HIMANSHU SINGHAL

115891944 | MINI PROJECT 1 | ENSE698E | SENSORS AND SYSTEMS

PROBLEM STATEMENT

1. **Spectrogram:** First, explore the spectral characteristics of the data. Familiarize yourself with the database by examining the spectrogram for several of the datafiles. Explore what happens as you use different parameters – different windows, different spectral and temporal resolution. Explain the role of windowing in forming a spectral estimate, using spectrograms of the data formed using different windowing parameters to illustrate your answer. Provide the code used to create the spectrograms, the images generated, and parameters used, along with an explanation of windowing.

2. **Filter:** Next, explore the transient characteristics of the data. Design a finite impulse response filter to process the data. Use a bandpass filtering characteristic, choosing the Highpass and lowpass corners based on your observations from the spectrographic analysis above – for example, allowing frequencies from 2 Hz to 8 Hz to pass through while suppressing frequencies outside these ranges. Determine the magnitude and phase transfer characteristics of your filter as a function of frequency. Apply the filter to the transient data. Provide the code, the detailed filter design, magnitude/phase vs frequency, and 1-2 comparisons of raw vs processed data.

3. **Classify:** Design and train a classifier to discriminate between data from walking, climbing, and descending conditions. What is the detailed structure of your classifier, and what procedures have you followed to train it on the dataset? Provide the code used for constructing and training the classifier. Evaluate the performance on your training set and on a test set that was not used for training. Provide the code, a graphical representation of the classifier architecture, and test results on the training and test data.

4. **Analysis:** Analyse the detection failures – the false negatives and false positives. How can you tune the classifier in order to shift the relative balance between false positive and false negative detections? Discuss these trade-offs and explore how they relate to the performance of the classifier that you have designed.

GIVEN:

Data (ASCII files) and a MATLAB code (*displayModel*) is provided by the paper titled “Analysis of human behavior recognition algorithms based on acceleration data” presented in IEEE International Conference of Robotics and Automation. Although, the complete dataset contains 14 activities and 16 volunteers, focus of this report is limited to three activities - Climb stairs, walk and Descend Stairs and 3 volunteers- 1 female (f1) and 2 males (m1 & m2).

TABLE OF CONTENTS

1. SPECTROGRAM:

1.1 Introduction

A spectrogram is a visual way of representing the signal strength or loudness of a signal over time at various frequencies present in a waveform. It helps to see how energy (power) levels vary over time. Spectrograms are two dimensional graphs with a third dimension represented by colors. Time runs from left to right on the horizontal axis while the vertical axis represents frequency. The third-dimension color represents the energy or power of a frequency where dark color corresponds to low amplitudes or power while brighter colors represent stronger amplitudes or energy.

Spectrogram displays 2 main characteristics which are of interests in this report-

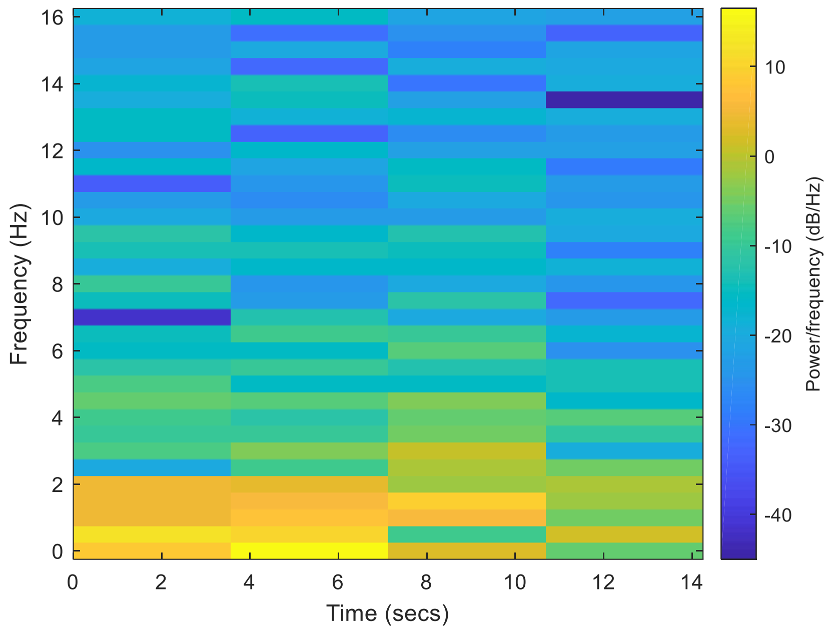
- Spectral Resolution
  Spectral resolution is the ability to resolve spectral features and bands into their separate components. Spectral resolution of a spectrograph is defined as the ability to distinguish between two wavelengths separated by a small amount
- Temporal Resolution
  Temporal Resolution refers to the precision of a measurement with respect to time. Often there is a tradeoff between Temporal Resolution and the Spectral resolution. It depends on the order of grating, wavelength and number of grooves illuminated.

Spectrogram function in MATLAB – spectrogram (X, WINDOW, NOVERLAP, F, Fs)
It can be seen from the function that a spectrogram depends on the following factors -

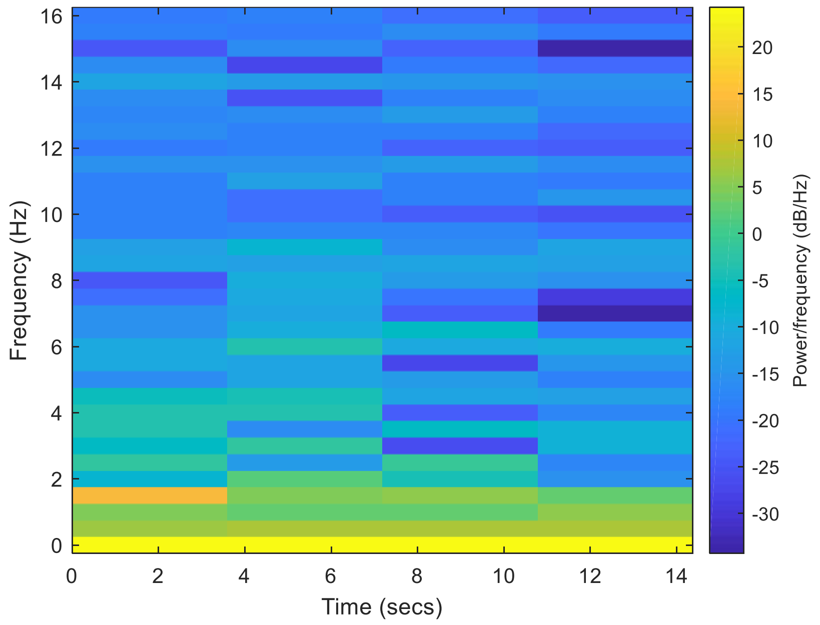
- Input (X)
  The input data for which the spectrogram is being plotted. The data should be continuous to plot an optimal spectrogram.
- Window
  Windowing is a process of shaping a signal before its spectral analysis. There exist multiple types of window but for this report, only 3 types of windowing viz. Kaiser, Hamming and Flat top were used.
- % Overlap (Noverlap)
  It is the amount of overlap between two adjoining segments/samples. Normally, a small % of overlap is considered to avoid any aliasing which leads to loss of information.
- Frequency (F)
  Operating/cyclical frequency of the spectrogram
- Sampling Frequency (Fs)
  It is the rate at which the samples are observed i.e. number of samples per unit time.

