamuri
Faculty of Engineering, Environment and Computing,
*MSc Data Science and Computational Intelligence (*EECT044*)*
Coventry University, United Kingdom
yendamurij@uni.coventry.ac.uk

Abstract: Human Activity Recognition is the predominantly classifying activity of a person utilizing responsive sensors that are synchronized with human actions. In general, users while carrying their mobiles a contextual data is processed simultaneously to understand the human activities. These realities make Human Activity Recognition is more remarkable and popular. Currently, smart mobiles are being manufactured with various in-built detecting sensors like gyroscope, accelerometer, GPS sensor and compass sensor. A device can be structured to get the condition of the user.. In this paper different Machine Learning techniques will be applied to data set accordingly. The data set consist of output values of Sensor signal (accelerometer and gyroscope) from smartphone collected 30 volunteers whose age ranges from 19 to 48. Everyone carryout six particular activities like "strolling or walking, strolling upstairs, walking downstairs, standing,lying,sitting". In this report, three different classification techniques will be employed by implementing and comparing a Decision tree, K-nearest algorithm and Support Vector Machine (SVM) algorithm.The acquired dataset was orbitarily split or seaparate into training dataset and testing dataset 70% and 30% respectively.

Keywords: DecisionTree, Support vector machine, K-Nearest Neighbour, UCI, Accuracy, Machine Learning.

I. INTRODUCTION

These days smart mobiles are turned into an expanding number of famous in human every day ways of life. The vast majority utilized for searching for data, playing games and having access to the social community, however there have been numerous helpful studies on smartphones. Activity Recognition (AR) is one of the most extreme innovation behind many packages on smartphone comprising of health tracking,human survey gadget, fall detection and home automation and many others. Smartphone action acknowledgment system is an energetic area of studies since they could prompt new sorts of keen applications. The combination of those shrewd devices in our everyday existence is quickly developing. In this paper, we utilize a Smartphone for HAR with ability programs in assisted living innovation[1].

Human Activities might look very simple to any person visually, but when it comes to machine it has been very difficult and challenging ever since. This has been very difficult of machine to understand human activity because any human activity ever the simplest ones like standing or walking might look very simple to human eye but practically they consist of contribution of different parts of the body which are performed involuntarily by the body. Standing might look very simply but it consists of Strength, balance, stability and orientation of the body. Since every human being have their own way of performing this activity and there is no standard format to do such things. Since due to the enormous variation of human activity is has been very challenging of machine to understand these. To overcome this problem, there would be huge number of applications for this implementation[2].

The technology is being advanced day by day we are making progress in this field. The Microelectromechanical systems (MEMS) and Inertial sensors are now we can see that in a lot of products like smart phone, smartwatch and fitness trackers.

In this paper dataset was obtained from the UCI repository and we collected a data of 30 volunteers doing 6 different normal human activities. We used three different classification techniques will be employed by implementing and comparing a Decision tree, K-nearest algorithm and Support Vector Machine (SVM) algorithm. The aim or and day by day exercises completed by an individual given a bunch of perceptions and the encompassing environment[1].objective of HAR is to identify vdifferent

II. THE DATA SET

Straight forwardly open accessible UCI HAR information base has been utilized in this examination This data was recorded by Jorge L. Reyes-Ortiz and his colleges at University degli

Studi di Genova, Genova, Italy. To obtain this data an experiment has carried out with a group of 30 persons. All 30 volunteers are whose age ranges from 19 to 48. Every one should perform six distinct tasks like "**strolling, strolling upstairs walking upstairs, walking downstairs, sitting, lying, standing**". Each and every volunteer were wearing a smartphone on their waist. They used the embedded Accelerometer and gyroscope sensors to capture three axial acceleration and three axial angular velocity at a constant rate of 50Hz. The whole experiment visual was recorded using a video camera. Which further was used to label the data manually. The resulting data set further split into two parts testing and training data set 30% and 70% respectively[4]

III. SENSORS IN SMARTPHONES

Android devices have worked in sensors that estimates movement (accelerometers, gravity sensors, and gyrator), direction (magnetometers), and different environmental conditions (indicators, photometers, and thermometers).
Equipment based sensors are actual parts incorporated or fused with a PDA and they surmise their rough information by specifically assessing perticular natural parameters, for example, increasing speed, geomagnetic field quality, or exact change[1].

