Identification of Human Motion Primitive Activities using Accelerometer Readings

by
Arpit Gupta
University of South Florida

Introduction

This paper focuses on identification of Human Motion Primitive (HMP) activities by using the models built from the data acquired from the Wrist-worn accelerometer. The identification of HMP activities is composed by several stages, including data acquisition, data processing, and data mining models. The data processing includes data cleaning and the feature extraction techniques to define the inputs for the models. This paper also presents a comparison of different types of models in order to choose the type for the implementation of the final model. Results of this project probes that the best accuracies are achieved with Data Mining models with an accuracy higher than 85%.

Triaxial accelerometers are sensors that provide simultaneous measurements in three orthogonal directions, for analysis of all of the vibrations being experienced by a structure. Each unit incorporates three separate sensing elements that are oriented at right angles with respect to each other. These are used to measure the acceleration of the micro movements, allowing the creation of models for identifying the HMP activities. After the development of a model for the identification of HMP, it could be integrated in the creation of a personal digital life coach, important for the monitoring of Alzheimer patients and elderly people, and people with some type of impairment, or for training the people's lifestyle.

Data Description

In order to build models, I have used an openly available and labelled HMP dataset which was the work of the Researchers Barbara Bruno, Fulvio Mastrogiovanni, Antonio Sgorbissa in the Laboratory for Ambient Intelligence and Mobile Robotics DIBRIS, University of Genova. The Dataset is composed of the tri-axial accelerometer recordings of below 14 simple HMP performed by a total of 16 volunteers (11 Male and 5 Female). The data are collected by a single tri-axial accelerometer attached to the right-wrist of the volunteer.

1. brush_teeth: to brush one's teeth with a toothbrush (complete gesture)
2. climb_stairs: to climb a number of steps of a staircase
3. comb_hair: to comb one's hair with a brush (complete gesture)
4. descend_stairs: to descend a number of steps of a staircase
5. drink_glass: to pick a glass from a table, drink and put it back on the table
6. eat_meat: to eat something using fork and knife (complete gesture)
7. eat_soup: to eat something using a spoon (complete gesture)
8. getup_bed: to get up from a lying position on a bed
9. liedown_bed: to lie down from a standing position on a bed
10. pour_water: to pick a bottle from a table, pour its content in a glass on the table and put it back on the table

11. sitdown_chair: to sit down on a chair
12. standup_chair: to stand up from a chair
13. use_telephone: to place a telephone call using a fixed telephone (complete gesture)
14. walk: to take a number of steps

Out of these 14 activities, I have taken only 7 activities (climb_stairs, drink_glass, getup_bed, pour_water, sitdown_chair, standup_chair, walk) which have enough amount of data (see Fig.1) to build a better model for identifying the activities.

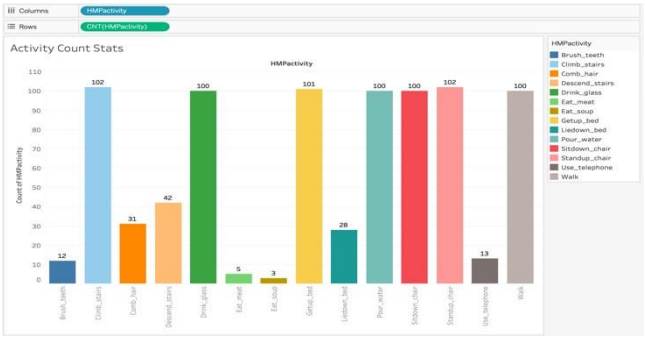


Fig 1. Activities Count Stats

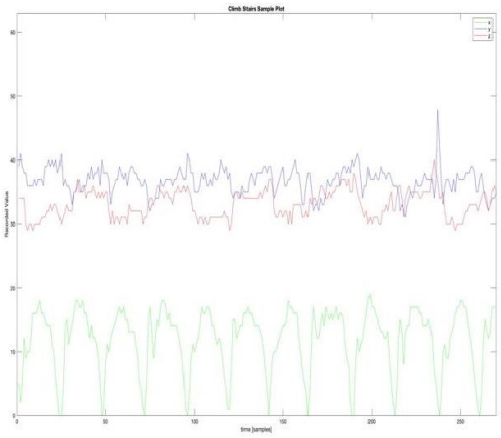
Each file in the dataset corresponds to a volunteer performing an activity for a specific time period. For example, the file "**Accelerometer-2011-03-24-10-24-39-climb_stairs-f1.txt"** refers to an accelerometer recording that was taken on March 24, 2011, starting from 10:24.39 AM. The recording refers to the HMP "climb_stairs" executed by the volunteer with ID "f1". Each raw file consists of three-tab separated columns of data, each corresponding to acceleration readings [0,63] from an accelerometer attached to the right wrist of the user along three axes:

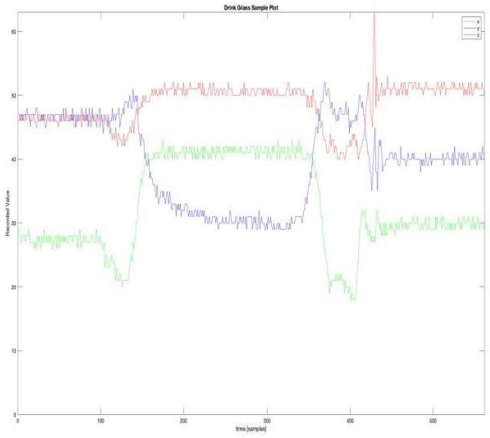
- x axis: pointing towards the hand
- y axis: pointing towards the left
- z axis: perpendicular to the plane of the hand

Exploratory Data Analysis

Extracting feature information was a very challenging task as the raw data had just three variables of acceleration each along the x, y and z axis which was very hard to interpret. So, I had to visualize the data graphically to compare, contrast and investigate underlying patterns in the data.

Below I can see the plots of the acceleration along the three axes corresponding to the Human Primitives.





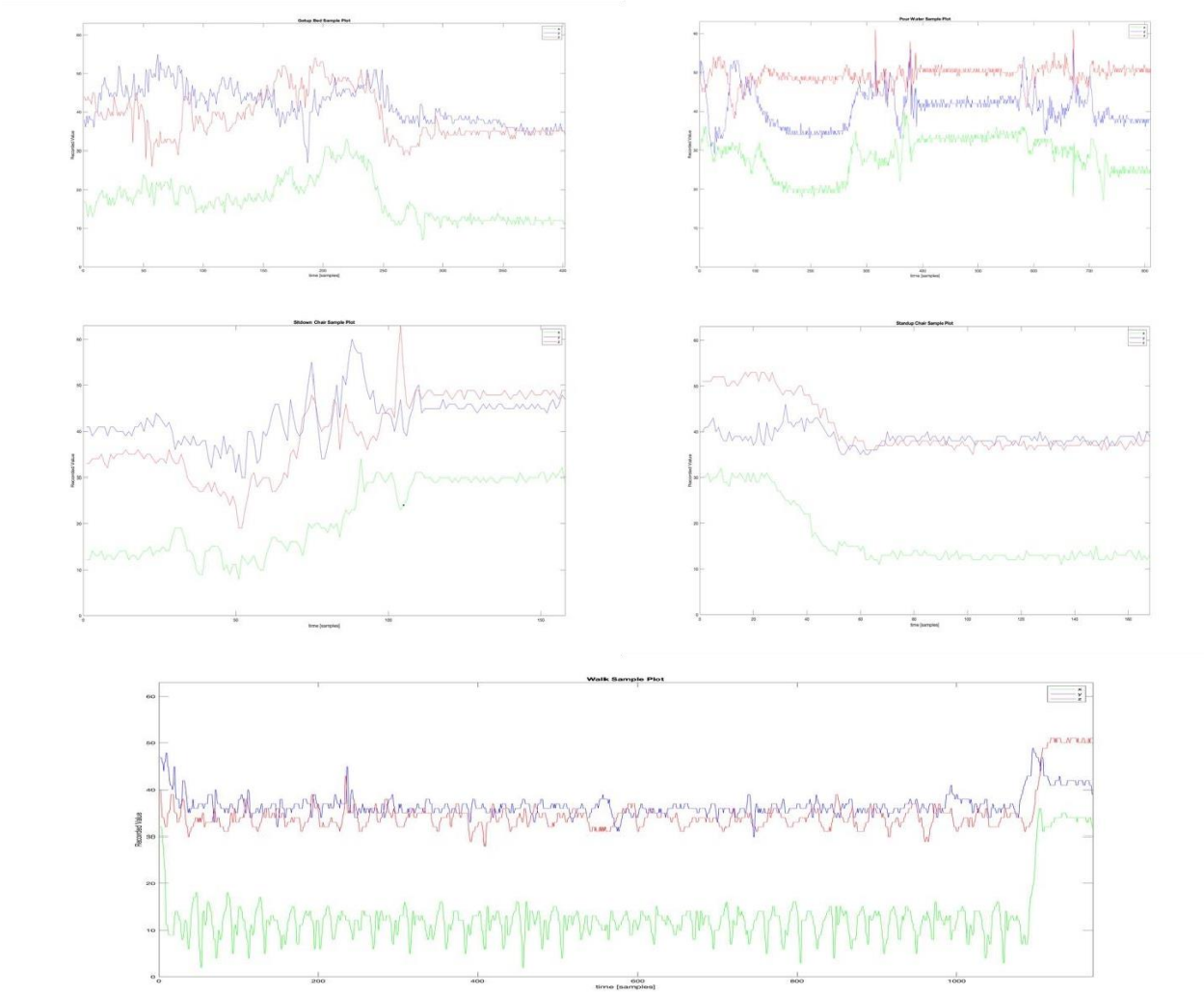


Fig 2. Sample plots for 7 activities

I can see from the above graphs the difference in the changes of accelerations during each activity. I can see a clear distinguishing pattern in the x-axis acceleration amongst all the activities.

Feature Engineering

From the x, y and z patterns observed from the above plots, as they Ire time varying signals,I have derived the following features from each file which can be used for building up the models that identifies the activities.

- **Meanx:** Mean of the recorded values along x-axis
- **Meany:** Mean of the recorded values along y-axis
- **Meanz:** Mean of the recorded values along z-axis
- **Medianx:** Median of recorded values along x-axis
- **Mediany:** Median of recorded values along y-axis
- **Medianz:** Median of recorded values along z-axis
- **Corxy:** Correlation betIen x and y

- **Corxz:** Correlation betIen x and z
- **Coryz:** Correlation betIen y and z
- **Varx:** Variances along x-axis
- **Vary:** Variances along y-axis
- **Varz:** Variances along z-axis
- **Varxy:** Covariances of x and y