Below are some of the examples of the spectrogram plotted using the spectrogram function and the code provided in *displayModel.* All the spectrograms are plotted using the values along the **x axis** only.
The following parameters were used to plot the below spectrograms –

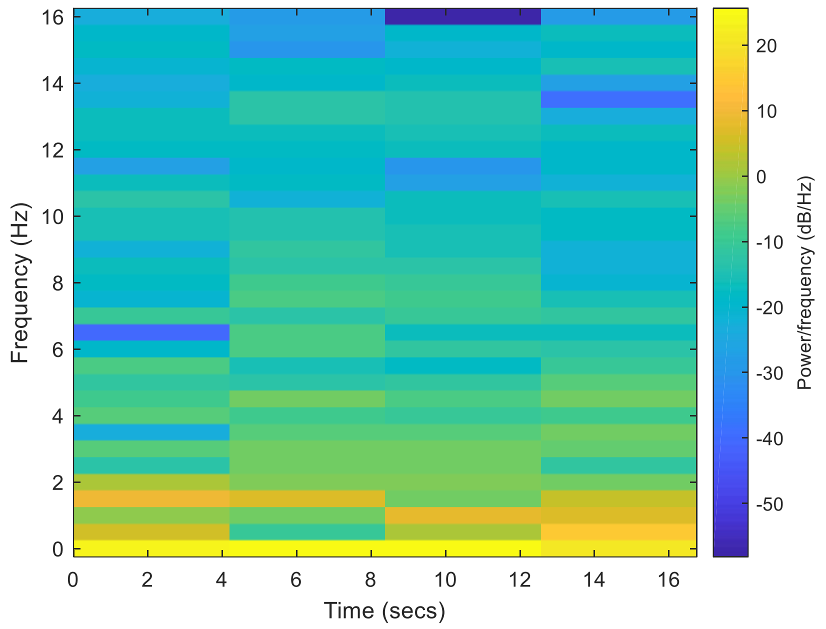
Window	Kaiser
Sampling Frequency	32Hz
Beta	1
Length	Varying
Frequency	64Hz



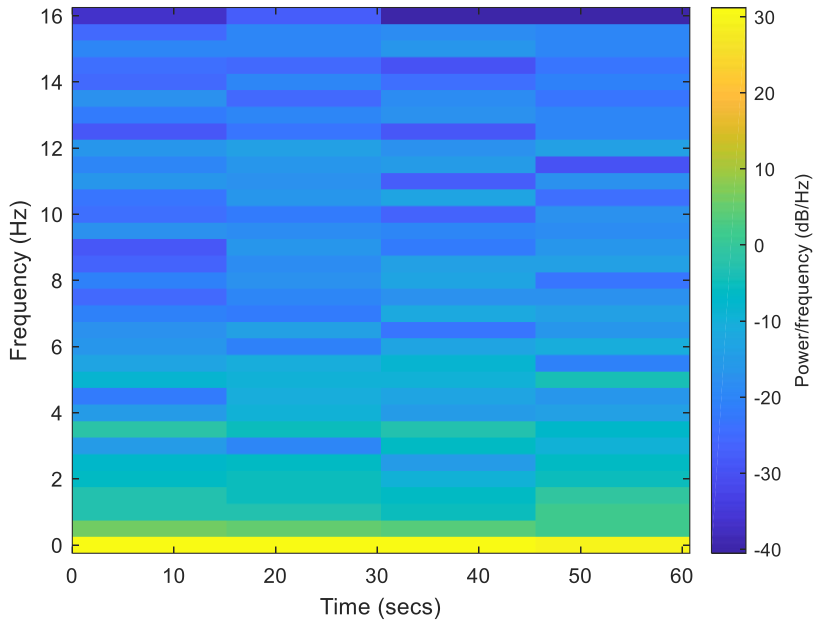
Accelerometer-2011-05-30-22-00-32-climb_stairs-m2



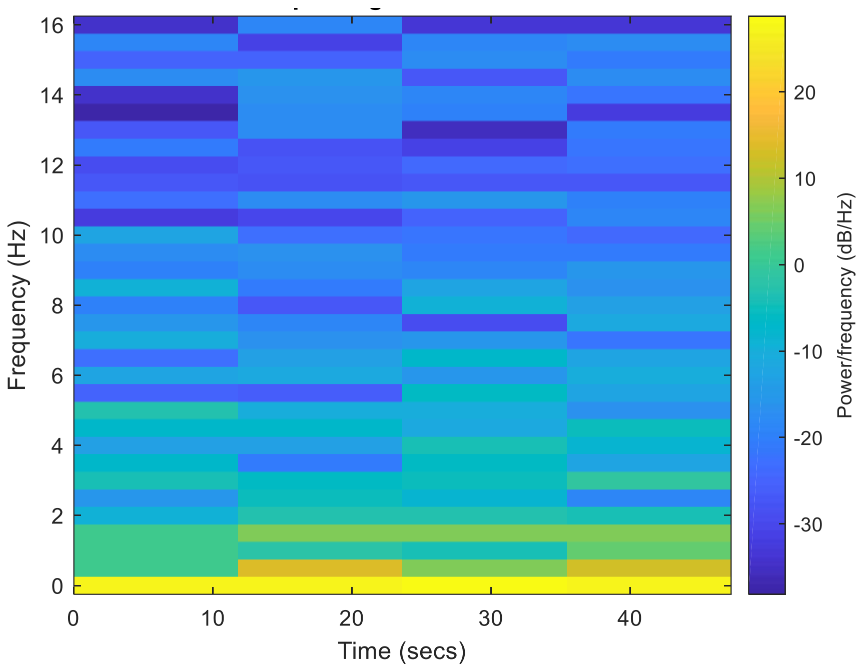
Accelerometer-2011-05-30-10-20-47-climb_stairs-m1



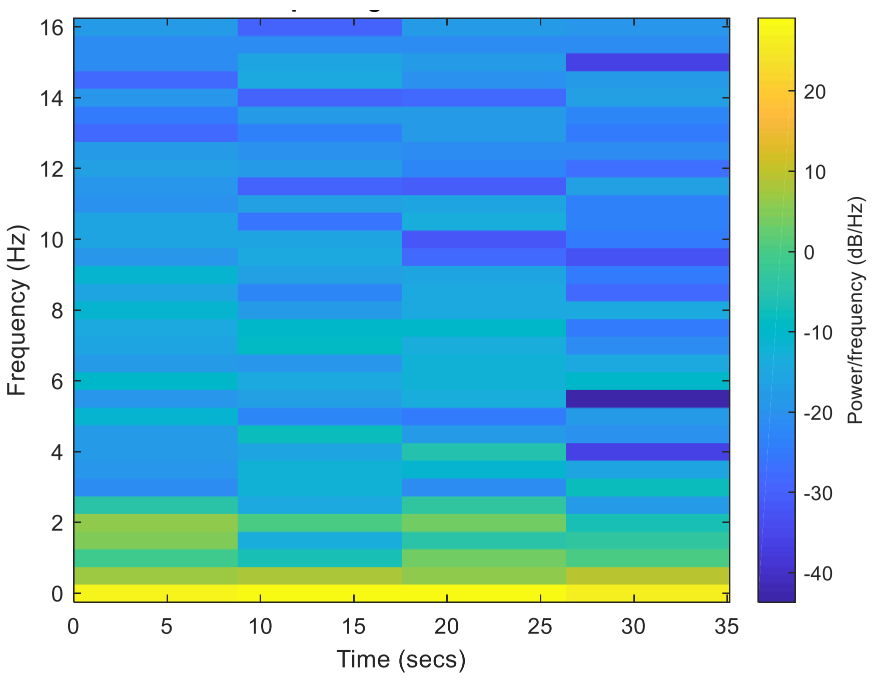
Accelerometer-2011-04-11-11-58-30-climb_stairs-f1