1. *Accelerometers:* An accelerometer is one of the vital equipment in the smart mobiles and smart wearable devices.The primary capacity of this is to detect the changes in the direction of smart device regarding datum and modify the direction to suits the overview edge of the user.For example, when you are looking for website page with expanded width, you can get this scene to see from changing the direction of phone to level. Exceptional way camera mode additionally changes the landscape or scene to representation or portrait to landscape when we change the direction of device/camera.

2. *Compass Sensor:* The smart Compass is a standard device to distinguish the heading as for the north-south shaft of the earth's magnetic field.
Compass usefulness in phones is regularly founded on a more complex sensor called a magnetometer; it is used to evaluate the quality and course of attractive fields.

3. *Gyroscope:* The utilization of Gyroscope is to keep up and control the position, level or direction dependent on the standard of angular momentum.
Whenever 'Gyros' used close by accelerometer recognizes or detects movement from 6-Axis for example right, left, up, down, forward and in reverse. It similarly distinguishes the move, pitch, and yaw movements.

4. *GPS (Global Positioning System) Sensor:* Global Positioning System(GPS) is a framework which tracks the objective or 'explores' the things by picture or map with the help of GPS satellites. iPhone series, HTC android mobiles, Samsung Galaxy mobile phones, Sony, Nokia Lumia, and various more smartphones reinforce GLONASS (Globalnaya Navigatsionnaya Sputnikovaya Sistema) GPS structure for route highlight.

How Data Was Recorded Using Sensors

By utilizing the sensors(Gyroscope and Accelerometer) in a mobilephone, selected people have caught or captured '3-axial linear acceleration'(tAcc-XYZ) from accelerometer and '3-axial angular velocity' (tGyro-XYZ) from Gyroscope with a several variations.

IV. DATA ANALYSIS

The dataset was changed over in a CSV file and imported into a pandas information out line frame. The just preprocessing required was to consolidate the subjects and movement segments or columns to their perticular or respective windowed features. Upon investigation, it was tracked down that the dataset didn't have any missing qualities. The informational index was investigated further to comprehend the different highlights and their impacts on the exercises. The 6 exercises are as follows[5].

Table. 1

List of activities performed

Activity	Activity Labels
1	Walking
2	Walking upstairs
3	Walking downstairs
4	Sitting
5	Standing
6	Laying

The training dataset has a sum of 7352 perceptions or windows of information. It's anything but a sum of 563 time and recurrence highlights where every perception compares to one of the 6 wandering class exercises.
We additionally have the ID of an individual volunteer for every record. A sum of 10299 records and train and test dataset parts are 70% and 30%. The 6 classes are changed over to numerical or mathmetical values sequentially: (1, 2, 3, 4, 5, 6).
Beneath figure-1shows the distinctive human exercises with their count in the given training dataset.Number of data points are shown for each activity.

Attribute Representation:

Human Activity Recognition utilizing multichanel time-arrangement signals and having presented the the attribute representation and attribute based organizations.dataset don't have explained quality. Their annotations are connected with specific coarse activitys: walk,jump,run.

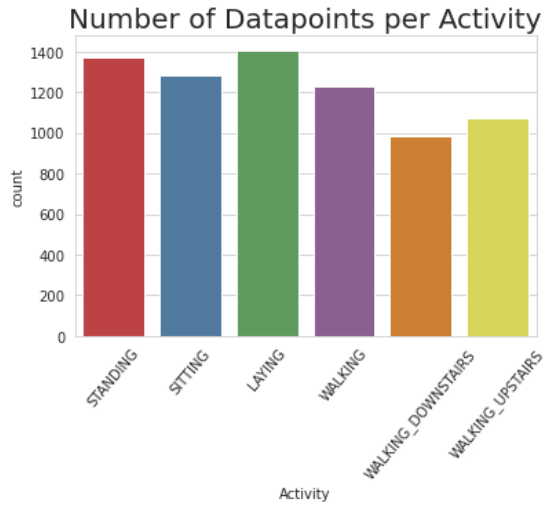


Figure 1: Human activities with their count in training dataset.

Below figure shows the dissemination of two sorts of action (static:standing, sitting and laying, dynamic:strolling or walking, walking downstairs, and walking upstairs) utilizing tBodyAccMag-mean() highlight, which is taken from the accelerometer of phones. Obiviously the two kinds of exercises are successfully detachable or distinct. There are no absent and copy values[4].