- **Varxz:** Covariances of x and z
- **Varyz:** Covariances of y and z
- **Peak Amplitudes of x, y and z:** The time-domain signals do not capture complete information about the signals. A time-domain signal is typically a convolution or a superimposition of a number of waves which operate at different frequencies. To capture this information, I use the Fast-Fourier transform which converts a time domain signal to its respective frequency domain representation.

  As can be seen in Fig.3., the time-domain signal is the waveform that I have seen previously which can be broken down into its component waves with different frequencies.

  So, this representation in the frequency domain thus gives us a lot more information than a simple waveform like amplitude, phase of the signals, etc...

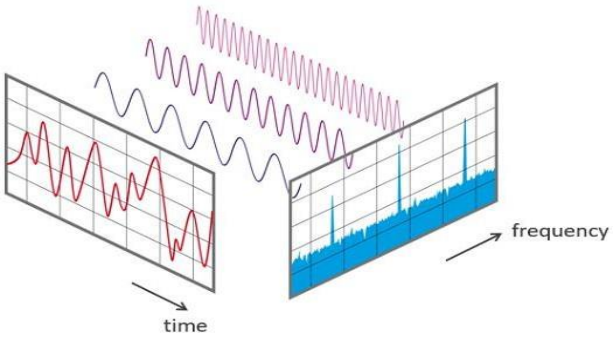


Fig 3. FFT-Time-Frequency-View

From this frequency domain signals, I have created 3 features **fftxmax, fftymax and fftzmax**, which are peak amplitudes of the x, y and z signals respectively.

- **Specenergy:** This feature gives us the spectral energy contained in the signals for performing an HMP activity. So, it is the algebraic sum of the spectral energy contained in x, y and z signals, which has been calculated from the frequency domain representation using the Parseval's Theorem.

- **xbtwnzandy:** This feature gives us the percentage of samples where value of x is betIen values of y and z. This feature is extracted for classifying the drink glass activity, which has a pattern that is different from other activities i.e., z>x>y for some of the samples. This pattern has been observed by plotting graphs for multiple files related to drink glass activity as in Fig 2.