Accelerometer-2011-04-05-18-30-57-walk-f1



Accelerometer-2011-05-30-21-39-55-walk-m2

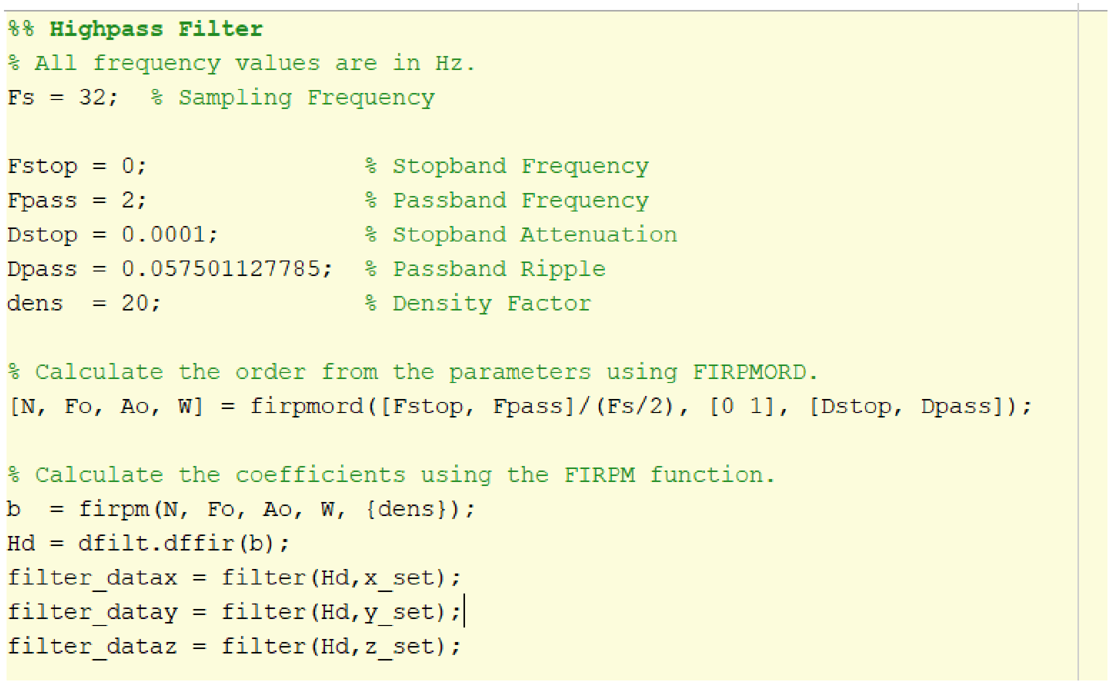


Accelerometer-2012-06-11-11-38-12-walk-m1

It can be observed from the above spectrograms that a common error exists in each of them. There exists a very high power or amplitude signal at 0 frequency indicating the presence of noise or an unwanted signal. This signal should be eliminated to proceed further with the report using a high pass filter

1.2 Highpass Filter

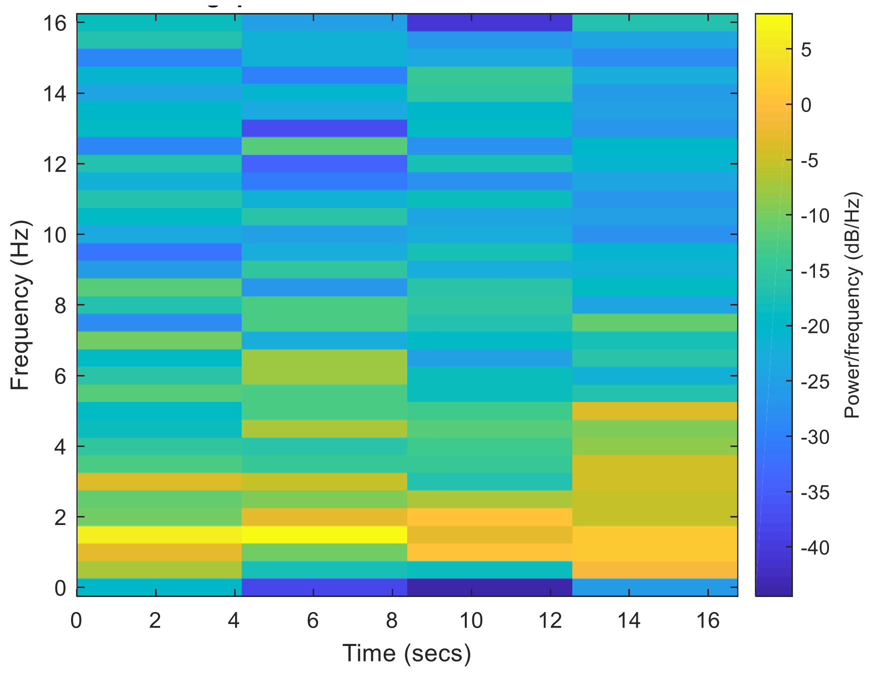
High pass filter is a type of filter that blocks any signal below a set desired frequency and passes all the signal above that set desired frequency. For this report, any signal below 2 Hz is blocked considering the frequency of the activities performed by a normal human being. The output of this high pass filter is used to further progress with the report and plot the spectrograms. Code for the same is shown below,



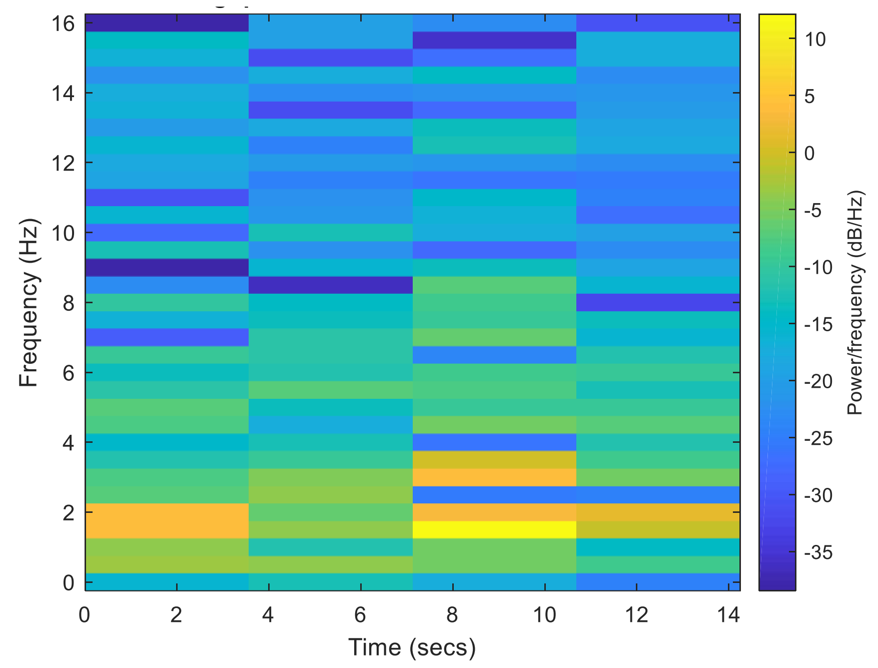
```
%% Highpass Filter
% All frequency values are in Hz.
Fs = 32;  % Sampling Frequency

Fstop = 0;                % Stopband Frequency
Fpass = 2;                % Passband Frequency
Dstop = 0.0001;           % Stopband Attenuation
Dpass = 0.057501127785;   % Passband Ripple
dens  = 20;               % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fstop, Fpass]/(Fs/2), [0 1], [Dstop, Dpass]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);
filter_datax = filter(Hd,x_set);
filter_datay = filter(Hd,y_set);
filter_dataz = filter(Hd,z_set);
```