Output:<AxesSubplot:title={'center':'Dynamic Activities(closerview)'},xlabel='tBodyAccMagmean', ylabel='Density'>

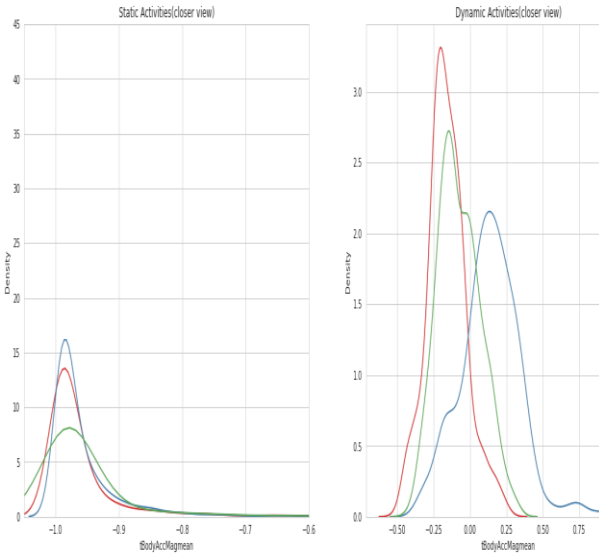


Figure 2: Static Activity and Dynamic Activity

V. CLASSIFICATION TECHNIQUES:

In This paper three different clasification techniques will be employed by implementing and comparing a Decision Tree,K-nearest algortithm and support vector machine(SVM) algorithm.

Decision Tree:

A Decision tree is a well known algorithm that generally used for classification.It is a graphical representation of all the possible solutions to a decision and the decisions are based on specific conditions.It is a tree-structured classifier,where the inner nodes represent the features of a dataset,leaf node represent the outcome and branches represent the decision rules. It separates instances by arranging or sorting from root to leaves. During traversing through each segment of node the algorithm must make a decision which was made depending on certain condition. These conditions of each node segment were set during the training of the model. Here, we check degradation utilizing Gini and gain data from entropy. These actions the nature of the split. We utilize different max profundities (0 to 8), which is the greatest profundity of the tree[7].

Root Node: Root node is from where the decision tree begins. It addresses the whole dataset, which remoter gets separated or split into at least two homogeneous sets.

Leaf Node: Leaf nodes are the last output node, and the tree cannot be divide further after getting a leaf node.

Branch Node: A tree formed by splitting the tree.

Pruning: Pruning is the process of eliminating the unwanted branches from the tree.

Support Vector Machine

Support Vector Machine is quite possibly the most famous Supervised Learning calculations, which is basically utilized for classification problems in machine learning.SVM is usually used to pre-process any features of data is non-linearly separable, then the support vector machine algorithm outputs a higher dimensional space in which the non-linearly spreadable data can be linearly separable[4].

The main aim of SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can

stright forward put the new data or information point in the right category in the later. This most and best decision boundary is called a hyperplane.SVM algorithm can be utilized for Face recognition, classification, text category, etc.It can be divided of two distinct types[8].

1. Linear SVM
2. Non-linear SVM

*1.**Linear SVM:***Used for directly seaparable information if a information set can be portrayed into two different classes by utilizing a solitary straight line, then, at that point such information is called as directly detachable information.

*2.**Non-linear SVM**:* Utilized for Non-directly seaparated data, If dataset can't be described by utilizing direct line.The simplest implementation is that the hard margin SVM during which data must be linearly separable to permit the algorithm to converge. $L = \{v: \langle w, v\rangle + b = 0\}, \|w\| = 1$

The distance of a point from the hyperplane L can be evaluated in that way: $d(x,L) = |\langle w,x\rangle + b|$

Gaussian kernel and linear kernel are used generally used to classify the data set and then the accuracy is measured. Since in most cases linear kernel performs much better.

K-Nearest Neighbours(KNN)

K-Nearest Neighbor is one of the simple Machine Learning algorithms dependent on Supervised Learning technique.The KNN algorithm also called a lazy learner that simply memorizes the training data and learns how to appropriately classify each test instance when it is presented. K-NN algorithm can be utilized for Regression as well as for Classification however for the most part it is utilized for the Classification issues. It is a non-parametric algorithm, which implies it does not make any assumption on hidden data.The most common value among the k number of instances is chosen as a decider for any new classification instance. Due to this nature of k-nearest Neighbours classification algorithm is very sensitive to local structure of data[6]. Here the quantity of the nearest neighbour is very important in his classification algorithm. As there is no specific mathematical formula to calculate the value of K. so when using k-nearest neighbour classification algorithm we have to experiment with random values to choose the value of ‘k’ and check at what value of k we are getting better classification of result or accuracy[7].

