HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES

Thesis submitted in partial fulfillment of the requirements for the

Post Graduate Diploma in Data Science

By

SUNKARA VENKATA SREERAM

19125760094

Under the guidance of

Prof. Anantpadmanabh Divanji
Senior Associate Faculty

Manipal ProLearn

Bangalore

MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES

Thesis submitted in partial fulfillment of the requirements for

Post Graduate Diploma in Data Science

By

SUNKARA VENKATA SREERAM

19125760094

Under the guidance of

Prof. Anantpadmanabh Divanji
Senior Associate Faculty

Manipal ProLearn

Bangalore

MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES

Thesis submitted in partial fulfillment of the requirements for

Post Graduate Diploma in Data Science

By

SUNKARA VENKATA SREERAM

19125760094

Examiner 1	**Examiner 2**
Signature:	Signature:
Name:	Name:

MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

CERTIFICATE

This is to certify that the project work titled

HUMAN ACTIVITY RECOGNITION WITH SMARTPHONES

is a bonafide record of the work done by

SUNKARA VENKATA SREERAM

19125760094

In partial fulfillment of the requirements for the award of **Post Graduate Diploma in Data Science** under Manipal Academy of Higher Education, Manipal, Manipal and the same has not been submitted elsewhere for any kind of certification/recognition.

(Signature)

Prof. Anantpadmanabh Divanji

Senior Associate Faculty

Manipal ProLearn

Bangalore

Table of Contents

ACKNOWLEDGEMENT

I would like to express my deep gratitude to **Prof. Anantpadmanabh Divanji**, my mentor for the project who clarified my understanding towards the data and guided me in data cleaning, data preparation and modelling.

I would also like to thank the project mid-term review panelists which includes **Dr. Subhabaha Pal** for their valuable feedback and suggestions that I incorporated in my project.

My thanks are also extended to professors who taught us subjects in Data Science Diploma program specifically Exploratory Data Science, Data Visualization, Unstructured Data, Machine Learning and Banking Analytics. The teachings in these subjects helped me to complete the project.

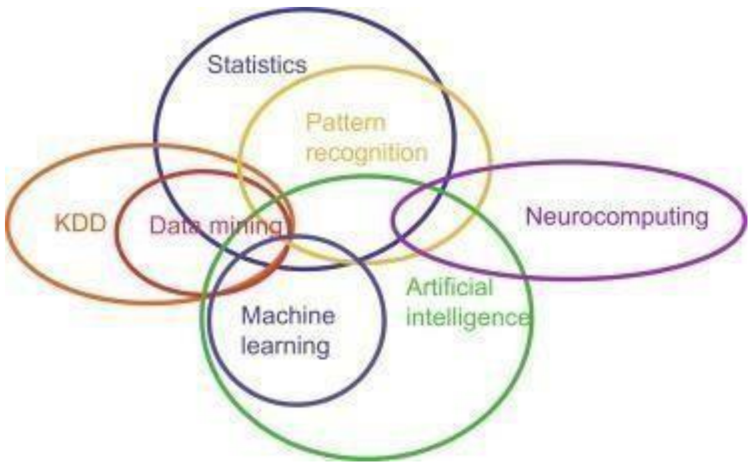
Finally, I wish to thank Manipal Global Academy team who supported with logistics and helped me in completing the project and Post Graduate Diploma in Data Science program on time.

Sunkara Venkata Sreeram

19125760094

ABSTRACT

Machine learning is a subfield of Artificial Intelligence (AI) and has evolved from pattern recognition, used to explore the structure of the data and fit into the models which can be understood and utilized by users. It answers the question as to how to construct a computer program using historical data, to solve a given problem and automatically improve the efficiency of the program, with experience. In the recent years, various machine learning applications have been developed, like model to classify new astronomical structures, detecting fraudulent banking transactions, information-filtering systems that learn user reading preferences, neurobiological studies, autonomous vehicles that learn to drive on highways. Also at the same time, there has been an important progress in the concepts and the algorithms that form the foundation of machine learning.



The machine learning field, which can be briefly defined as enabling computers make successful predictions using past experiences, has exhibited an impressive development recently with the help of the rapid increase in the storage capacity and processing power of computers.

1. INTRODUCTION

1.1 MOTIVATION

Since 1970 a large growth in human life expectancy has been observed in India. This growth has expanded in the whole world principally due to the great achievements in health care field. As a result, the proportion of elderly people is rapidly increasing. Aging people in general lives in isolated conditions. In addition to that some of them are not capable to live normally and take advantages from health care facilities services. Building remote monitoring systems for elderly patients who live alone or without permanent caretaking will improve their quality of life. For better decision making these remote monitoring systems needs some regular and trustful information about patients. In this perspective, this work provides a comprehensive, state of the art review of the current situation of human activity recognition (HAR) solutions in the context of inertial sensors in smartphones.

1.2 PROJECT SCOPE

Future scope of this project will involve adding more parameters and factors like the gyroscopic ratios, direction, motion of sensors, etc. The more the parameters are taken into account more will be the accuracy. The recognition part of his system is based on an SVM model already trained, capable of predicting activities performed by users. We can easily come with a condition to separate static activities and dynamic activities by using density plot. The use of traditional algorithms and data mining techniques can also help predict the smartphones performance structure as a whole.

1.3 PROJECT GOAL

Machine learning is a subfield of Artificial Intelligence (AI) and has evolved from pattern recognition, used to explore The goal of this project is to build a machine learning model and a signal processing pipeline in offline mode capable of processing signals collected using smart phone inertial sensors and producing useful datasets will be used as inputs of a machine learning model capable of recognizing some of human daily activities (sitting, walking …) included in the dataset (see datasets and Inputs section) with a low error rate. The signal processing pipeline and the final model could be used as a good source of information about patient’s daily activities needed by remote monitoring systems.

1.4 Literature/Market survey

Human activity recognition has been studied for years and researchers have propose different solutions to attack the problem. Existing approaches typically use vision sensor, inertial sensor and the mixture of both. Machine learning and threshold-base algorithms are often applied. Machine learning usually produces more accurate and reliable results, while threshold-based algorithms are faster and simpler. One or multiple cameras have been used to capture and identify body posture. Multiple accelerometers and gyroscopes attached to different body positions are the most common solutions. Approaches that combine both vision and inertial sensors have also been purposed. Another essential part of all these algorithms is data processing. The quality of the input features has a great impact on the performance. Some previous works are focused on generating the most useful features from the time series data. The common approach is to analyze the signal in both time and frequency domain.

1.5 Organization of the report

The purpose of a report is to inform the reader. It is helpful, both to the reader and to the writer, if the report is logically organized. Over the years a standard format for reports has been worked out. Although there may be circumstances when it is advisable to change the format to fit a particular need, following the format ensures that all the essential information is included and that it is treated in a logical way. The format usually adopted is that described in a British Standard (1972). The standard components of a report are as follows:

Title
Summary
List of contents
Introduction
Main body of the report Conclusions
Recommendations
Appendix
References.

2. PROJECT DESCRIPTION

2.1 BUSINESS UNDERSTANDING

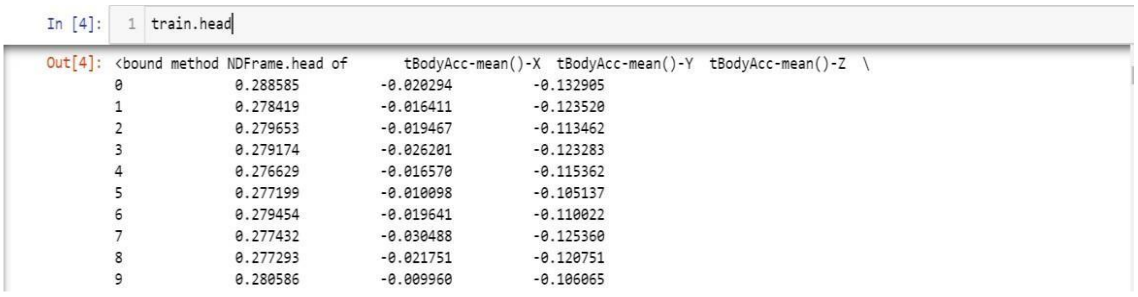
The experiments were carried out with a group of 30 volunteers within an age bracket of 19-48 years. They performed a protocol of activities composed of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). The experiment also included postural transitions that occurred between the static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to- sit, stand- to-lie, and lie-to-stand. All the participants were wearing a smartphone (Samsung Galaxy S II) on the waist during the experiment execution. They captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz using the embedded accelerometer and gyroscope of the device. The experiments were video-recorded to label the data manually. The resulted dataset stored in the directory Raw-Data could be considered as the original labelled dataset and from it different subsets would be generated. Signals (Acceleration and Gyroscope) were pre-processed by applying noise filters and then sampled in fixed width sliding windows of 2.56 sec and 50% overlap.