Code for the Highpass Filter

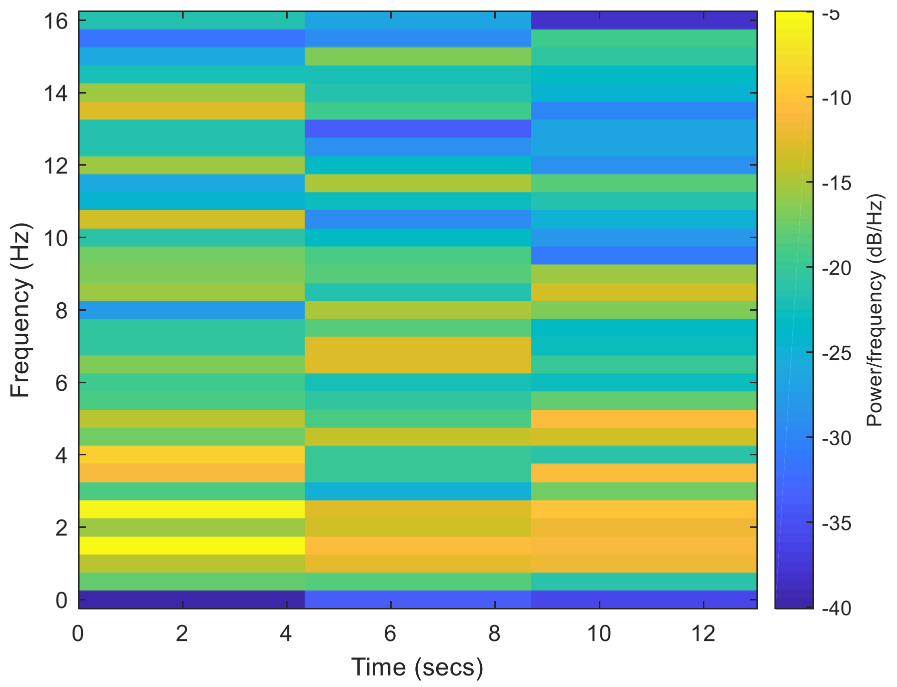
Spectrogram using the output of the high pass filter as an input was plotted for an activity performed by 3 different human beings. It can be observed that the DC noise is eliminated from the spectrograms and it now optimally represents the task being performed.



Accelerometer-2011-04-11-11-58-30-climb_stairs-f1



Accelerometer-2011-05-30-22-00-32-climb_stairs-m2



Accelerometer-2012-05-28-17-56-40-climb_stairs-m1

1.3 Windowing

Windowing is a process of shaping a signal by not only truncating it but by balancing the sidelobes and main lobes so that they transition smoothly. It reduces the errors when frequency content of a signal is computed by taking a limited duration snapshot which lasts for a longer time.

When a signal is truncated, we are altering the frequency domain in a rather surprising way where we end up convolving (i.e. smearing) all frequency terms with a "window" function. If nothing other than truncating is done, then that function becomes trivial which spreads the frequency content of the original signal over the entire spectrum, and if there is a dominant component, then everything else gets buried by it. The shorter the blocks, the worse the effect is as the window gets fatter in the frequency domain. Windowing with shaped window, such as Hamming, Hanning or Kaiser, alters the frequency domain response, making the smearing clearer as a result. The following function was used to plot the spectrogram -

spectrogram (input, kaiser (L, Beta), NOVERLAP, F, Fs)

1.3.1 Kaiser Window

It is one of the windowing functions used in FIR filter and spectral analysis. It has 2 parameters

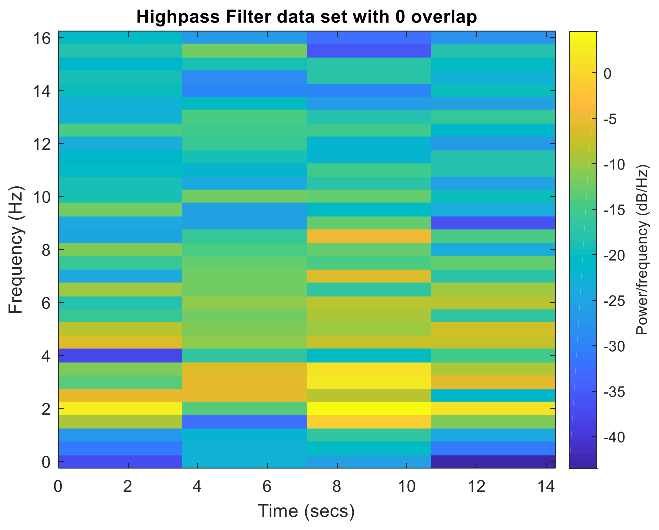
- L – Length of the kaiser window (frequency points per segment)
- Beta – tradeoff between sidelobes and main lobes. Larger beta value increases the width of the main lobe thus reducing the temporal resolution.

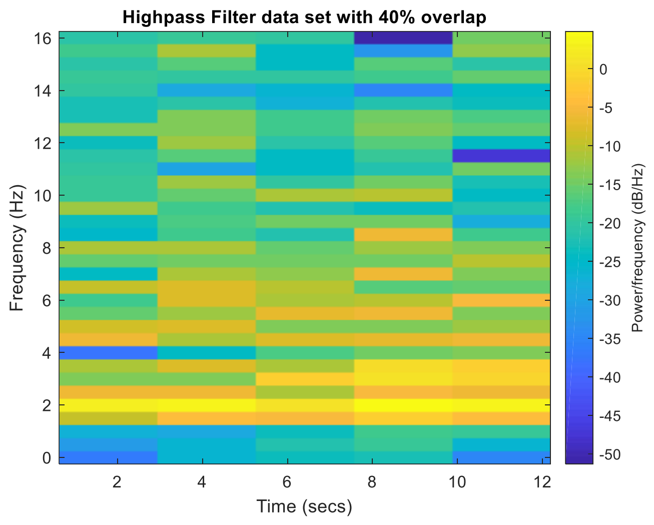
Plotting spectrograms with varying parameters using the Kaiser window:

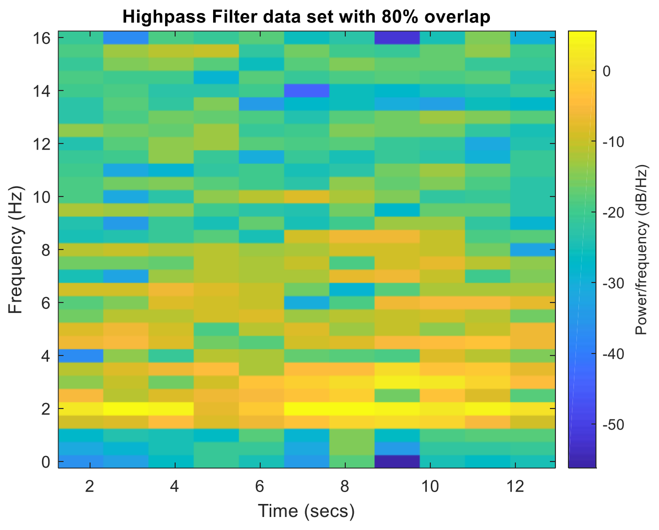
A. KAISER WINDOW WITH CHANGING % OVERLAP

```
spectrogram(filter_data,kaiser(114,1),0,64,32,'yaxis')
title('Highpass Filter data set with 0 overlap')
figure(2),
spectrogram(filter_data,kaiser(114,1),40,64,32,'yaxis')
title('Highpass Filter data set with 40% overlap')
figure(3),
spectrogram(filter_data,kaiser(114,1),80,64,32,'yaxis')
title('Highpass Filter data set with 80% overlap')
```

Parameters with changing % overlap – 0%, 40% and 80%	
Window	Kaiser
Length (L)	114
Frequency (F)	64
Sampling Frequency (Fs)	32
Beta	1







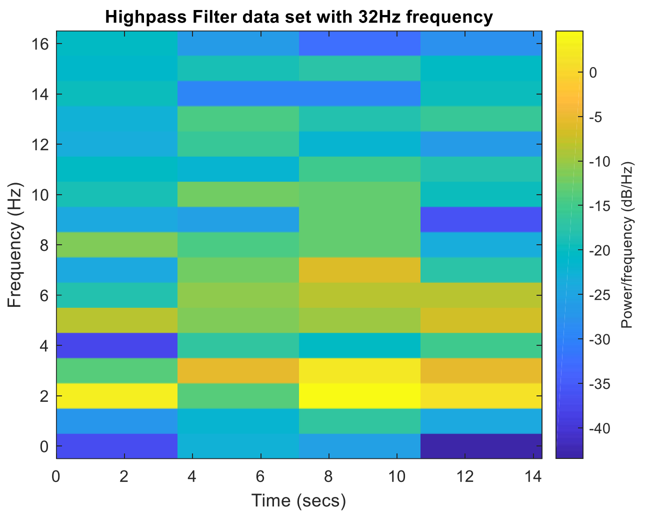
Interpretation

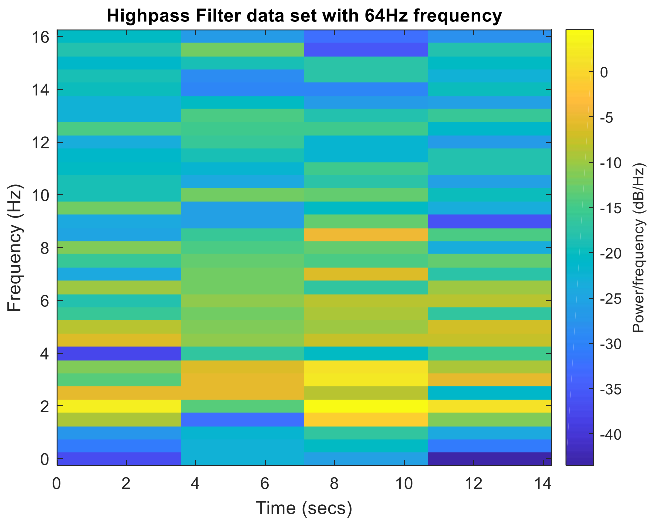
It can be observed from these spectrograms that increase in % overlap increases the number of segments. There exists an overlap with the adjoining segment/samples thus increasing the spectral resolution and enhancing temporal resolution.

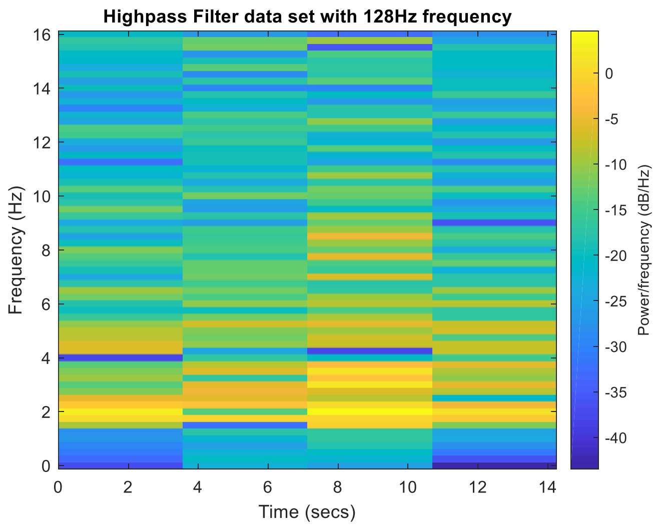
B. KAISER WINDOW WITH CHANGING FREQUENCY (F)

```
spectrogram(filter_data,kaiser(114,1),0,32,32,'yaxis')
title('Highpass Filter data set with 32Hz frequency')
figure(2),
spectrogram(filter_data,kaiser(114,1),0,64,32,'yaxis')
title('Highpass Filter data set with 64Hz frequency')
figure(3),
spectrogram(filter_data,kaiser(114,1),0,128,32,'yaxis')
title('Highpass Filter data set with 128Hz frequency')
```

Parameters with changing Frequency (F)	
Window	Kaiser
Beta	1
% overlap	0
Sampling Frequency (Fs)	32
Length (L)	114