Use Of KNN Algorithm:

Assume there are two classes, i.e., Category A and Category B, and

classifications.to deal with this type of issue, we need a K-NN algorithm. With the assistance of K-NN, we can without much of a stretch perceive the classification or class of a particular dataset.

VI. EXPERIMENTAL METHODS

A. Data Preprocessing:

The original data was submitted to the UCI Machine Learning Repository inside a folder called UCI HAR Dataset Data. This data set consist separate folder file for both training data and testing data.

Now the test folder of 30 intrests(volunteers) information is alternatively parted to 70% test and 30%ttrain information.

Each data point compares these 6 Activities.

Now both training and test data consist of a subfolder of inertial Signal and three files of Subject test, X test and y test. 3 axial gyroscope data and 3 axial total acceleration . here 3 axial mean X,Y, and Zaxis.

Data was recorded by using below steps:

By using the sensors(Gyroscope and accelerometer) in a smartphone, they have captured '3-axial linear or direct acceleration'(tAcc-XYZ) from accelerometer and '3-axial angular velocity' (tGyro-XYZ) from Gyroscope with several varieties[2].

Pre-processing Steps:

1. Pre-processing and gyroscope utilizing noise filters.
2. Splitting information into fixed windows of 2.56 seconds (128 data points) with 50% overlap.
3. Splitting of accelerometer data into gravitational (total) and body movement segments.

The UCI HAR Dataset includes the following files:

- 'README text file
- 'features information.text.': it Shows information about the variables used on the feature vector.
- 'features.txt': List of all activitys
- 'activity_labels.txt': Links the class labels with their activity name.
- Train Data:
  'Dataset X,train.text file
  Dataset subject,train.txt'dataset,y_train.txt'
- Test Data
  Dataset X_test.text file
  dataset subject_test.text file
  'UCI_HAR_dataset subject_train..text file

B.Implementation:

Initially all library files and dataset has been imported successfully.Then obtain the train and test data sets.used different techniques and checked whether data has any null values. Finally machine learning algorithms are used to find out the best accuracy.

EXPERIMENT RESULTS

Now we are going to use the classification technic described above will be use to classify the data set recorded frorm the smart phone by 30 different volunteers doing 6 different tasks. For each volunteer the data recorder and a sample ate of 50hz with 2.56 sec sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window).
The total number of instances are 10299 with the total number of attributes at 561. Both SVM model and knn model are generated and trained on the training data which is around 70 percentage of the original data.The models were generated using a virtual studio (Python3), on a desktop computer with an Intel Core i7@ 3.30GHz and 16.00 GBs of RAM, running on a Windows 10 operating system.
however SVM model which provides the most expected accuracy . Below tables showing the both training and testing data.

Table. 2 Training Data

Laying	1407
Standing	1374
Sitting	1286
Walking	1226
Walking_Upstairs	1073
Walking_Downstairs	986
Name: Activity, dtype:	int64

Table 3. Testing Data

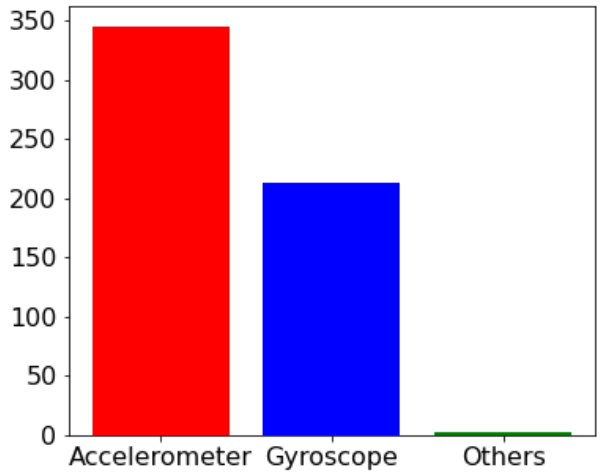
Laying	537
Standing	532
Sitting	496
Walking	491
Walking_Upstairs	471
Walking_Downstairs	420
Name:Activity,dtype:	Int64

Table 4. Training and Testing Data

Class	Count	Percentage
Laying	1944	18.876
Standing	1906	18.507
Walking	1722	16.72
Sitting	1777	17.254
Walking_Upstairs	1544	14.992
Walking_Downstairs	1406	13.652

Percentage of both train and test data

Though the fact that the perceptions for every activity are notperticularly equal the data set. It gives an even circulation.spreding of the activity observations. even after the division of
the data into the training and testing dataset, t.he equilibrium inanalysis in perception remains constant.
The below chart shows the activity count for different activities(Accuracy,Gyroscope,Others)



Here Accelerometer primary capacity of this is to find or detect the changes in the direction of smart device regarding datum and modify the direction to suits the overview edge of the user.
Gyroscope will keep up and control the position, level or direction dependent on the standard of angular momentum.