2.2 DATASET UNDERSTANDING

The dataset with a group of 30 volunteers with an age of 19-48 years. Everyperson performed six activities wearing a smartphone. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3- axial angular velocity at a constant rate of 50 HZ. Where 70% and 30% volunteers divided for training and testing data. There are multiple variables in the dataset, like

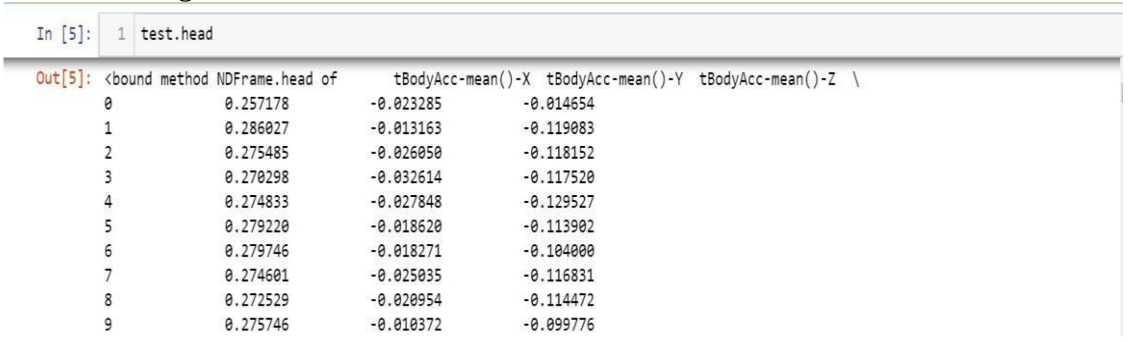
Feature	**Count**
fBody Acc	79
fBody Gyro	79
fBody Accjerk	79
t Gravity Acc	40
T Body Acc	40
tBody Accjerk Mag	13
tBody Acc Mag	13

Size of training set: **7352** records



```
In [4]:  train.head

Out[4]: <bound method NDFrame.head of       tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  \
        0              0.288585          -0.020294          -0.132905
        1              0.278419          -0.016411          -0.123520
        2              0.279653          -0.019467          -0.113462
        3              0.279174          -0.026201          -0.123283
        4              0.276629          -0.016570          -0.115362
        5              0.277199          -0.010098          -0.105137
        6              0.279454          -0.019641          -0.110022
        7              0.277432          -0.030488          -0.125360
        8              0.277293          -0.021751          -0.120751
        9              0.280586          -0.009960          -0.106065
```

Size of testing set: **3200** records



```
In [5]:  test.head

Out[5]: <bound method NDFrame.head of       tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  \
        0              0.257178          -0.023285          -0.014654
        1              0.286027          -0.013163          -0.119083
        2              0.275485          -0.026050          -0.118152
        3              0.270298          -0.032614          -0.117520
        4              0.274833          -0.027848          -0.129527
        5              0.279220          -0.018620          -0.113902
        6              0.279746          -0.018271          -0.104000
        7              0.274601          -0.025035          -0.116831
        8              0.272529          -0.020954          -0.114472
        9              0.275746          -0.010372          -0.099776
```

2.3 DATA LIMITATIONS

- Missing values
- Extreme values or Outliers
- Duplicate records

3. APPROACH

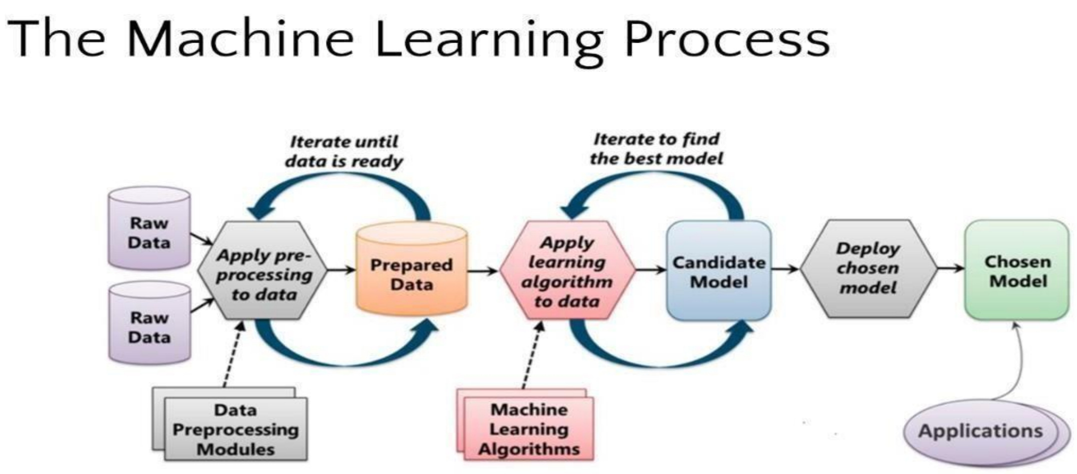

The Machine Learning Process
Iterate until data is ready
Iterate to find the best model
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

Fig.1

3.1 EXPLORATORY DATA ANALYSIS

In data mining, Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. EDA is used for seeing what the data can tell us before the modelling task. It is not easy to look at a column of numbers or a whole spreadsheet and determine important characteristics of the data. It may be tedious, boring, and/or overwhelming to derive insights by looking at plain numbers. Exploratory data analysis techniques have been devised as an aid in this situation.

Exploratory data analysis is generally cross-classified in two ways.

- First, each method is either non-graphical or graphical.
- And second, each method is either univariate or multivariate (usually just bivariate).

Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to

- maximize insight into a data set
- uncover underlying structure
- extract important variables
- detect outliers and anomalies
- test underlying assumptions
- develop parsimonious models

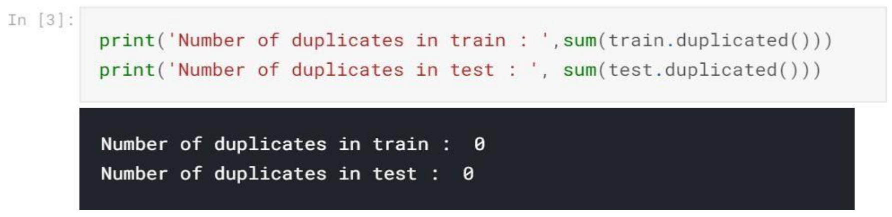
3.2 DATA CLEANING

Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results. There are several methods for cleaning data depending on how it is stored along with the answers being sought.

Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set's accuracy without necessarily deleting information.

Benefits of Data Cleaning

1. It removes major errors and inconsistencies that are inevitable when multiple sources of data are getting pulled into one dataset.
2. Using tools to cleanup data will make everyone more efficient since they'll be ableto quickly get what they need from the data.
3. Fewer errors means happier customers and fewer frustrated employees.
4. The ability to map the different functions and what your data is intended to do and where it is coming from your data.



```
In [3]:
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
```

Fig.3

So no need of doing any type of null value treatment .

3.3 DATA PREPROCESSING

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues.

In Real world data are generally incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names.

3.4 MODELLING, PREDICTION AND PERFORMANCE EVALUATION

Modelling

The outputs of prediction and feature engineering are a set of label times, historical examples of what we want to predict, and features, predictor variables used to train a model to predict the label. The process of modeling means training a machine learning algorithm to predict the labels from the features, tuning it for the business need, and validating it on holdout data. The output from modeling is a trained model that can be used for inference, making predictions on new data points.

Similar to feature engineering, modeling is independent the previous steps in the machine learning process and has standardized inputs which means we can alter the prediction problem without needing to rewrite all our code. If the business requirements change, we can generate new label times, build corresponding features, and input them into the model.

Prediction

Prediction refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome.

The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.

Why Prediction?

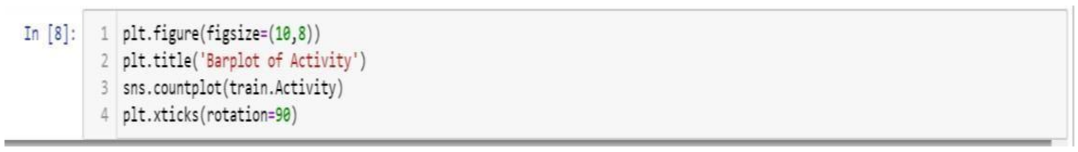
Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things – customer churn likelihood, possible fraudulent activity, and more. These provide the business with insights that result in tangible business value.

For example, if a model predicts a customer is likely to churn, the business can target them with specific communications and outreach that will prevent the loss of that customer.

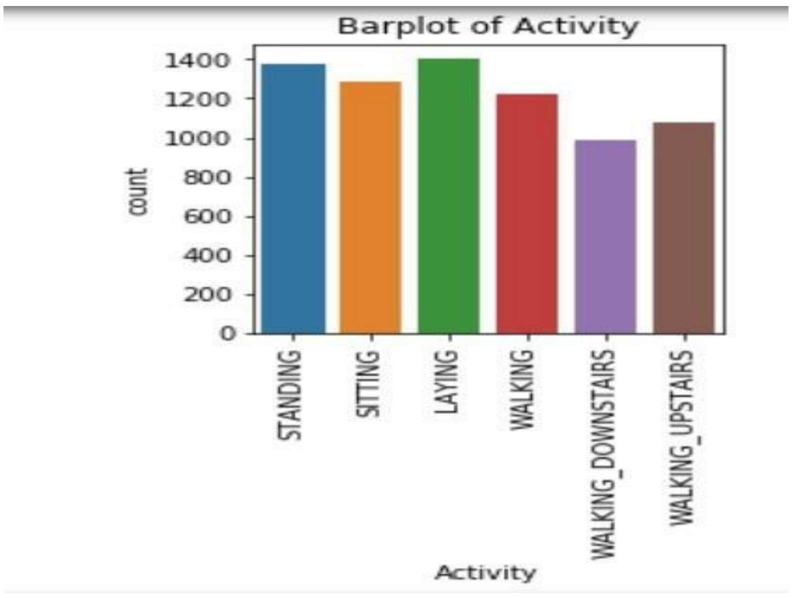
Performance Evaluation

Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis. The most common cross-validation technique is k-fold cross-validation, where the original dataset is partitioned into k equal size subsamples, called folds.

There is no data cleaning required for this dataset. There is almost same number of observations across all the six activities so this data does not have class imbalance problem.