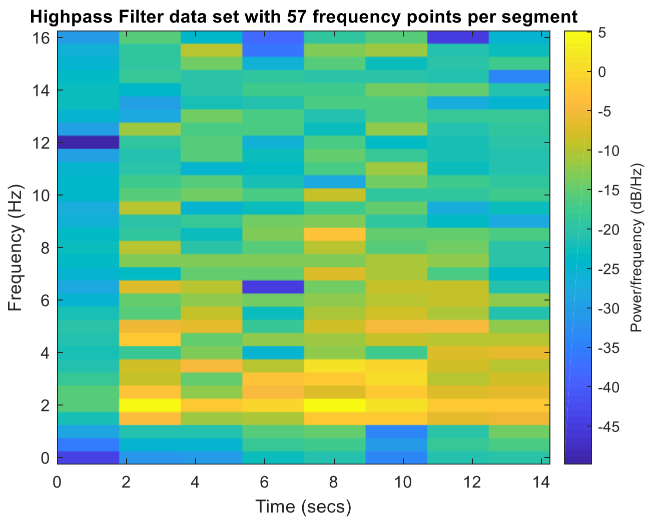


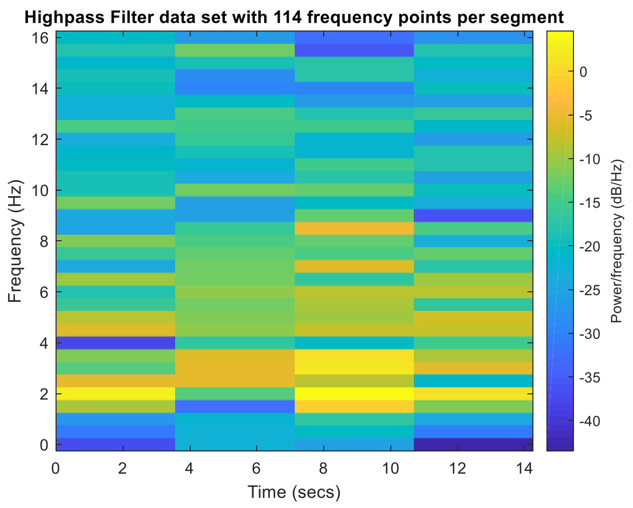
Interpretation

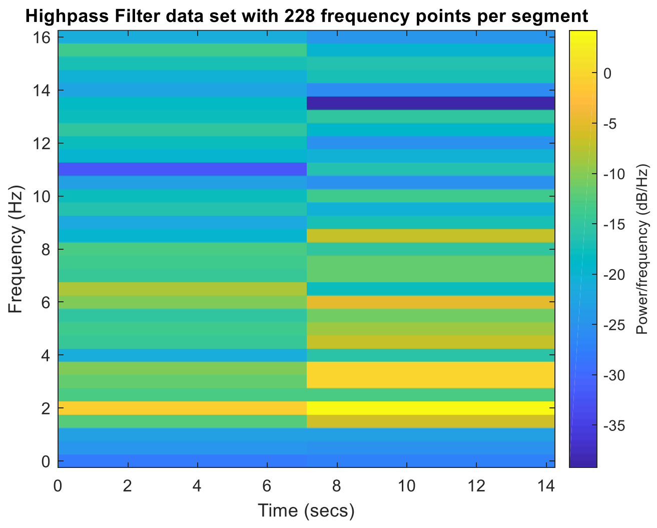
It can be observed from these spectrograms that increase in Frequency increases the frequency resolution or results in a finer data for a given frequency. Increase in frequency increases the temporal resolution and much more data can be obtained for a given frequency.

C. KAISER WINDOW WITH CHANGING LENGTH OF THE WINDOW (L)

Parameters with changing Length (L)	
Window	Kaiser
Beta	1
% overlap	0
Sampling Frequency (Fs)	32
Frequency (F)	64







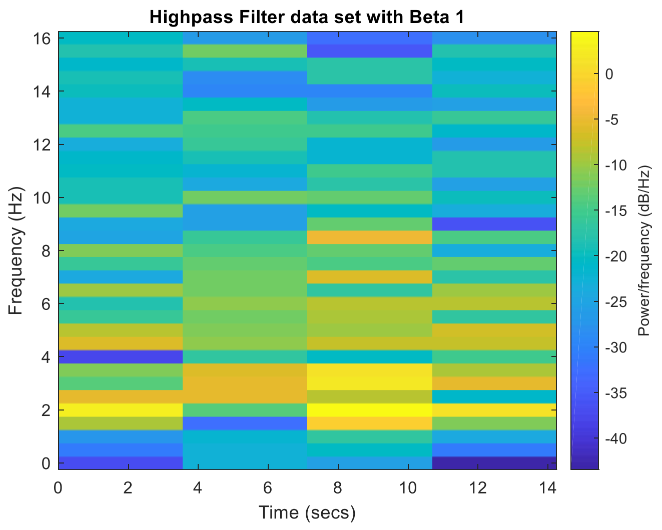
Interpretation

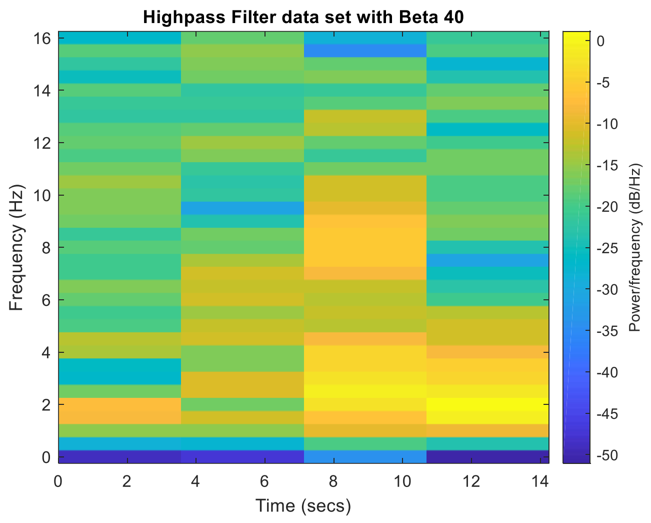
It can be observed from these spectrograms that increase in Frequency points per segment (length) decreases the number of segments at same sampling frequency. A normal spectrogram divides a signal into 8 equal segments. Therefore, increase in frequency points per segment decreases spectral resolution.

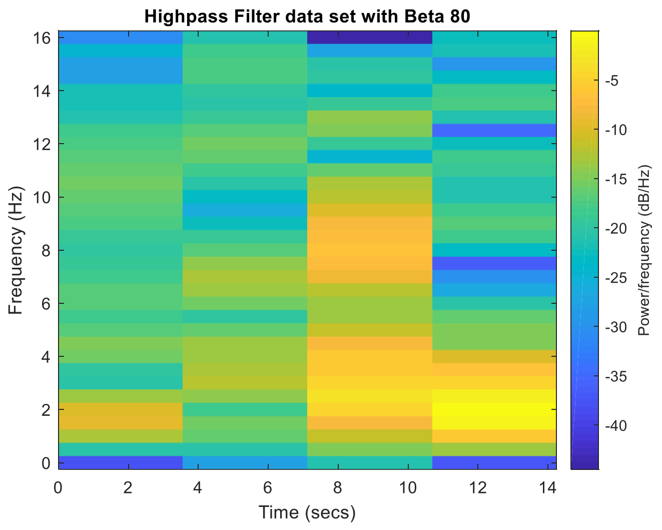
D. KAISER WINDOW WITH CHANGING BETA

```
spectrogram(filter_data,kaiser(114,1),0,64,32,'yaxis')
title('Highpass Filter data set with Beta 1')
figure(2),
spectrogram(filter_data,kaiser(114,40),0,64,32,'yaxis')
title('Highpass Filter data set with Beta 40')
figure(3),
spectrogram(filter_data,kaiser(114,80),0,64,32,'yaxis')
title('Highpass Filter data set with Beta 80')
```

Parameters with changing Length (L)	
Window	Kaiser
Length (L)	114
% overlap	0
Sampling Frequency (Fs)	32
Frequency (F)	64