Model Building and Checking Accuracy

I have created a function to calculate accuracy of different models and print y_predict of values. Used 3 machine learning classifiers and their accuracy as shown below.
1.Decision Tree
2.Support Vector Machine

3.K-Nearest Neighbour

Evalution Metrics:

Accuracy can be determined by number of right forecasts at whole number of predictions.
This estimation is crucial as we need to know whether a human is strolling or sitting perfectly. It is the essential measurement to contrast between our models.
In condition 1, we get an precise score using below formula

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy of 3 classifiers:

Table 5. Classifiers

ACCURACY	
Support Vector Machine	95.04580929759076%
Decision Tree	85.78215134034612%
K-Neares Neighbour	90.02375296912113%

Benefits of Human Activity Recognition

Activity recognition is the reason for the improvement of many pottential possible applications in health ,wellness and sports.
1.Monitor Health: Examine the movement of a person from the information gathered by various devices.
2.Detect The Activity Patterns : Find which are the factors that figure out which activity is doing a person.
3.Find Activity: Determine a predictive model that can recognize a person's activity from the signals got by the sensors.
4.***Improve Wellbeing:*** Configuration individualized exercise tables to improve the health of a person[9].

Conclusion

Finally SVM performs better than the Decision Tree and K-Nearest Neighbour algorithms Though K-nearest algorithm is closer in accuracy. In this research paper, the overall design used to create human activity recognition framework and design issues like choice of sensors. Although the accurate efficiency of the model can be comparing with the amount of training time taken for the accuracy. As training time effect the cost of the model. Since KNN required more time to train and perform much worse that other models. From this experiment we can conclude that the lazy learner require to calculate the Euclidian distance from new instances for all the training instances its perform the best compare to any other model. In this experiment, it was tracked down that simple machine learning calculations can do well with suitable or appropriate boundary tuning, and we can determine the importance of the outcome by the testing.

VII. Appendix

Below google drive link contain the entire python code:

https://drive.google.com/file/d/1kp4YfvyfwTqlR6nJMTT2JNcYGna3lJx1/view?usp=sharing

References

[1]. Sandeep kumarpolu, International Journal for Innovative Research in Science & Technology, November 2018, Human Activity Recognition on Smartphones using Machine Learning Algorithms.

[2]. Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.

[3] Abheeshth Mishra,EDA of Human Activity Recognition

[4] Jakaria Rabbi, March 2021, Human Activity research and Recognition from Smartmobiles utilizing Machine Learning Algorithms.

[5] Mustafa Badshah, Dr-Robert Chun, May 2019, Sensor-Based Human Activity Recognition Using Smartphones.

[6] Erhan Bulbul, October 2018, Human Activity Recognition Using Smartphone.

[7] DecisionTree using machine learning algorithm https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm, https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

[8] Support vector machine https://scikit-learn.org/stable/modules/svm.html,https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[9] Human Activity Recognition(HAR) using machine learning Algorithms https://www.neuraldesigner.com/solutions/activity-recognition

Appendices

Fig 1.Import all required libraries

```
In [2]: #Load Libraries
import pandas as pd
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

Fig 2.Load the train data

```
In [3]: train=pd.read_csv("train.csv")
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(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

Fig 3.Load the test data

```
In [3]: import pandas as pd
test=pd.read_csv("test.csv")
test.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()	an
0	0.257178	-0.023285	-0.014654	-0.938404	-0.920091	-0.667683	-0.952501	-0.925249	-0.674302	-0.894088	...	-0.705974	
1	0.286027	-0.013163	-0.119083	-0.975415	-0.967458	-0.944958	-0.986799	-0.968401	-0.945823	-0.894088	...	-0.594944	
2	0.275485	-0.026050	-0.118152	-0.993819	-0.969926	-0.962748	-0.994403	-0.970735	-0.963483	-0.939260	...	-0.640736	
3	0.270298	-0.032614	-0.117520	-0.994743	-0.973268	-0.967091	-0.995274	-0.974471	-0.968897	-0.938610	...	-0.736124	
4	0.274833	-0.027848	-0.129527	-0.993852	-0.967445	-0.978295	-0.994111	-0.965953	-0.977346	-0.938610	...	-0.846595	

5 rows × 563 columns

Fig 4.Shape of the test data