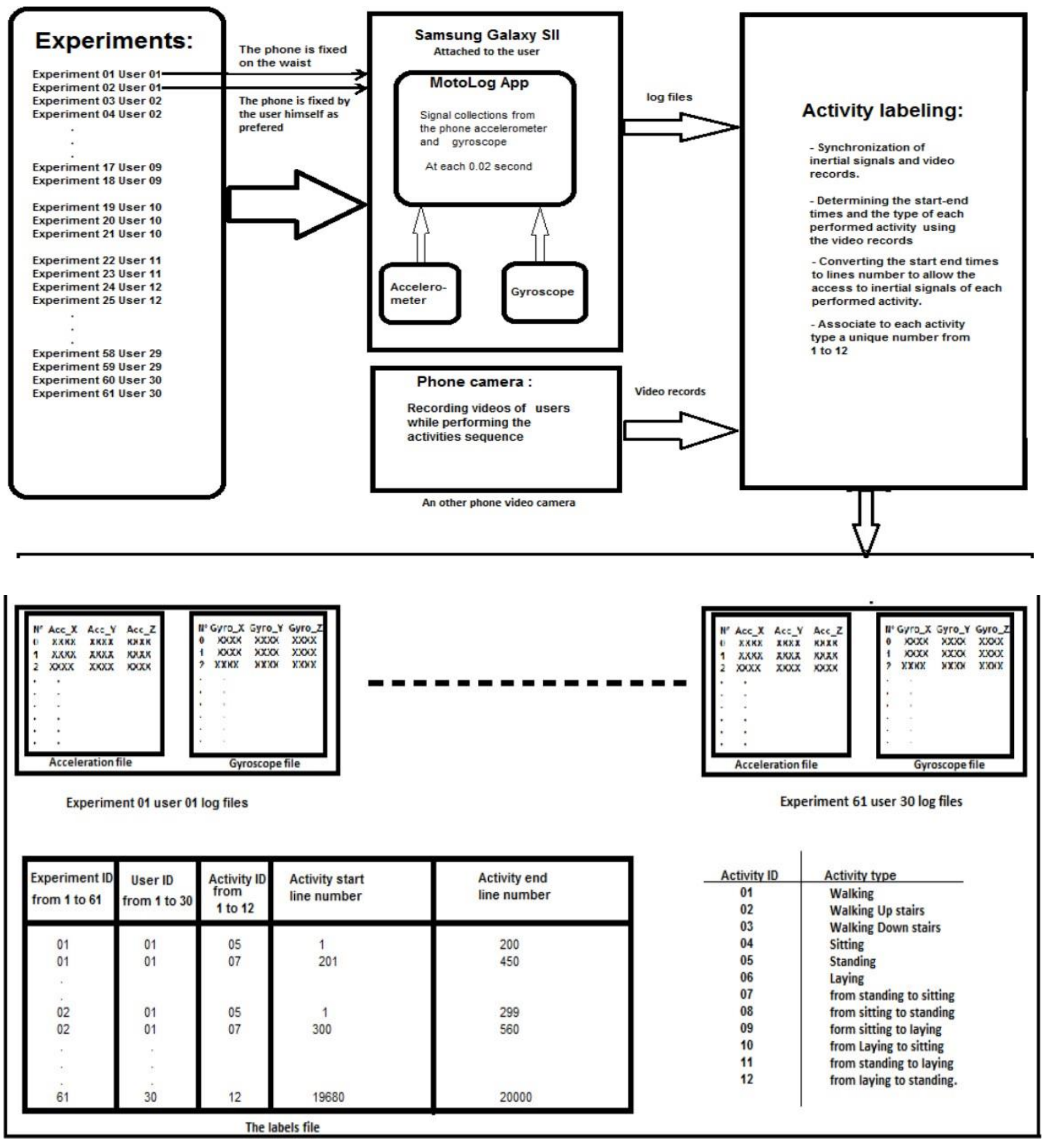


```
In [8]:
plt.figure(figsize=(10,8))
plt.title('Barplot of Activity')
sns.countplot(train.Activity)
plt.xticks(rotation=90)
```



4. DESIGN

4.1 TECHNOLOGY USED


Experiments:
Experiment 01 User 01
Experiment 02 User 01
Experiment 03 User 02
Experiment 04 User 02
Experiment 17 User 09
Experiment 18 User 09
Experiment 19 User 10
Experiment 20 User 10
Experiment 21 User 10
Experiment 22 User 11
Experiment 23 User 11
Experiment 24 User 12
Experiment 25 User 12
Experiment 58 User 29
Experiment 59 User 29
Experiment 60 User 30
Experiment 61 User 30
The phone is fixed on the waist
The phone is fixed by the user himself as prefered
Samsung Galaxy SII
Attached to the user
MotoLog App
Signal collections from the phone accelerometer and gyroscope
At each 0.02 second
Accelero-meter
Gyroscope
log files
Phone camera :
Recording videos of users while performing the activities sequence
An other phone video camera
Video records
Activity labeling:
- Synchronization of inertial signals and video records.
- Determining the start-end times and the type of each performed activity using the video records
- Converting the start end times to lines number to allow the access to inertial signals of each performed activity.
- Associate to each activity type a unique number from 1 to 12
N° Acc_X Acc_Y Acc_Z
N° Gyro_X Gyro_Y Gyro_Z
Acceleration file
Gyroscope file
Experiment 01 user 01 log files
Experiment 61 user 30 log files

Experiment ID from 1 to 61	User ID from 1 to 30	Activity ID from 1 to 12	Activity start line number	Activity end line number
01	01	05	1	200
01	01	07	201	450
.				
.				
02	01	05	1	299
02	01	07	300	560
.	.			
.	.			
.	.			
61	30	12	19680	20000

The labels file

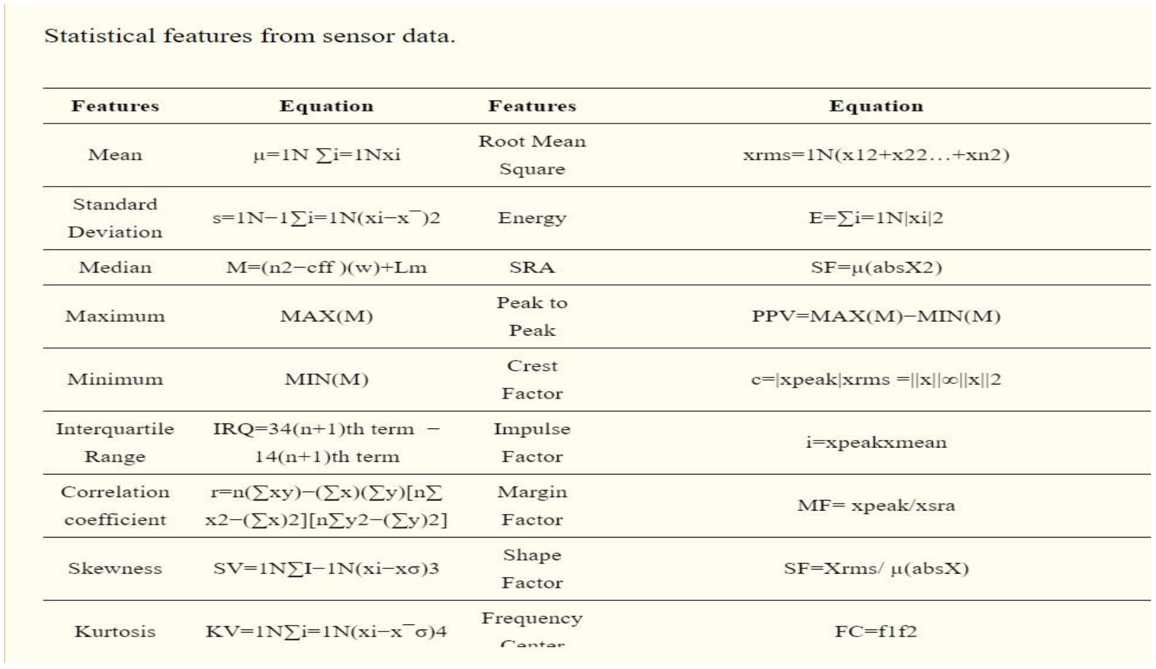
Activity ID	Activity type
01	Walking
02	Walking Up stairs
03	Walking Down stairs
04	Sitting
05	Standing
06	Laying
07	from standing to sitting
08	from sitting to standing
09	form sitting to laying
10	from Laying to sitting
11	from standing to laying
12	from laying to standing.

4.2 STATISTICAL FEATURE EXTRACTION

Statistics is a robust instrument that can operate on data and produce meaningful information. It can produce technical data by which data analysis is possible from high-level perspective. From a high-level view, statistics is the use of mathematics to perform technical analysis of data. Statistics is capable of producing both visualization graphs and in-depth statistical data, which are more information-driven and work in a targeted way to reach concrete conclusions about our data rather than just projecting. Thus, statistics creates deeper and fine-grained insights, which exactly depict the data structuring. Utilizing other data science techniques side by side optimally helps to produce even more meaningful and in-depth information.

Feature is a statistical function that works brilliantly to extract meaningful information of data in a natural way. From the perspective of human activity recognition, a particular pattern is generated from a particular physical movement of users. As an example, the “run” activity has a particular pattern as it involves superior physical effort from a human, which is quite different from the “walking” activity pattern. Thus, inertial sensors like an accelerometer and gyroscope can measure the intensity of each physical effort and produce different pattern distributions. Collected pattern distribution i.e., statistical data information from these sensors can distinguish between “walking” and “running” activity. Hence, the standard deviation or any other statistical feature is capable of highlighting the difference between these two activities.

In the feature extraction process, we extract the time and frequency domain statistical feature from three axial data from both accelerometer and gyroscope. To extract the maximum information, a total of 23 base features have been applied to collected data, which includes both time domain and frequency domain statistical parameters. The 23 features, including their corresponding statistical formulas, are formulated in

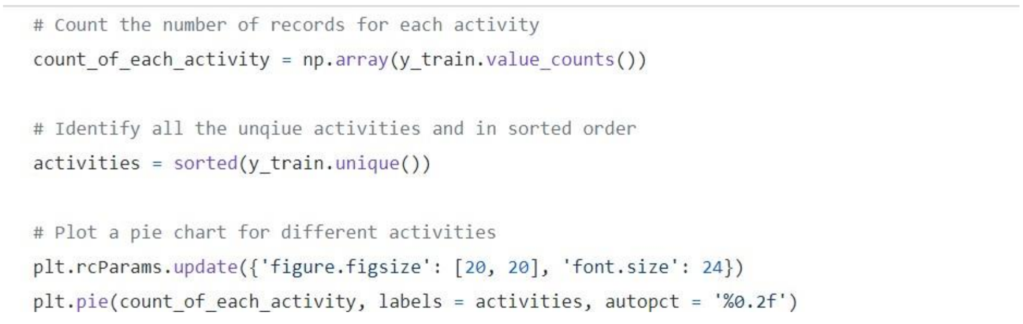


Statistical features from sensor data.