- **xless10:** This feature gives us the percentage of samples where value of x is less than 10. Initially I build a decision tree model and evaluated the performance on test data then there is misclassification for Sitdown_Chair and Standup_Chair with Climb_Stairs, then by observing the plots for various files related to those activities, I have derived feature xless20. But when rebuild model along with this feature and evaluated it then there is a misclassfication for getup_bed with Climb_stairs, then I have changed feature xless20 to xless10 by observing the various plots.

- Along with these I have extracted Volunteer ID (**vol**) and the **HMPactivity** from the file name.

The baseline accuracy of this data is 14.47% (102/705). So, any model that has accuracy more than this baseline is a useful model. As this is a classification data, I have used classifier models like Decision Tree, Gradient Boosting, Random Forest and Support Vector Machines. And as the data is not a skeId data, I have used misclassification rate as the metric for evaluating theperformance of models.

Results

Initially, I have split the featured dataset into train (60%), validation (20%) and test (20%) datasets, then built the models Decision tree, Gradient boosting and Random Forest models with all models having maximum tree depth of 6 (which was tuned as not to overfit on the training data). When compared the models then I found that Random Forest performed Ill as in below statistics.

Fit Statistics
Model Selection based on Valid: Misclassification Rate (_VMISC_)

Selected Model	Model Node	Model Description	Valid: Misclassification Rate	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error
Y	HPDMForest	HP Forest	0.15714	0.010355	0.01914	0.032270
	Boost	Gradient Boosting	0.17857	0.052374	0.10766	0.062415
	Tree	Decision Tree	0.21429	0.025291	0.11483	0.045384

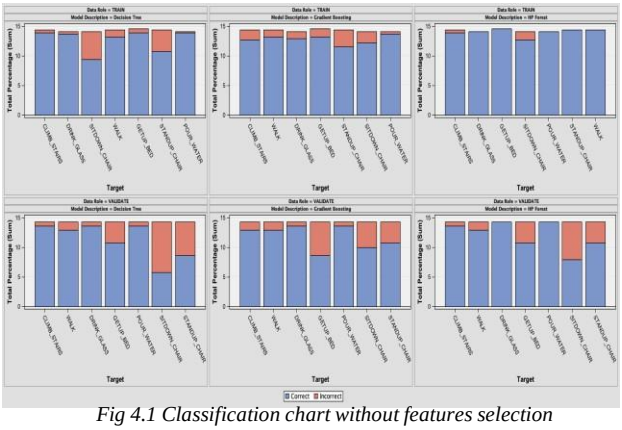


Fig 4.1 Classification chart without features selection

Later, as the features Ire extracted by observing the patterns of the data, I have done feature selection on the featured dataset by using the variable importance in Decision tree. Then I came to know that the features varxy, varxz and xbtwnzandy doesn't have any importance in identification of activities. Now when I built models on this selected dataset then also, I observed that the Random Forest is identifying the activities Ill and it is also better than the previous models that built without any feature selection. Below are the statistics

Fit Statistics
Model Selection based on Valid: Misclassification Rate (_VMISC_)

Selected Model	Model Node	Model Description	Valid: Misclassification Rate	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error
Y	HPDMForest	HP Forest	0.13571	0.012526	0.04067	0.035236
	Boost	Gradient Boosting	0.18571	0.052572	0.10766	0.062750
	Tree	Decision Tree	0.21429	0.032043	0.14354	0.046608

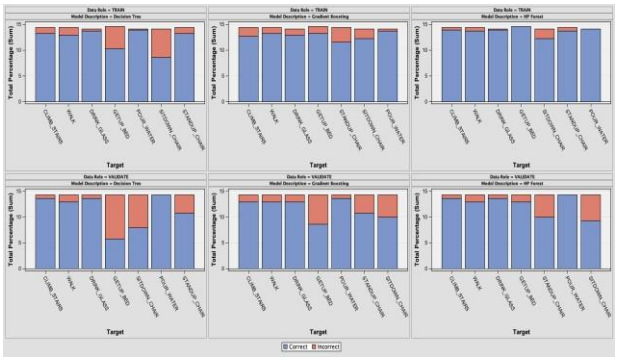


Fig 4.2 Classification chart with features selection

Now I further reduced our features dimension by using the Principal Component Analysis (PCA), which uses an orthogonal transformation to convert a set of correlated features to a set of linearly uncorrelated features. Here I have created 10 Principal components, where in each there are some Iights allocated to each feature as shown in below figure of eigen vectors. Now built the models on this PCA data, the I found that the Gradient Boosting and Decision tree models have more advantage with this PCA data for identifying the activities whereas for Random Forest Model the misclassification rate increased slightly by using the PCA data.