```
In [14]: test.shape
Out[14]: (2947, 563)
```

Fig 5.Checking the null values and shape of the test data

```
n [11]: # For training data
print("Training Data: {}".format(training_data.shape))
print("Null values present in training data: {}".format(training_data.isnull().values.any()))

# For testing data
print("Testing Data: {}".format(testing_data.shape))
print("Null values present in testing data: {}".format(testing_data.isnull().values.any()))
```

```
Training Data: (7352, 563)
Null values present in training data: False
Testing Data: (2947, 563)
Null values present in testing data: False
```

Fig 6. Check the Duplicate values of both train and Test data

```
: print('No of duplicates in train : {}'.format(sum(train.duplicated())))
print('No of duplicates in test : {}'.format(sum(test.duplicated())))
```

```
No of duplicates in train : 0
No of duplicates in test : 0
```

Fig 7. Plot the moving Activities

```
In [18]: sns.set_palette("Set1", desat=0.80)
facetgrid = sns.FacetGrid(train, hue='Activity', size=6,aspect=2)
facetgrid.map(sns.distplot,'tBodyAccMagmean', hist=False)\
    .add_legend()
plt.annotate("Stationary Activities", xy=(-0.956,17), xytext=(-0.9, 23), size=20,\
             va='center', ha='left',\
             arrowprops=dict(arrowstyle="simple",connectionstyle="arc3,rad=0.1"))

plt.annotate("Moving Activities", xy=(0,3), xytext=(0.2, 9), size=20,\
             va='center', ha='left',\
             arrowprops=dict(arrowstyle="simple",connectionstyle="arc3,rad=0.1"))
plt.show()
```

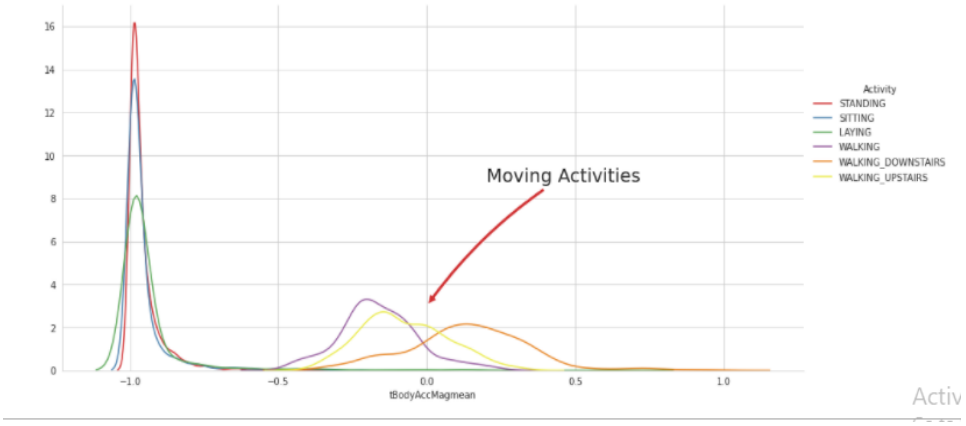


Fig 8. Check the Static and Dynamic Activities

```
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]



plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.title("Static Activities(closer view)")
sns.distplot(train[train["Activity"]=="SITTING"]['tBodyAccMagmean'],hist = False, label = 'Sitting')
sns.distplot(train[train["Activity"]=="STANDING"]['tBodyAccMagmean'],hist = False,label = 'Standing')
sns.distplot(train[train["Activity"]=="LAYING"]['tBodyAccMagmean'],hist = False, label = 'Laying')
plt.axis([-1.05, -0.6, 0, 45])
plt.subplot(1,2,2)
plt.title("Dynamic Activities(closer view)")
sns.distplot(train[train["Activity"]=="WALKING"]['tBodyAccMagmean'],hist = False, label = 'Sitting')
sns.distplot(train[train["Activity"]=="WALKING_DOWNSTAIRS"]['tBodyAccMagmean'],hist = False,label = 'Standing')
sns.distplot(train[train["Activity"]=="WALKING_UPSTAIRS"]['tBodyAccMagmean'],hist = False, label = 'Laying')
```