Features	Equation	Features	Equation
Mean	μ=1N ∑i=1Nxi	Root Mean Square	xrms=1N(x12+x22…+xn2)
Standard Deviation	s=1N−1∑i=1N(xi−x¯)2	Energy	E=∑i=1N|xi|2
Median	M=(n2−cff)(w)+Lm	SRA	SF=μ(absX2)
Maximum	MAX(M)	Peak to Peak	PPV=MAX(M)−MIN(M)
Minimum	MIN(M)	Crest Factor	c=|xpeak|xrms =||x||∞||x||2
Interquartile Range	IRQ=34(n+1)th term − 14(n+1)th term	Impulse Factor	i=xpeakxmean
Correlation coefficient	r=n(∑xy)−(∑x)(∑y)[n∑ x2−(∑x)2][n∑y2−(∑y)2]	Margin Factor	MF= xpeak/xsra
Skewness	SV=1N∑I−1N(xi−xσ)3	Shape Factor	SF=Xrms/ μ(absX)
Kurtosis	KV=1N∑i=1N(xi−x¯σ)4	Frequency Center	FC=f1f2

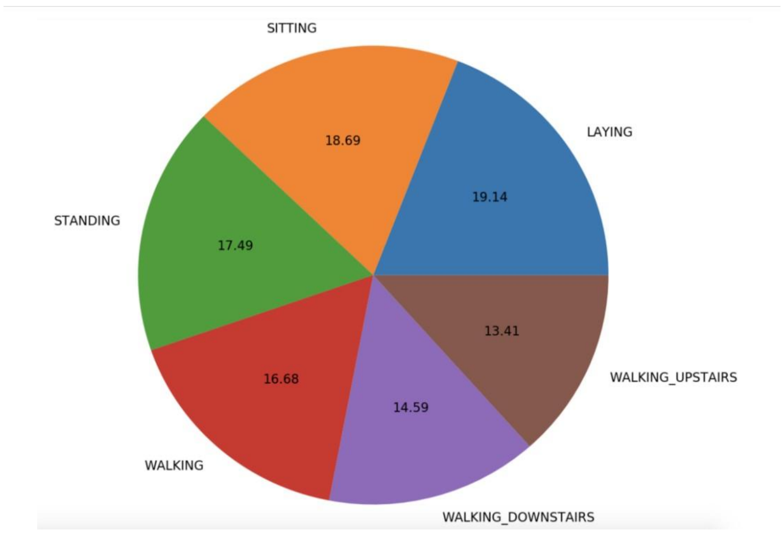
4.3 DATA VISUALIZATION

One of the key things when working with data is to ensure that all classes are of approximately equal size. This is essential so that the machine learning algorithm is not biased towards any one class.



```
# 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': [20, 20], 'font.size': 24})
plt.pie(count_of_each_activity, labels = activities, autopct = '%0.2f')
```

Pie shares each activity in the data set



5. EXPLORATORY DATA ANALYSIS

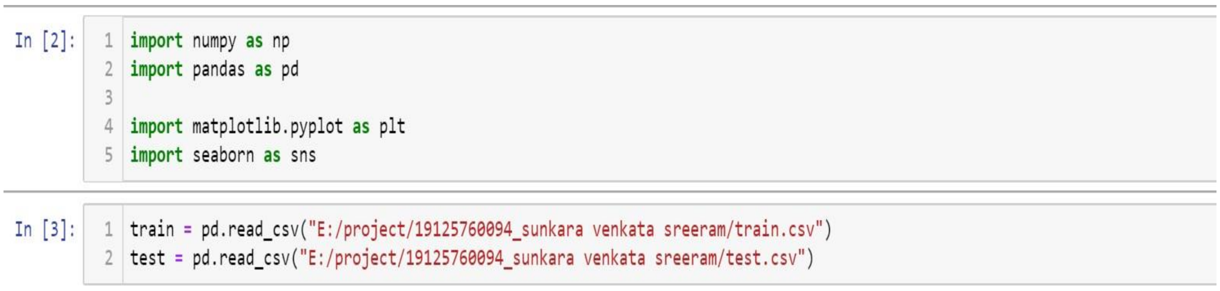
Based on the common nature of activities we can broadly put them in two categories.

Static and dynamic activities:

- SITTING, STANDING, LAYING can be considered as static activities with no motion involved
- WALKING, WALKING_DOWNSTAIRS, WALKING_UPSTAIRS can be considered as dynamic activities with significant amount of motion involved

Let's consider tBodyAccMag-mean() feature to differentiate among these two broader Set of activities. If we try to build a simple classification model to classify the activity using one variable at a time then probability density function(PDF) is very helpful to assess importance of a continuous variable.

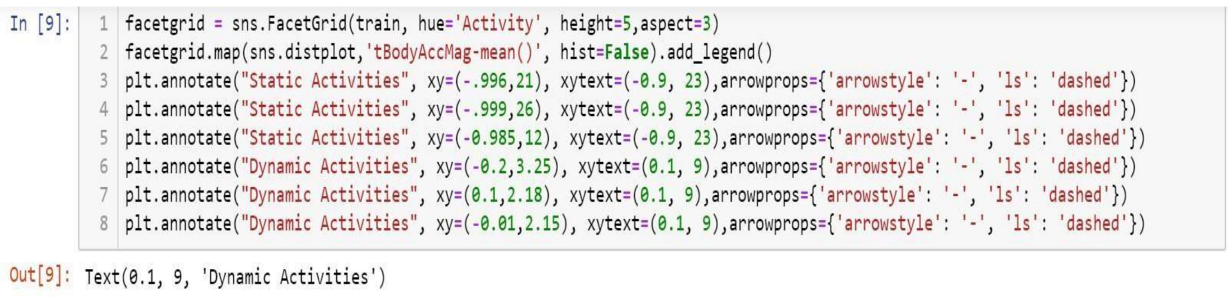
Data Import:



```
In [2]:
import numpy as np
import pandas as pd

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

```
In [3]:
train = pd.read_csv("E:/project/19125760094_sunkara venkata sreeram/train.csv")
test = pd.read_csv("E:/project/19125760094_sunkara venkata sreeram/test.csv")
```

5.1 ANALYZING T-BODY ACC MAG-MEAN FEATURE:



```
In [9]:
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=(-.996,21), xytext=(-0.9, 23),arrowprops={'arrowstyle': '-', 'ls': 'dashed'})
plt.annotate("Static Activities", xy=(-.999,26), xytext=(-0.9, 23),arrowprops={'arrowstyle': '-', 'ls': 'dashed'})
plt.annotate("Static Activities", xy=(-0.985,12), xytext=(-0.9, 23),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'})

Out[9]: Text(0.1, 9, 'Dynamic Activities')
```

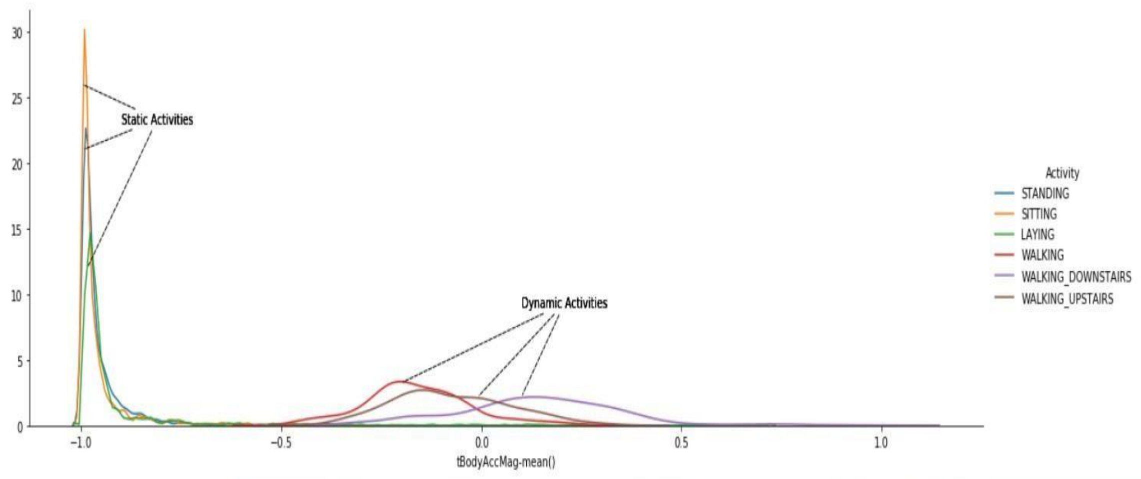
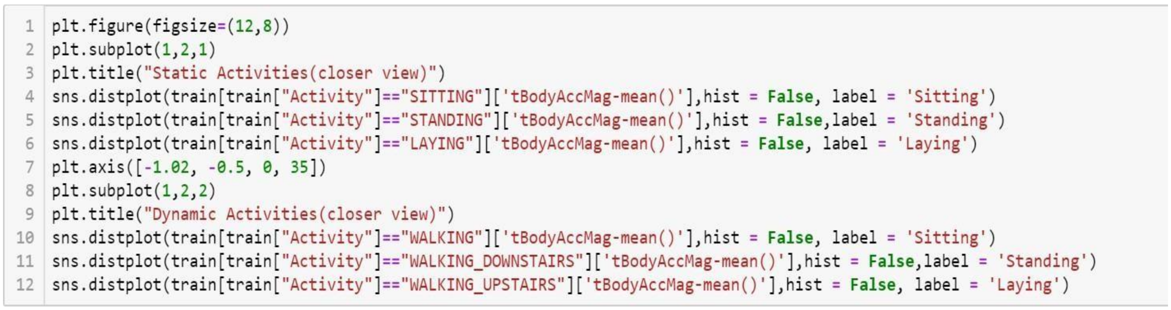


Fig.5

Using the above density plot we can easily come with a condition to separate 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.