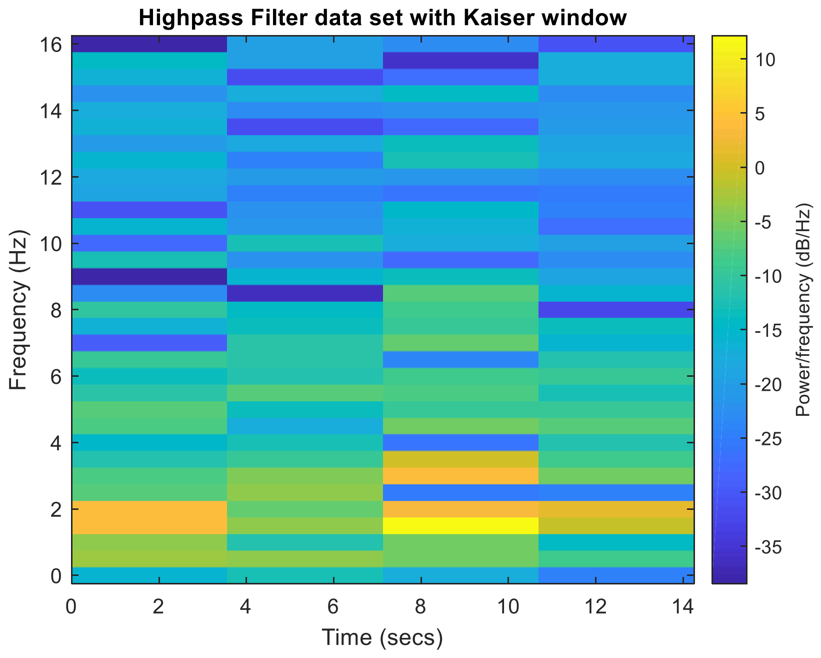






Interpretation

It can be observed from these spectrograms that beta affects the sidelobe attenuation of the Fourier transform of the window. Increase in Beta increases the decreases the amplitude of the sidelobes, thus reducing frequency resolution.



After changing all the parameters for the Kaiser window, it can be concluded that the optimal spectrogram can be plotted with the following parameters as shown :

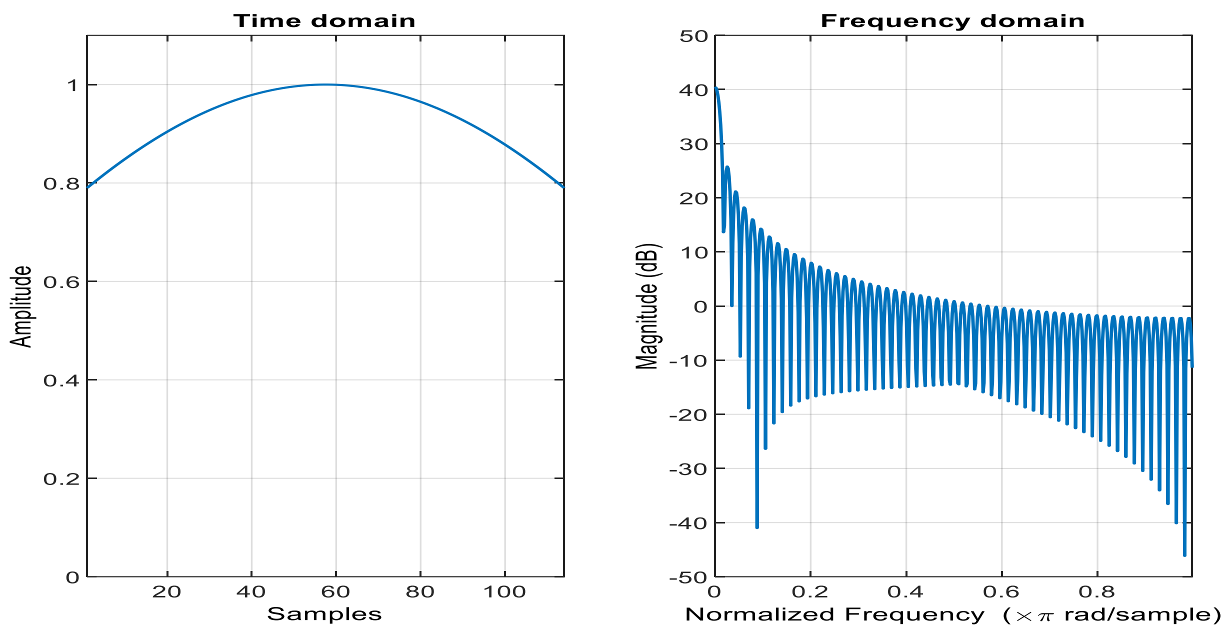
L = 114

B = 1

F = 64

Fs = 32

% overlap = 0



Transient characteristics of the Kaiser window for the above-mentioned parameters

1.3.2 Flat top Window

Flat top window has two deciding parameters, one of which is fixed here because this report only deals with FFT, therefore only symmetric is used for flat top window.

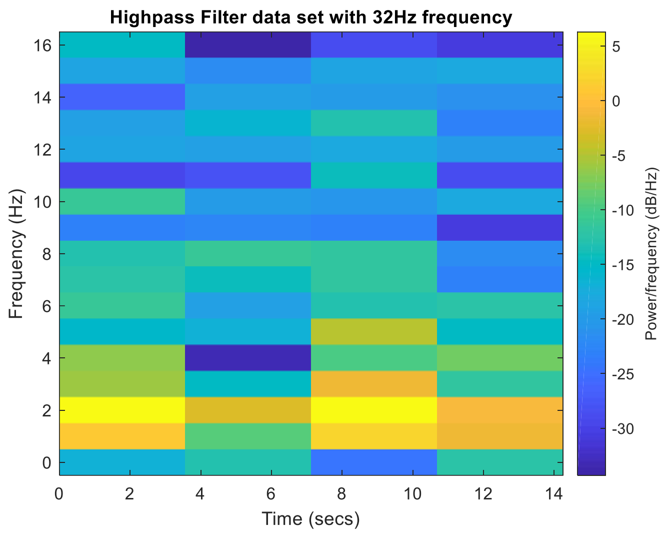
- L – Length of the window
- Flag – symmetric or periodic

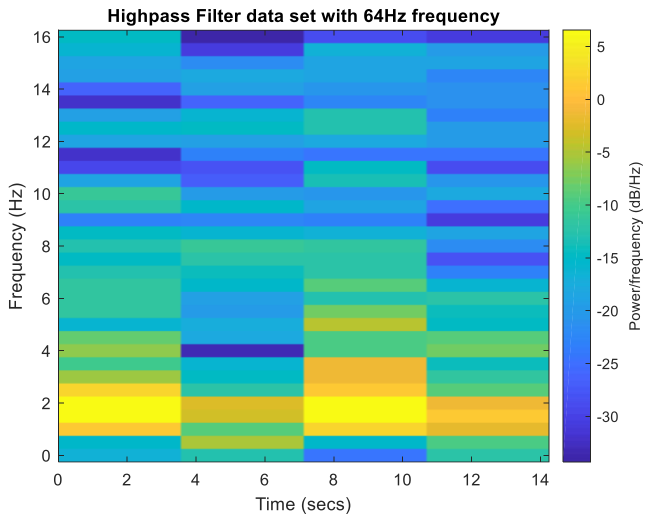
Function - spectrogram (input, flattopwin (L), 0, F, Fs)