```
<AxesSubplot:title={'center':'Dynamic Activities(closer view)'}, xlabel='tBodyAccMagmean', ylabel='Density'>
```

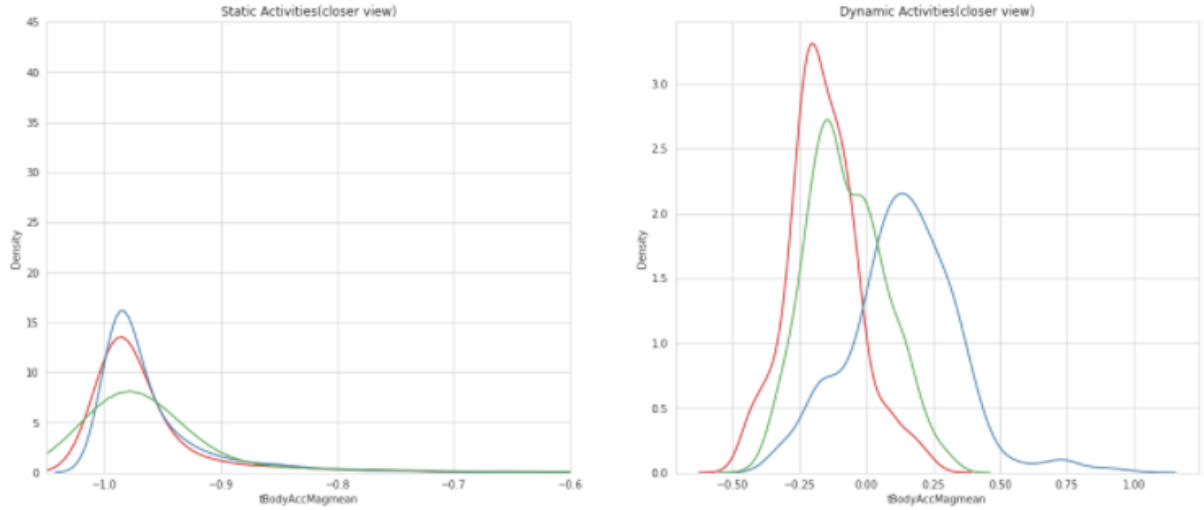


Fig 9. Train And Test Activity Count

```
In [21]: train.Activity.value_counts()

Out[21]: LAYING                1407
         STANDING              1374
         SITTING               1286
         WALKING               1226
         WALKING_UPSTAIRS      1073
         WALKING_DOWNSTAIRS     986
         Name: Activity, dtype: int64

In [22]: test.Activity.value_counts()

Out[22]: LAYING                537
         STANDING              532
         WALKING               496
         SITTING               491
         WALKING_UPSTAIRS      471
         WALKING_DOWNSTAIRS    420
         Name: Activity, dtype: int64
```

Fig 10. Count the number of records for each activity

```
# Count the number of records for each activity
count_of_each_activity = np.array(y_train.value_counts())

# Identify all the unqiue activities and in sorted order
activities = sorted(y_train.unique())

# Plot a pie chart for different activities
plt.rcParams.update({'figure.figsize': [5, 5], 'font.size': 10})
plt.pie(count_of_each_activity, labels = activities, autopct = '%0.2f')
```

Fig 11.Count for each activity

```
# Count for each type
acc = 0
gyro = 0
others = 0
for column in X_train.columns:
    if 'Acc' in str(column):
        acc += 1
    elif 'Gyro' in str(column):
        gyro += 1
    else:
        others += 1

# Show bar plot for the three types
plt.rcParams.update({'figure.figsize': [6, 5], 'font.size': 16})
plt.bar(['Accelerometer',   'Gyroscope', 'Others'], [acc, gyro, others], color = ('y', 'g', 'b'))
```

```
<BarContainer object of 3 artists>
```

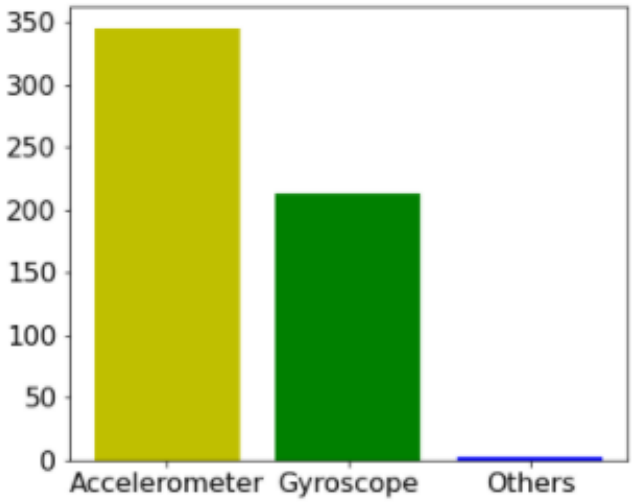


Fig 12. Set time series for each activity