```python
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, 35])
plt.subplot(1,2,2)
plt.title("Dynamic Activities(closer view)")
sns.distplot(train[train["Activity"]=="WALKING"]['tBodyAccMag-mean()'],hist = False, label = 'Sitting')
sns.distplot(train[train["Activity"]=="WALKING_DOWNSTAIRS"]['tBodyAccMag-mean()'],hist = False,label = 'Standing')
sns.distplot(train[train["Activity"]=="WALKING_UPSTAIRS"]['tBodyAccMag-mean()'],hist = False, label = 'Laying')
```

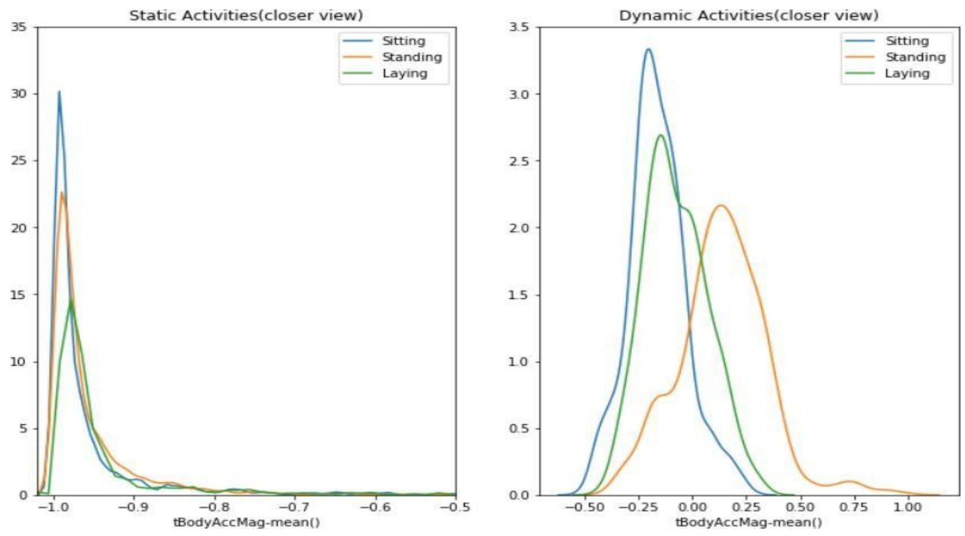
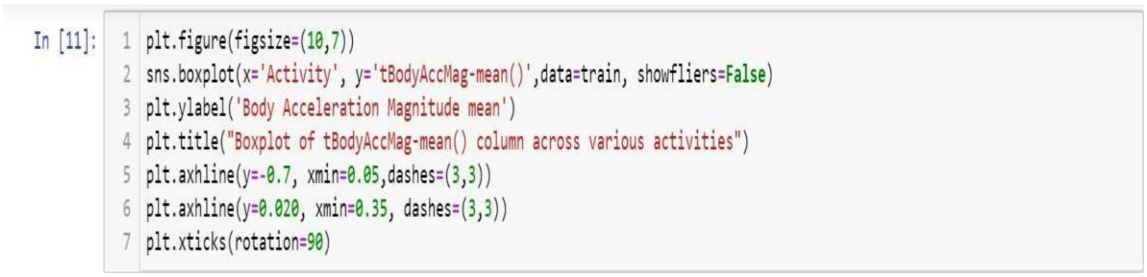


Fig.6

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

BOX PLOT



```
In [11]:
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=-0.7, xmin=0.05,dashes=(3,3))
plt.axhline(y=0.020, xmin=0.35, dashes=(3,3))
plt.xticks(rotation=90)
```

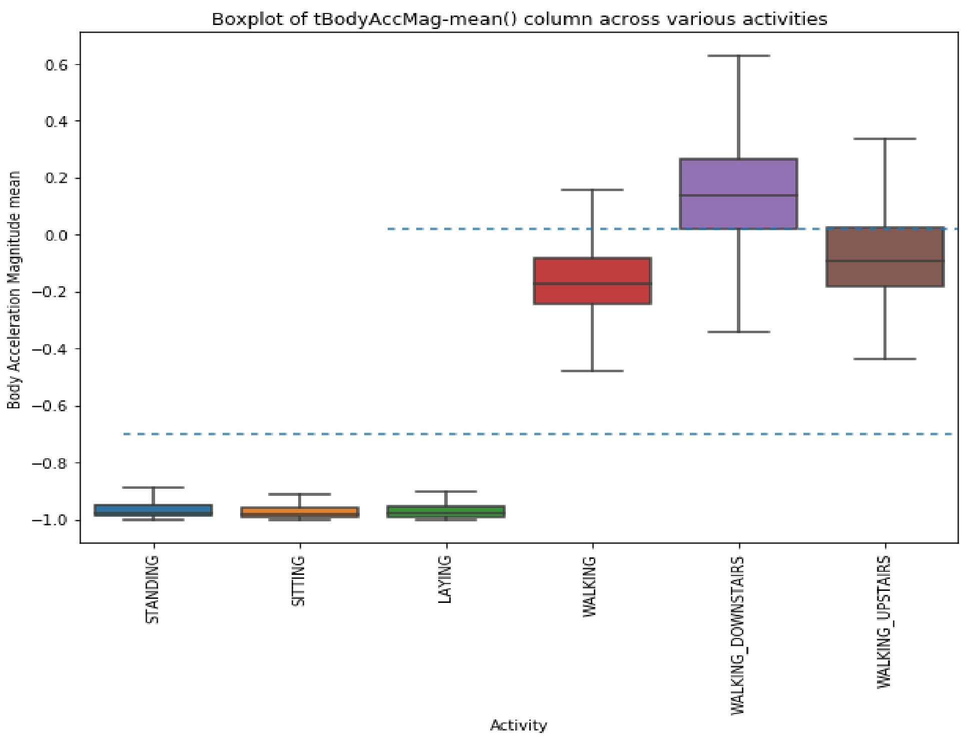


Fig.7

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

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

Also, we can easily separate 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.

5.2 ANALYZING ANGLE BETWEEN X-AXIS AND GRAVITY MEAN FEATURE:

```
In [11]:  plt.figure(figsize=(5,3))
          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 = 90)

Out[11]:  (array([0, 1, 2, 3, 4, 5]), <a list of 6 Text xticklabel objects>)
```

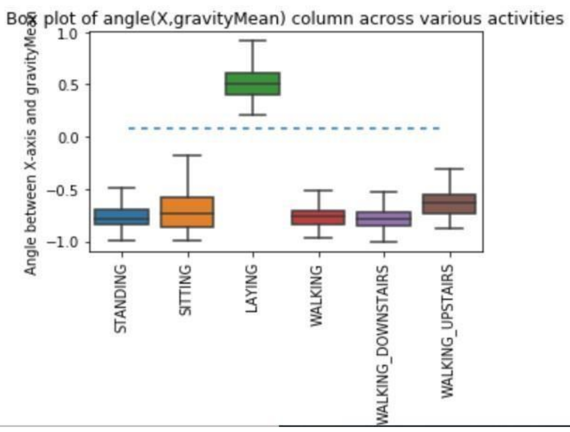


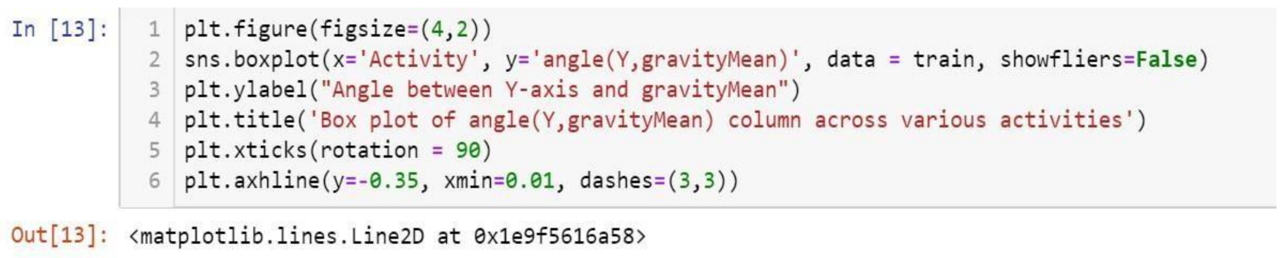
Fig.8

From the boxplot we can observe that angle(X,gravityMean) perfectly separates LAYING from other activities.

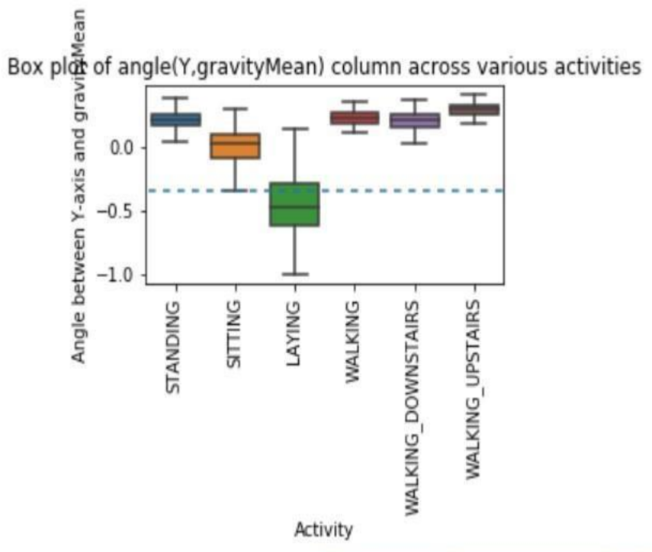
```
if(angle(X,gravityMean)>0.01):
    Activity = "LAYING"
else:
    Activity = "others"
```

5.3 ANALYGING ANGLE BETWEEN Y-AXIZ AND GRAVITY MEAN FEATURE:

Similarly, using Angle between Y-axis and gravity Mean we can separate LAYING from other activities but again it leads to some misclassification error.

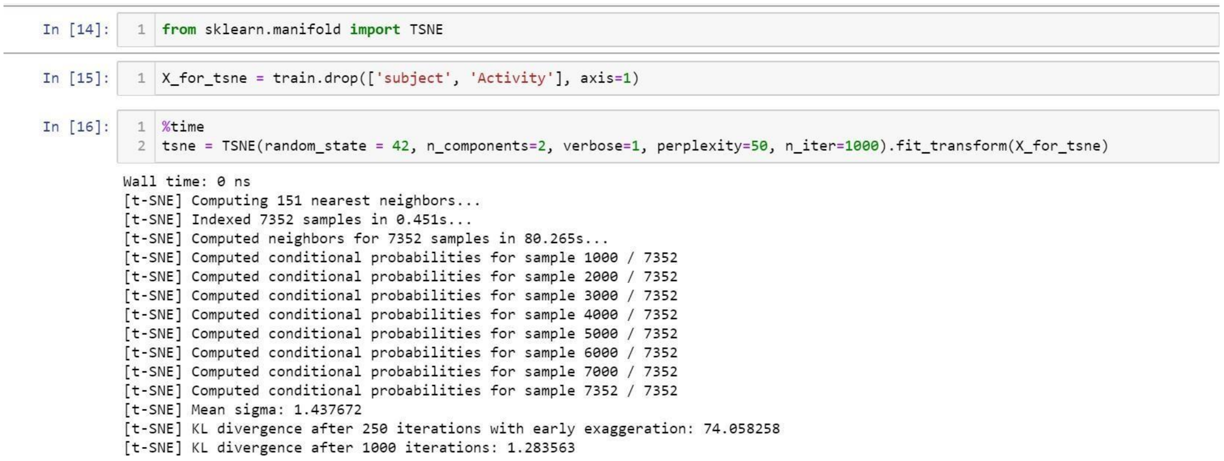


```
In [13]: plt.figure(figsize=(4,2))
         sns.boxplot(x='Activity', y='angle(Y,gravityMean)', data = train, showfliers=False)
         plt.ylabel("Angle between Y-axis and gravityMean")
         plt.title('Box plot of angle(Y,gravityMean) column across various activities')
         plt.xticks(rotation = 90)
         plt.axhline(y=-0.35, xmin=0.01, dashes=(3,3))

Out[13]: <matplotlib.lines.Line2D at 0x1e9f5616a58>
```



5.4 VISUALIZING DATA USING T-SNE:

T-distributed stochastic neighbor embedding is a non- linear dimensionality reduction algorithm used for exploring high dimensional data.
It maps multi-dimensional data to two or more dimensions suitable for Human observations. Using t-SNE data can be visualized from an extremely high dimensional space to a low dimensional space and still it retains lots of actual information. Given training data has 561 unique features, using t-SNE let's visualize it to a 2D space.