Eigenvectors

Variable	Prin1	Prin2	Prin3	Prin4	Prin5	Prin6	Prin7	Prin8	Prin9	Prin10
corxy	-0.09038	-0.09508	0.16811	0.21166	0.48828	-0.47971	-0.41444	0.45336	-0.06165	-0.03612
corxz	-0.20614	-0.21728	0.27891	-0.13222	-0.12171	0.35087	-0.41950	-0.09438	0.62031	-0.12322
coryz	-0.27545	-0.01580	0.04011	-0.09758	0.51637	0.14468	0.19600	-0.13469	0.07445	0.01155
fftxmax	0.22505	0.39770	0.16417	-0.09683	0.12592	0.12587	0.04428	-0.06914	-0.11595	-0.16364
fftymax	-0.02902	0.50831	0.15906	0.07540	0.02631	0.10361	-0.19001	-0.05055	-0.04998	-0.03148
fftzmax	0.05731	0.51003	0.13047	-0.05975	0.07868	0.10512	-0.20214	-0.00757	-0.03800	-0.08766
meanx	0.38730	-0.08614	0.08503	-0.06200	0.10189	0.09842	0.21415	0.04249	-0.04215	-0.03020
meany	0.06317	-0.08260	0.22487	0.62477	-0.02494	0.19352	0.05335	-0.06796	-0.00137	0.06531
meanz	0.36619	-0.12507	-0.00648	-0.14684	0.13451	0.14913	-0.17977	0.22093	0.11481	0.01322
medianx	0.38324	-0.06535	0.04159	-0.02934	0.12373	0.12730	0.16778	0.03137	0.03356	0.31341
mediany	0.12163	-0.08256	0.20274	0.60990	0.03568	0.08325	0.08310	-0.14432	0.06931	-0.02356
medianz	0.36306	-0.10755	0.00581	-0.13896	0.12823	0.16241	-0.16461	0.19349	0.10170	0.33604
specenergy	-0.01646	0.21411	0.56352	-0.11333	-0.08362	-0.13209	0.06372	0.01530	0.08391	0.34901
varx	-0.10145	-0.15690	0.47949	-0.16108	-0.14216	0.07776	0.42224	0.48080	-0.08903	-0.40478
vary	0.27076	-0.00614	0.15507	-0.10891	0.05014	-0.62213	0.18041	-0.39998	0.43129	-0.15809
varyz	-0.22022	-0.06181	0.02532	-0.09000	0.58753	0.21338	0.23109	-0.20829	0.02536	0.02988
varz	-0.15650	-0.27773	0.36689	-0.19512	-0.10537	-0.08139	-0.16176	-0.31808	-0.46488	0.39795
xless10	-0.28329	0.25066	-0.12488	0.08225	-0.09622	-0.08143	0.31189	0.33855	0.37352	0.51085

Fit Statistics

Model Selection based on Valid: Misclassification Rate (_VMISC_)

Selected Model	Model Node	Model Description	Valid: Misclassification Rate	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error
Y	HPDMForest	HP Forest	0.14286	0.016988	0.03828	0.035958
	Boost	Gradient Boosting	0.14286	0.061617	0.13636	0.064043
	Tree	Decision Tree	0.17143	0.033901	0.15072	0.039882

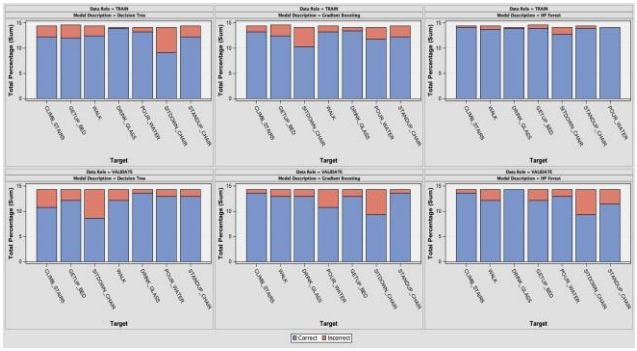


Fig 4.3 Classification chart with PCA data

Also, I have built SVM model on the original featured data without any feature selection and PCA which performed Ill in identifying HMP activities better than the above models. There are 2 ways for multiclass classification in SVM – One-vs-Rest classification and One-vs-One Classification. I have built both models in python using LinearSVC (One-vs-Rest Classifier) and SVC (One-vs-One Classifier) with kernels Linear and Poly with degree 2. Below are the accuracies that I have obtained.