```
standing_activity = training_data[training_data['Activity'] == 'STANDING']
# Reset the index for this dataframe
standing_activity = standing_activity.reset_index(drop=True)
```

```
# Set time series for each subject
time = 1
index = 0
time_series = np.zeros(standing_activity.shape[0])
for row_number in range(standing_activity.shape[0]):
    if (row_number == 0
        or standing_activity.iloc[row_number]['subject'] == standing_activity.iloc[row_number - 1]['subject']):
        time_series[index] = time
        time += 1
    else:
        time_series[index] = 1
        time = 2
    index += 1

# Combine the time_series with the standing_activity dataframe
time_series_df = pd.DataFrame({ 'Time': time_series })
standing_activity_df = pd.concat([standing_activity, time_series_df], axis = 1)
```

Fig 13. Support Vector Classifier Accuracy

```
accuracy_scores = np.zeros(3)
```

```
accuracy_scores = np.zeros(3)

# Support Vector Classifier
clf = SVC().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[0] = accuracy_score(y_test, prediction)*100
print('Support Vector Classifier accuracy: {}%'.format(accuracy_scores[0]))
```

```
Support Vector Classifier accuracy: 95.04580929759076%
```

Fig 14. Decision Tree Accuracy

```
clf = DecisionTreeClassifier().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[2] = accuracy_score(y_test, prediction)*100
print('DecisionTree Classifier accuracy: {}%'.format(accuracy_scores[2]))
```

```
DecisionTree Classifier accuracy: 85.78215134034612%
```

Fig 15. K-Nearest Neighbour Accuracy

```
clf = KNeighborsClassifier().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[1] = accuracy_score(y_test, prediction)*100
print('K Nearest Neighbors Classifier accuracy: {}%'.format(accuracy_scores[1]))
```

```
K Nearest Neighbors Classifier accuracy: 90.02375296912113%
```

Fig 16. Accuracy Of Various Algorithms:

```
colors = cm.rainbow(np.linspace(0, 1, 4))
labels = ['Support Vector Classifier','K Nearest Neighbors','Decision Tree']
plt.bar(labels,
        accuracy_scores,
        color = colors)
plt.xlabel('Classifiers')
plt.ylabel('Accuracy')
plt.title('Accuracy of various algorithms')
```

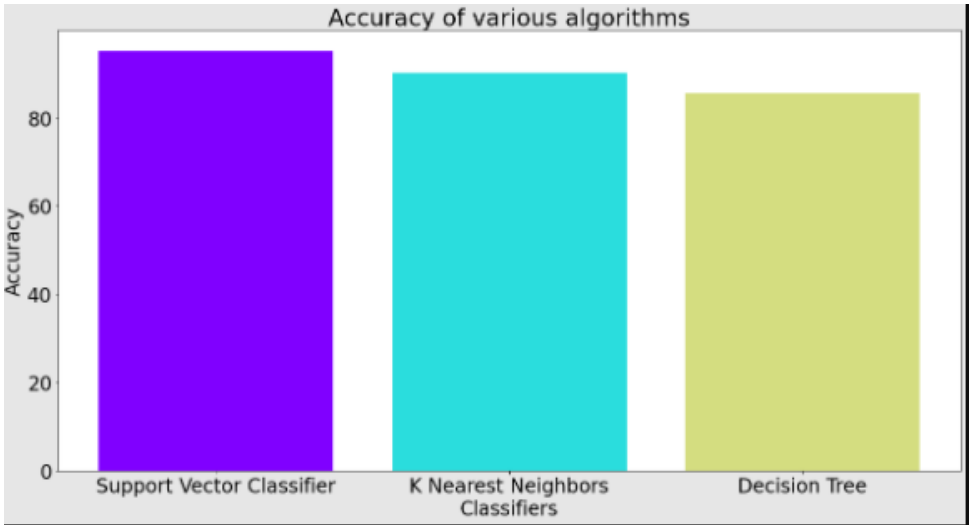


Fig 17. Confusion Matrix

```
lr_svm_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy using linear SVM : ",lr_svm_accuracy)
```

```
Accuracy using linear SVM :  0.9647098744485918
```

```
# function to plot confusion matrix
def plot_confusion_matrix(cm,lables):
    fig, ax = plt.subplots(figsize=(15,10)) # 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=lables, yticklabels=lables,
    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()
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test.values,y_pred)
plot_confusion_matrix(cm, np.unique(y_pred)) # plotting confusion matrix
```