```
In [14]:  from sklearn.manifold import TSNE

In [15]:  X_for_tsne = train.drop(['subject', 'Activity'], axis=1)

In [16]:  %time
          tsne = TSNE(random_state = 42, n_components=2, verbose=1, perplexity=50, n_iter=1000).fit_transform(X_for_tsne)

          Wall time: 0 ns
          [t-SNE] Computing 151 nearest neighbors...
          [t-SNE] Indexed 7352 samples in 0.451s...
          [t-SNE] Computed neighbors for 7352 samples in 80.265s...
          [t-SNE] Computed conditional probabilities for sample 1000 / 7352
          [t-SNE] Computed conditional probabilities for sample 2000 / 7352
          [t-SNE] Computed conditional probabilities for sample 3000 / 7352
          [t-SNE] Computed conditional probabilities for sample 4000 / 7352
          [t-SNE] Computed conditional probabilities for sample 5000 / 7352
          [t-SNE] Computed conditional probabilities for sample 6000 / 7352
          [t-SNE] Computed conditional probabilities for sample 7000 / 7352
          [t-SNE] Computed conditional probabilities for sample 7352 / 7352
          [t-SNE] Mean sigma: 1.437672
          [t-SNE] KL divergence after 250 iterations with early exaggeration: 74.058258
          [t-SNE] KL divergence after 1000 iterations: 1.283563
```

```
In [19]:  plt.figure(figsize=(7,4))
          sns.scatterplot(x =tsne[:, 0], y = tsne[:, 1], hue = train["Activity"],palette="bright")

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1e980090c50>
```

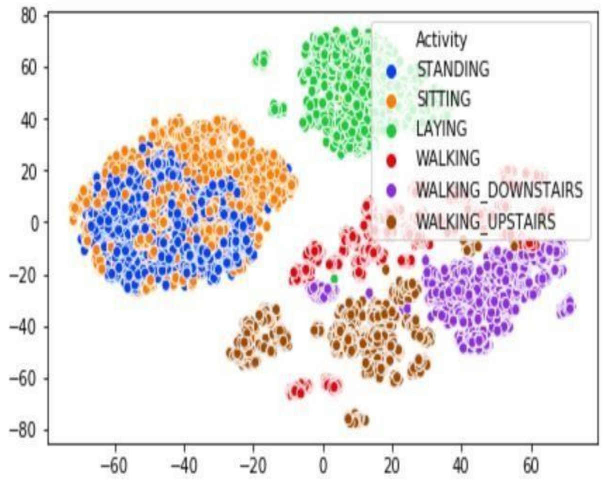
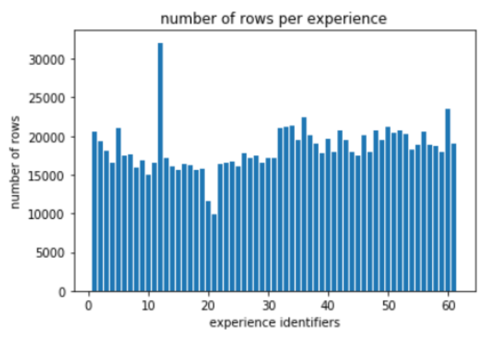
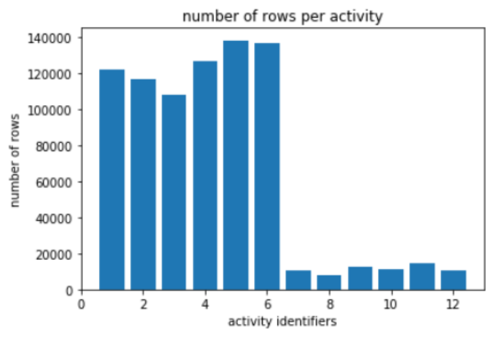


Fig.10

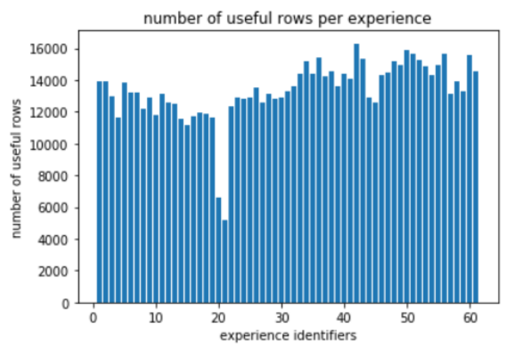
Number of rows per Experience



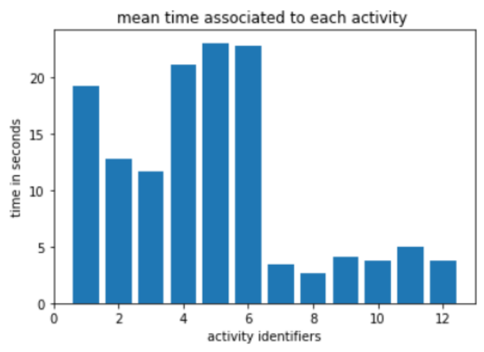
Number of rows per activity



Number of useful rows per Experience



Mean time in seconds per Activity



6. MODELLING

As mentioned above, our main objective is to construct a highly accurate classifier that generalizes well on data from new individuals. For this purpose, we have tested the performance of different classifiers, and assessed why some models performed well while others performed poorly. Algorithms implemented, as well as our motivation for each algorithm, include:

Multinomial model

One of the less complex models implemented. Given the size and high dimensionality of our data, we decided to start with a model less prone to overfitting that would serveas a baseline for the performance of more complex models.

Support vector machines

As indicated by the PCA figure, some clusters fully overlap while other clusters only partially overlap, dependent on how the corresponding activities were performed. Therefore, we would expect maximizing margins when separating these activities to result in good performance. We chose to implement SVMs using a one-vs-one approach that trains a separate classifier for each different pair of labels, as this generally outperforms a one-vs-all approach, particularly in the case of similar classes. We experimented with linear, radial-basis and polynomial kernels, tuning each model and evaluating their performance.

Gradient boosted trees - Our data is high dimensional and there is a high level of interaction among the features, both of which boosted trees tend to handle well. We were particularly interested to see how this model would perform compared to SVMs.

Linear discriminant analysis - The potential of the model for high accuracy was inferred from the projected data using PCA, which indicated visible clusters for each activity.

The parameters of each model require a certain amount of tuning and experimentation to optimize performance. Tuning for each of the models has been performed exclusively on the training data via 7-fold cross-validation, splitting the training data into disjoint training and validation sets, while the test data is held out solely for a final performance.

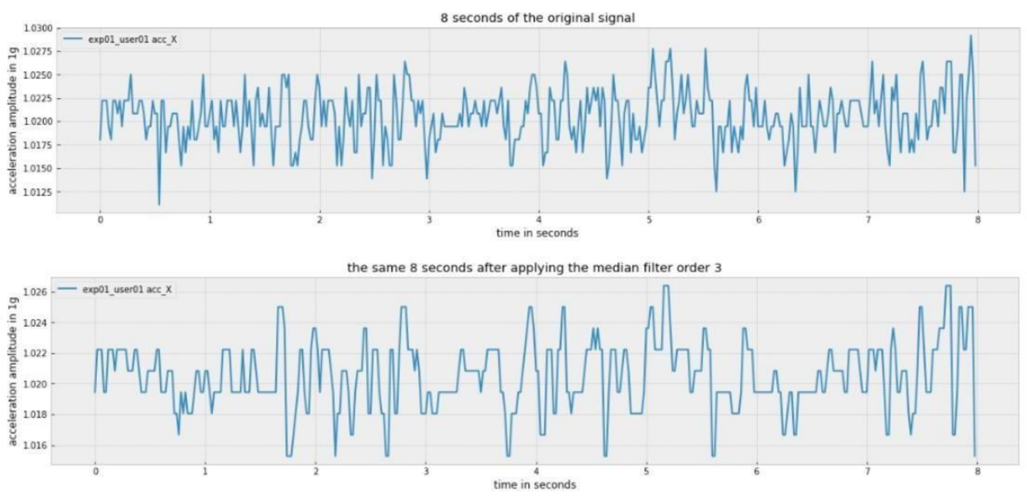
6.1 ALGORITHMS and TECHNIQUES

1. Logistic Regression
2. Linear and Kernel SVM
3. Decision Tree
4. RandomForest

6.2 CHALLENGES FACED

Noise filtering: It was difficult to The features selected for this database come from the accelerometer and gyroscope 3-axial raw signals t_Acc-XYZ and t_Gyro-XYZ. These time domain signals (**prefix 't' to denote time**) were captured at a constant rate of 50 Hz. Then they were filtered using a median filter and a 3rd order low pass Butterworth filter with a corner frequency of 20 Hz to remove noise.

1- Median Filter: was applied to reduce background noise.



2- 3rd order Low pass Butterworth filter with a cut-off, frequency = 20hz was applied to remove high frequency noise.

Resulted Signals are: total_acc_XYZ and
Gyro_XYZ. Gravity filtering:

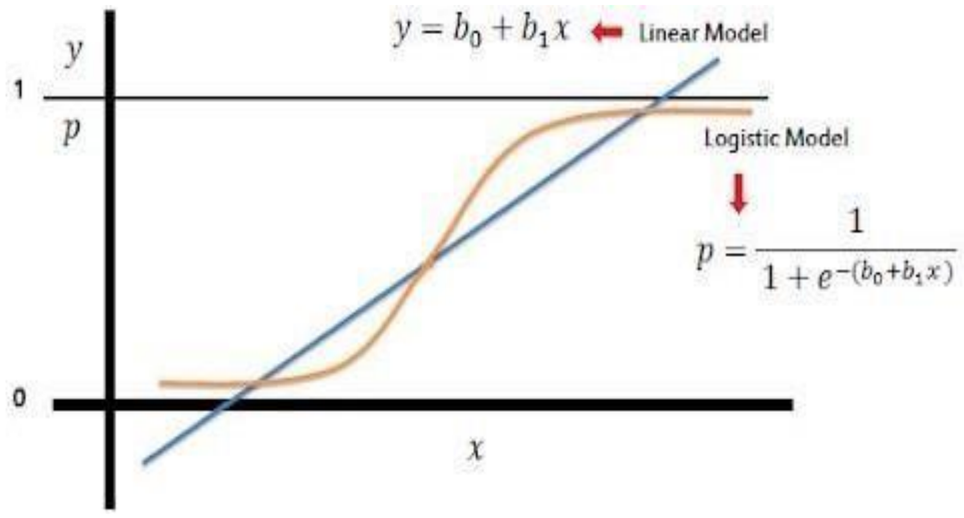
Similarly, the acceleration signal total-acc-XYZ was then separated into body and gravity acceleration signals (tBodyAcc-XYZ and tGravityAcc-XYZ) using another low pass Butterworth filter with a corner frequency of 0.3 Hz. Since the gravitational force is assumed to have only low frequency components.

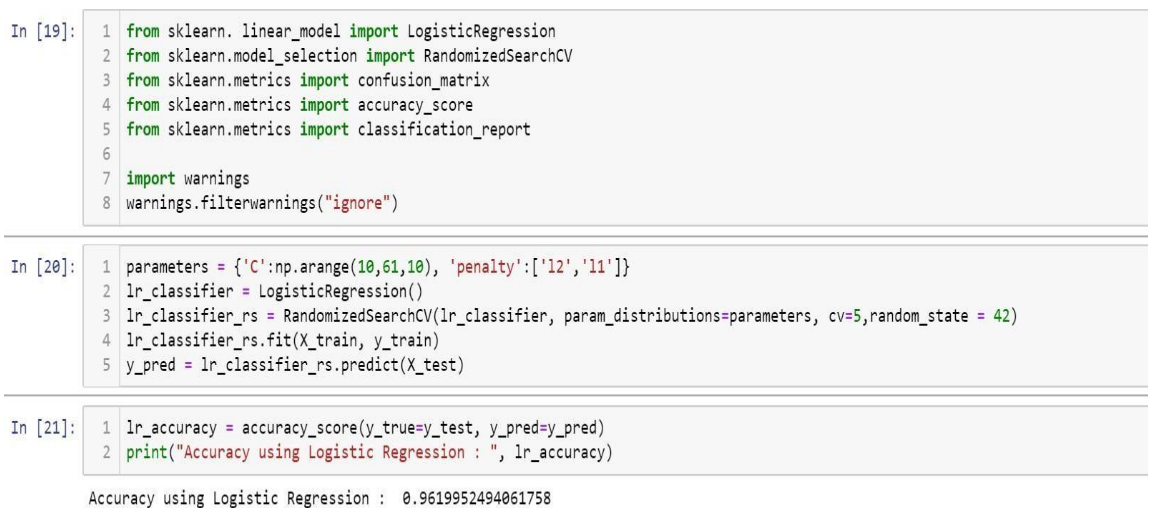
Resulted components are: total_acc-XYZ ==> tBody_acc-XYZ + tGravity_acc-XYZ

6.3 IMPLEMENTATION OF MODELS

1. LINEAR AND KERNEL SVM

Regression is a statistical technique that shows an algebraic relationship between two or more variables. Based on this algebraic relationship (rather a function), one can estimate the value of a variable, given the values of other variables. If any relation is found, regression is used to find degree of relationship – that can be then used for prediction. Explains the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Treated as classification technique**.**





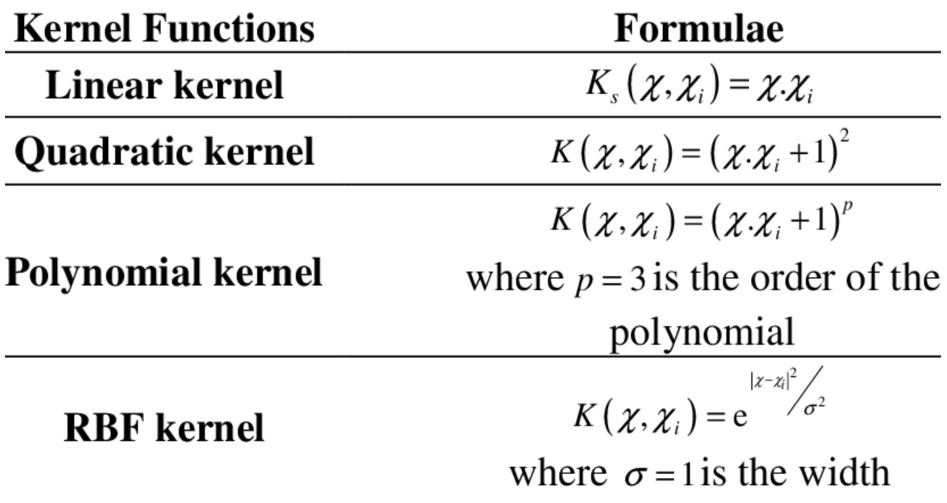
```
In [19]:
from sklearn. linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