SVM Models	Train Accuracy	Test Accuracy
Linear SVC (one-vs-Rest)	92.19%	90.07%
SVC Linear Kernel (One-vs-One)	95.21%	91.49%
SVC Poly Kernel degree 2 (One-vs-One)	94.68%	92.91%

The confusion matrix for SVM models on test dataset can be seen in below figure. The rows in the figure represent the actual HMP activities while the columns correspond to the HMP activities predicted by our models. The labels for each class are as follows 0:"Climb_stairs", 1:"Drink_glass", 2:"Getup_bed", 3:"Pour_water", 4:"Sitdown_chair", 5:"Standup_chair", 6:"Walk".

	0	1	2	3	4	5	6
0	27	0	0	0	0	0	1
1	0	23	0	1	0	0	0
2	0	0	18	0	2	1	0
3	0	0	0	15	0	0	0
4	0	0	1	0	17	3	0
5	0	0	2	1	0	18	0
6	2	0	0	0	0	0	9

Accuracy: 0.900709219858156

	0	1	2	3	4	5	6
0	27	0	0	0	0	0	1
1	0	23	0	1	0	0	0
2	0	0	19	0	2	0	0
3	0	0	0	15	0	0	0
4	0	0	1	0	18	2	0
5	0	0	3	0	0	18	0
6	2	0	0	0	0	0	9

Accuracy: 0.9148936170212766

	0	1	2	3	4	5	6
0	27	0	0	0	0	0	1
1	0	23	0	1	0	0	0
2	0	0	19	0	2	0	0
3	0	0	0	15	0	0	0
4	0	0	1	0	18	2	0
5	0	0	2	0	0	19	0
6	1	0	0	0	0	0	10

Accuracy: 0.9290780141843972

Fig 5. Confusion Matrices for LinearSVC, SVC Linear Kernel, SVC Poly Kernel with degree 2 Models

As it is evident from the classification charts and confusion matrices, the models Ire very successful in correctly identifying most of HMP activities while it struggled a little bit in identifying the activity among a combination of activities (getup_bed, sitdown_chair and standup_chair) and (climb_stairs and walk) because subset of micro movements in an activity are similar to the other activities in those combinations.

Conclusions

In this project, I looked at an approach to identify the Human Motion Primitive activities by extracting various features by observing the different patterns of the accelerometer data collected for each activity while performing. I have built lots of classifier models out of which the SVM model which built in python is the best model resulted in accurately predicting the correct HMP activity 92.91% of the times as compared to a random choice of 14.47% which can be considered significant. If I integrate this model to a wrist watch then doctors can monitor the results for treating the Alzheimer patients and elderly people, and people with some type of impairment.

References

1. Data Source:
   https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-worn+Accelerometer
2. For Feature Engineering:
   Ivan Miguel Pires, Nuno M. Garcia, Nuno Pombo, Francisco Flórez-Revuelta and Susanna Spinsante, "Pattern Recognition Techniques for the Identification of Activities of Daily Living using Mobile Device Accelerometer"., pp. 2-7
3. Understanding Waveforms:
   https://swphonetics.com/praat/tutorials/understanding-waveforms/
4. Understanding Fourier Transforms:
   https://en.wikipedia.org/wiki/Fast_Fourier_transform
   https://www.youtube.com/watch?v=spUNpyF58BY&t=779s
5. Parseval's Theorem:
   https://en.wikipedia.org/wiki/Parseval%27s_theorem
6. For SVM Multiclass classification:
   https://scikit-learn.org/stable/modules/svm.html

Appendix

1. MATLAB Code for plotting the graphs:

Below code is a function used to plot the Climb_stairs x, y and z patterns in MATLAB

Code:

```
function displayTrial(trial)
% function displayTrial(trial)
%
% displayTrial plots the acceleration recorded in the given accelerometer
% trial. Acceleration data are decoded and filtered with median filtering.
%
% Input:
%  trial --> name of the accelerometer trial to be displayed
%
% Output:
%  ---
%
% Example:
%  trial = 'Climb_stairs\Accelerometer-2011-03-24-10-24-39-climb_stairs-f1.txt';
%  displayTrial(trial);

% READ THE ACCELEROMETER DATA FILE
dataFile = fopen(trial,'r');
disp(dataFile);
data = fscanf(dataFile,'%d\t%d\t%d\n',[3,inf]);

% CONVERT THE ACCELEROMETER DATA INTO REAL ACCELERATION VALUES
% mapping from [0..63] to [-14.709..+14.709]
%noisy_x = -14.709 + (data(1,:)/63)*(2*14.709);
%noisy_y = -14.709 + (data(2,:)/63)*(2*14.709);
%noisy_z = -14.709 + (data(3,:)/63)*(2*14.709);

noisy_x = data(1,:);
noisy_y = data(2,:);
noisy_z = data(3,:);

% REDUCE THE NOISE ON THE SIGNALS BY MEDIAN FILTERING
n = 3;    % order of the median filter
x = medfilt1(noisy_x,n);
y  =  medfilt1(noisy_y,n);
z  =  medfilt1(noisy_z,n);
numSamples = length(x);

% DISPLAY THE RESULTS
time = 1:1:numSamples;
% noisy signal
figure,
   plot(time,noisy_x,'g',time,noisy_y,'b',time,noisy_z,'r');
   axis([0 numSamples 0 63]);
   title('Climb Stairs Sample Plot');
   xlabel('time [samples]');
   ylabel('Recorded Value');
   legend('x','y','z');

% clean signal
figure,
```

```
plot(time,x,'g',time,y,'b',time,z,'r');
axis([0 numSamples 0 63]);
title('Climb Stairs Sample Plot');
ylabel('Recorded Value');
xlabel('time [samples]');
legend('x','y','z');
```

2. R Code for Extracting the features:

Below is the code for extracting the features from each file of each activity and save the featured data into a csv file named “Featured_Data.csv”.

Code:

```
#Change the directory path in below statement to dataset path
setwd("/Users/rohithmovva/Downloads/HMP_Dataset")
dirlist <- list.dirs(recursive = F)[c(2,6,10,13,15,17,20)]
filelist <- sapply(dirlist,function(x) list.files(x))
resdf <- data.frame()
for (i in 1:length(dirlist))
{
 for (j in 1:length(filelist[[i]]))
 {
  file <- paste0(paste0(dirlist[i],"/",filelist[[i]][j]))
  dat <- read.table(file)
  names(dat) <- c("x_acc","y_acc","z_acc")

  cors <- cor(dat)
  variances <- var(dat)
  fftx <- fft(dat$x_acc)
  ffty <- fft(dat$y_acc)
  fftz <- fft(dat$z_acc)

  resdf <- rbind(resdf,data.frame(vol=gsub(".txt","",strsplit(filelist[[i]][j],"-")[[1]][9]),
                                  meanx = mean(dat$x_acc),
                                  meany = mean(dat$y_acc),
                                  meanz = mean(dat$z_acc),
                                  medianx = median(dat$x_acc),
                                  mediany = median(dat$y_acc),
                                  medianz = median(dat$z_acc),
                                  corxy=cors[1,2],
                                  coryz=cors[2,3],
                                  corxz=cors[1,3],
                                  varx=variances[1,1],
                                  vary=variances[2,2],
                                  varz=variances[3,3],
                                  varxy=variances[1,2],
                                  varyz=variances[2,3],
                                  varxz=variances[1,3],
                                  fftxmax=max(abs(Mod(fftx))), #Peak Amplitude of the Fourier Transform
                                  fftymax=max(abs(Mod(ffty))),
                                  fftzmax=max(abs(Mod(fftz))),

specenergy=(sum(Mod(fftx))/length(fftx))+(sum(Mod(ffty))/length(ffty))+(sum(Mod(fftz))/length
(fftz)), #Spectral Energy
                                  xbtwnzandy=mean((dat$z_acc > dat$x_acc & dat$y_acc < dat$x_acc)),
#drinkglasscharacteristic
                                  xless10=sum(dat$x_acc<10)/nrow(dat),
```

```
                    HMPactivity=gsub("./","",dirlist[i])
  ))
 }
}

write.csv(resdf,file="Featured_Data.csv")
```