import warnings
warnings.filterwarnings("ignore")
```

```
In [20]:
parameters = {'C':np.arange(10,61,10), 'penalty':['l2','l1']}
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_classifier_rs.predict(X_test)
```

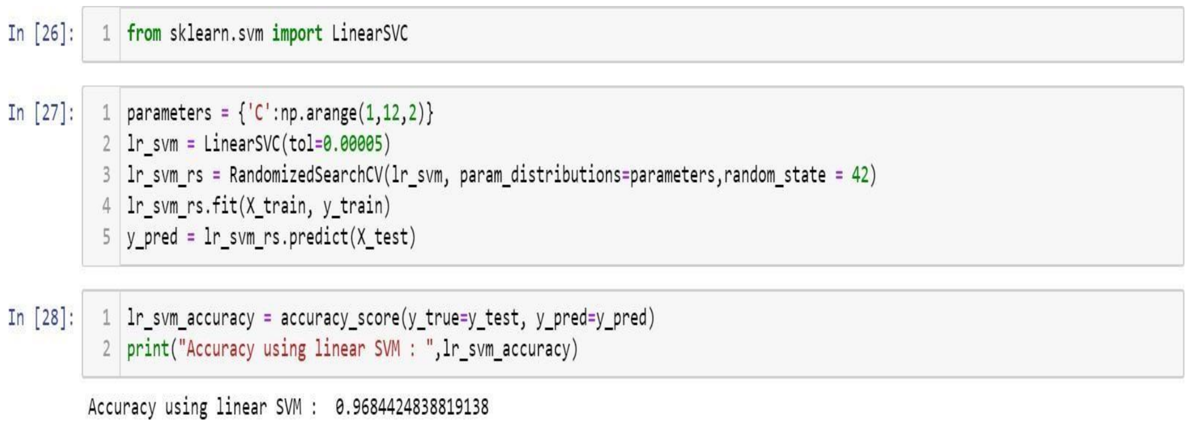
```
In [21]:
lr_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy using Logistic Regression : ", lr_accuracy)