3. Python Code for SVM Multiclass classification:

Code:

```
import numpy as np
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler #for Scaling the data mostly used for SVMs
from sklearn.svm import LinearSVC

#Change the Path to the Featured_Data.csv file path
acc_data=pd.read_csv("/Users/rohithmovva/Downloads/HMP_Dataset/Featured_Data.csv",
thousands = ',')
acc_data.describe()
acc_data.head()

#Shuffling the dataset
np.random.seed(42)
shuffle_index = np.random.permutation(len(acc_data))
acc_data=acc_data.iloc[shuffle_index]
acc_data.head()

#Splitting the data into train(80%) and test(20%) sets
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(acc_data.iloc[:,2:23],acc_data["HMPactivity"],test_si
ze=.2,random_state=42)
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)

## LinearSVC which uses One-Vs-Rest classification for Multiclass Classification
svm_clf = Pipeline([
      ("scaler", StandardScaler()),
      ("linear_svc", LinearSVC(C=1, loss="hinge", random_state=42)),
  ])

##Confusion Matrix and Training accuracy for LinearSVC
y_pred_t_linear=svm_clf.fit(x_train, y_train).predict(x_train)
from sklearn.metrics import accuracy_score, confusion_matrix
activity={0:"Climb_stairs",1:"Drink_glass",2:"Getup_bed",3:"Pour_water",4:"Sitdown_chair",5:"
Standup_chair",6:"Walk"}
labels=["Climb_stairs","Drink_glass","Getup_bed","Pour_water","Sitdown_chair","Standup_chair
","Walk"]
print(pd.DataFrame(confusion_matrix(y_train.values,y_pred_t_linear,labels=labels),
index=activity, columns=activity))
print('Accuracy: ' + str(accuracy_score(y_train.values, y_pred_t_linear)))

##Confusion Matrix and Test accuracy for LinearSVC
y_pred_linear=svm_clf.fit(x_train, y_train).predict(x_test)
print(pd.DataFrame(confusion_matrix(y_test.values,y_pred_linear,labels=labels), index=activity,
columns=activity))
print('Accuracy: ' + str(accuracy_score(y_test.values, y_pred_linear)))
```

```
## SVC with Kernel Linear uses One-Vs-One Classification
from sklearn.svm import SVC
svm_clf_l = Pipeline([
        ("scaler", StandardScaler()),
        ("svc_l", SVC(kernel="linear",C=1, random_state=42)),
    ])

##Confusion Matrix and Training accuracy for SVC with Linear Kernel
y_pred_t_l=svm_clf_l.fit(x_train, y_train).predict(x_train)
print(pd.DataFrame(confusion_matrix(y_train.values,y_pred_t_l,labels=labels), index=activity,
columns=activity))
print('Accuracy: ' + str(accuracy_score(y_train.values, y_pred_t_l)))

##Confusion Matrix and Test accuracy for SVC with Linear Kernel
y_pred_l=svm_clf_l.fit(x_train, y_train).predict(x_test)
print(pd.DataFrame(confusion_matrix(y_test.values,y_pred_l,labels=labels), index=activity,
columns=activity))
print('Accuracy: ' + str(accuracy_score(y_test.values, y_pred_l)))

## SVC with Kernel Poly with degree 2 uses One-Vs-One Classification
poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=2, coef0=1, C=1))
    ])

##Confusion Matrix and Training accuracy for SVC with Poly Kernel with degree 2
y_pred_t_poly=poly_kernel_svm_clf.fit(x_train,y_train).predict(x_train)
print(pd.DataFrame(confusion_matrix(y_train.values,y_pred_t_poly,labels=labels),
index=activity, columns=activity))
print('Accuracy: ' + str(accuracy_score(y_train.values, y_pred_t_poly)))

##Confusion Matrix and Test accuracy for SVC with Poly Kernel with degree 2
y_pred_poly=poly_kernel_svm_clf.fit(x_train,y_train).predict(x_test)
print(pd.DataFrame(confusion_matrix(y_test.values,y_pred_poly,labels=labels), index=activity,
columns=activity))
print('Accuracy: ' + str(accuracy_score(y_test.values, y_pred_poly)))
```