Accuracy using Logistic Regression :  0.9619952494061758
```

2. LINEAR AND KERNEL SVM

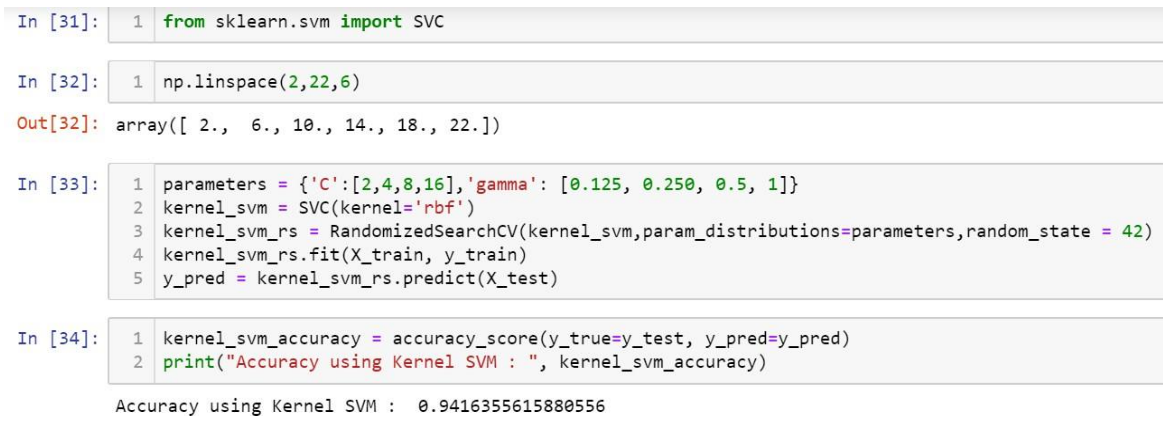
Linear SVM is a parametric model, an RBF kernel SVM isn't, and the complexity of the latter grows with the size of the training set. So, the rule of thumb is: use linear SVMs (or logistic regression) for linear problems, and nonlinear kernels such as the Radial Basis Function kernel for non-linear problems.



Kernel Functions	Formulae
Linear kernel	$K_s(\chi,\chi_i)=\chi{\cdot}\chi_i$
Quadratic kernel	$K(\chi,\chi_i)=(\chi{\cdot}\chi_i+1)^2$
Polynomial kernel	$K(\chi,\chi_i)=(\chi{\cdot}\chi_i+1)^p$ where $p=3$ is the order of the polynomial
RBF kernel	$K(\chi,\chi_i)=\mathrm{e}^{\|\chi-\chi_i\|^2/\sigma^2}$ where $\sigma=1$ is the width



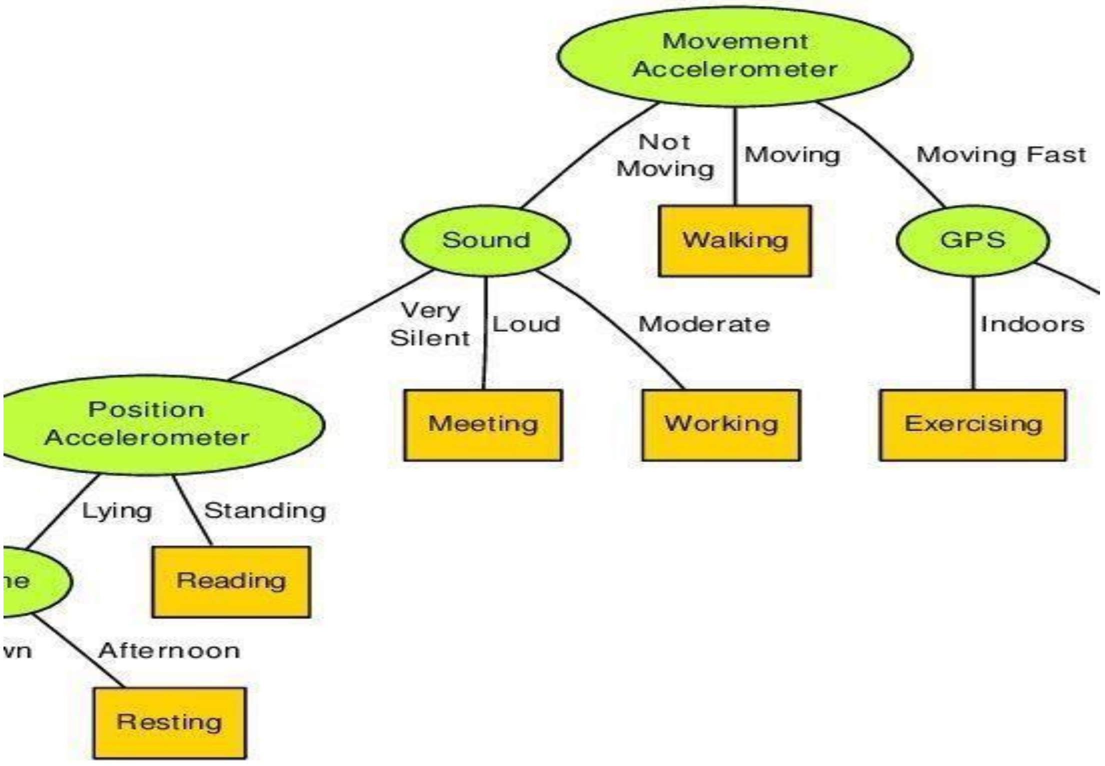
```
In [26]: from sklearn.svm import LinearSVC

In [27]: parameters = {'C':np.arange(1,12,2)}
         lr_svm = LinearSVC(tol=0.00005)
         lr_svm_rs = RandomizedSearchCV(lr_svm, param_distributions=parameters,random_state = 42)
         lr_svm_rs.fit(X_train, y_train)
         y_pred = lr_svm_rs.predict(X_test)

In [28]: 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 :  0.9684424838819138
```

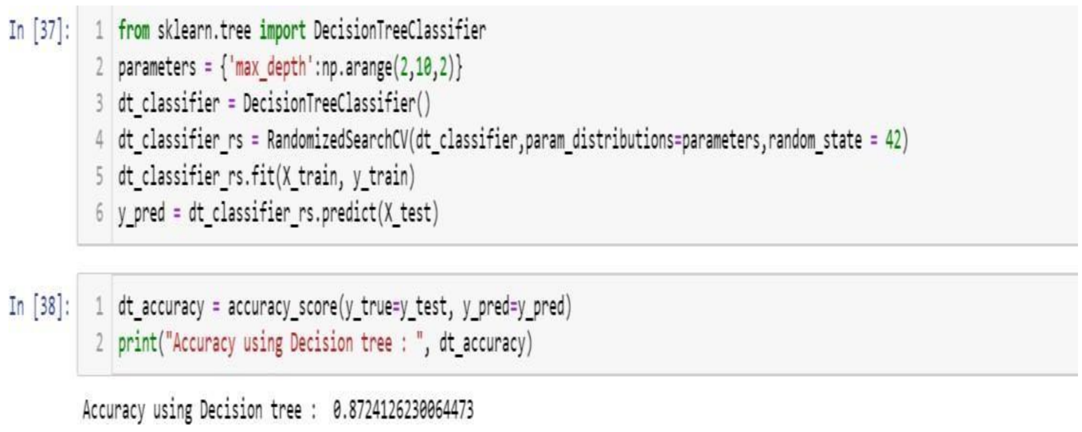


```
In [31]: from sklearn.svm import SVC

In [32]: np.linspace(2,22,6)
Out[32]: array([ 2.,  6., 10., 14., 18., 22.])

In [33]: parameters = {'C':[2,4,8,16],'gamma': [0.125, 0.250, 0.5, 1]}
         kernel_svm = SVC(kernel='rbf')
         kernel_svm_rs = RandomizedSearchCV(kernel_svm,param_distributions=parameters,random_state = 42)
         kernel_svm_rs.fit(X_train, y_train)
         y_pred = kernel_svm_rs.predict(X_test)

In [34]: kernel_svm_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
         print("Accuracy using Kernel SVM : ", kernel_svm_accuracy)

         Accuracy using Kernel SVM :  0.9416355615880556
```

3. DECISION TREE

Decision Tree algorithm belongs to the family of supervised learning algorithms. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data).





```
In [37]: from sklearn.tree import DecisionTreeClassifier
         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)
         y_pred = dt_classifier_rs.predict(X_test)

In [38]: dt_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
         print("Accuracy using Decision tree : ", dt_accuracy)

Accuracy using Decision tree :  0.8724126230064473
```

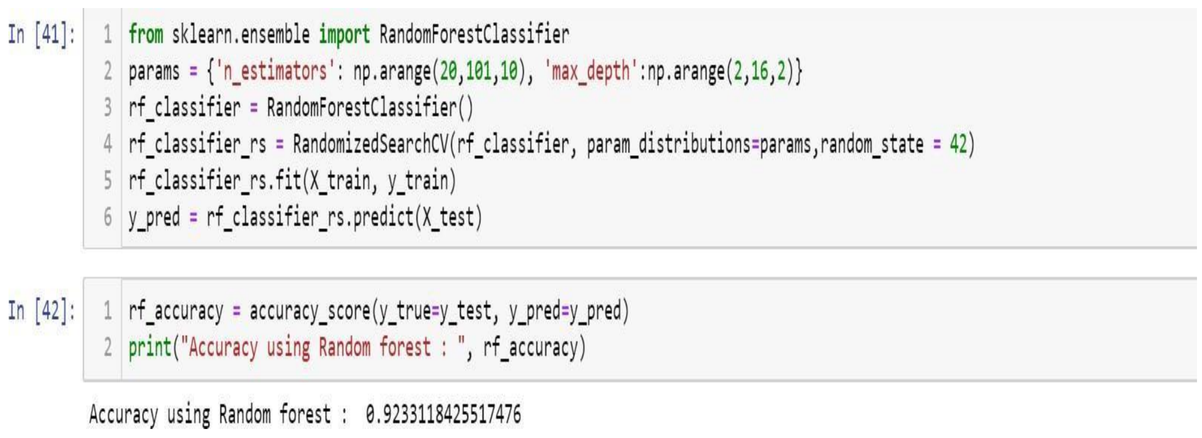
4. RANDOM FOREST

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

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (fi - yi)^2$$

Where N is the number of data points,
fi is the value returned by the model and
yi is the actual value for data point i.

MSE: In statistics, the mean squared error or mean squared deviation of an estimator measures the average of the squares of the errors that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss.



```
In [41]:
from sklearn.ensemble import RandomForestClassifier
params = {'n_estimators': np.arange(20,101,10), 'max_depth':np.arange(2,16,2)}
rf_classifier = RandomForestClassifier()
rf_classifier_rs = RandomizedSearchCV(rf_classifier, param_distributions=params,random_state = 42)
rf_classifier_rs.fit(X_train, y_train)
y_pred = rf_classifier_rs.predict(X_test)
```

```
In [42]:
rf_accuracy = accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy using Random forest : ", rf_accuracy)
```

```
Accuracy using Random forest :  0.9233118425517476
```

7. KEY RESULTS

7.1 Analysis of Results

The accuracy obtained through these models is as follows:

MODELS	ACCURACY
Logistic Regression	96.19995249
Linear SVM	96.84424838
Decision Tree	87.24126230
Random Forest	92.33118420
Kernel SVM	94.16355615

As we know that the algorithm gives more value that is the best accuracy model, by seeing we can clearly state that Linear SVM model has high accuracy than the other models.

8. CONCLUSION

Overall, our list of classifiers achieved relatively high performance. While the various models displayed similar test errors, the accuracy for individual users and specific activities did vary significantly. Sitting was the most difficult activity to classify, often being misclassified as standing, and perhaps having additional features to distinguish sitting from standing could help in this aspect. Since the linear kernel SVM had a higher misclassification rate when an individual was transitioning from standing to sitting, a model that captures the time dependency in the data, such as a hidden markov model, could be useful in this case. However, since the activities in the experiment occurred in a predefined order, a new dataset where the order of activities performed varied between users, and were consistent with how users generally transition between these activities, would be necessary when implementing a hidden markov model.

9. REFERENCES

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.

Anguita, Davide, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine. Vitoria-Gasteiz, Spain: International Workshop.

Bao, Ling and Stephen S. Intille. Activity recognition from user-annotated acceleration data, 2004.

Mannini, A., Sabatini, A.M.: Machine learning methods for classifying human physical activity from on-body accelerometers. Sensors 10(2) (2010) 11541175

Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.

10. APPENDIX

https://github.com/sreeram1999/Human-Activity-Recognition-With-Smartphones