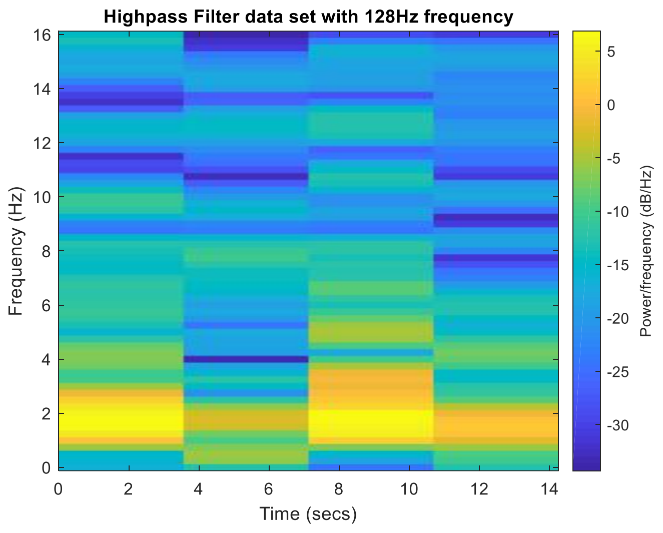
Plotting spectrograms with varying parameters using the flat top window:

A. FLATTOP WINDOW WITH CHANGING FREQUENCY (F)

Parameters with changing Frequency (F)	
Window	Flat top
Length (L)	114
% overlap	0
Sampling Frequency (Fs)	32
flag	symmetric







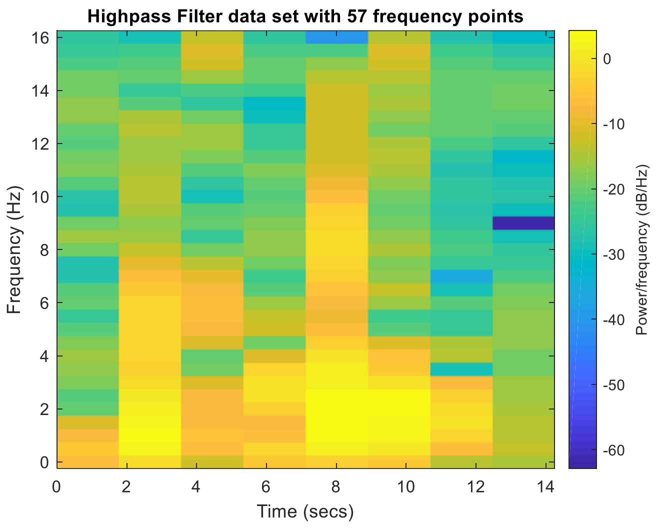
Interpretation

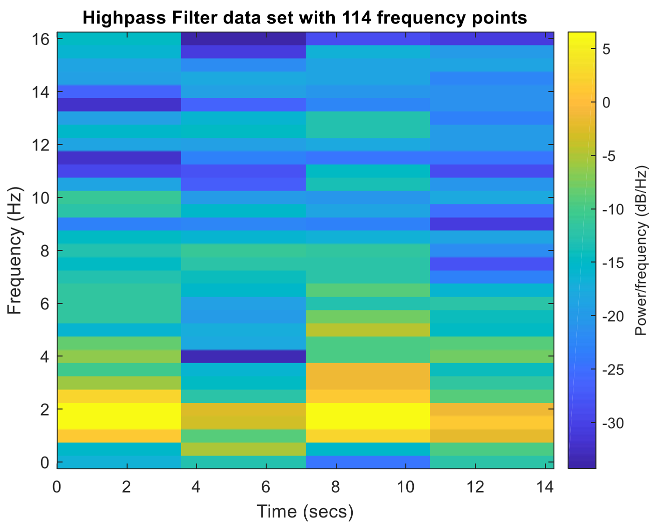
It can be observed from these spectrograms that irrespective of the window used, changing frequency increase the temporal resolution of the spectrogram i.e. the spectrogram has more information at a given frequency and the information of any 2 adjoining frequency levels can be easily distinguished. However, in this case, it becomes blur at higher frequency.

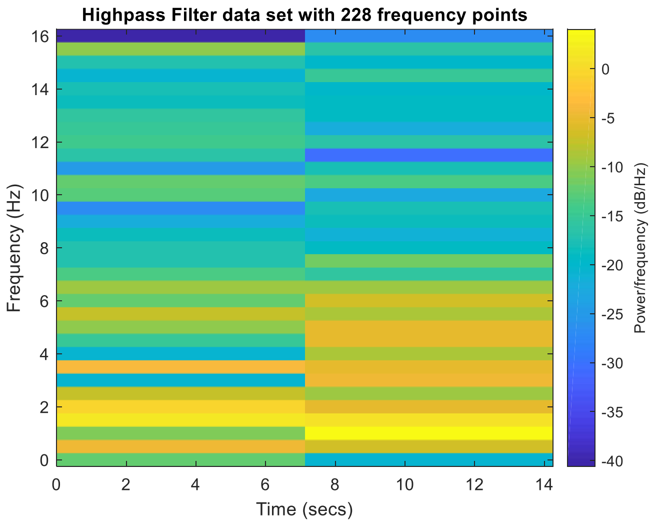
B. FLATTOP WINDOW WITH CHANGING LENGTH (L)

```
spectrogram(filter_data,flattopwin(57),0,64,32,'yaxis')
title('Highpass Filter data set with 57 frequency points')
figure(2),
spectrogram(filter_data,flattopwin(114),0,64,32,'yaxis')
title('Highpass Filter data set with 114 frequency points')
figure(3),
spectrogram(filter_data,flattopwin(228),0,64,32,'yaxis')
title('Highpass Filter data set with 228 frequency points')
```

Parameters with changing Length (L)	
Window	Flat top
Frequency	64
% overlap	0
Sampling Frequency (Fs)	32
flag	symmetric

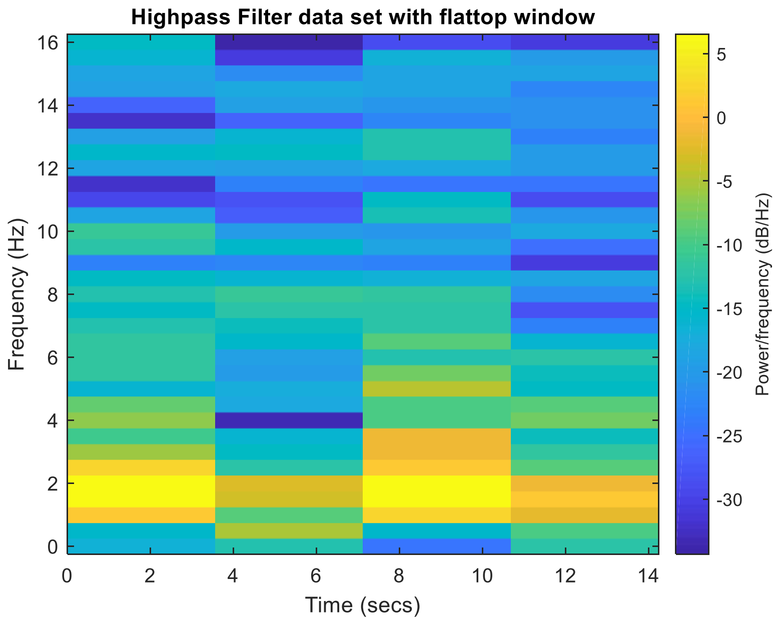






<u>Interpretation</u>

It can be observed from these spectrograms that irrespective of the window used, changing the window length decreases the number of segments/divisions for the same frequency and sample frequency and this decreases spectral resolution. However, it becomes blur at lower length or lower frequency points per segment.



After changing all the parameters for the flattop window, we can plot the optimal spectrogram as shown here with the following parameters :

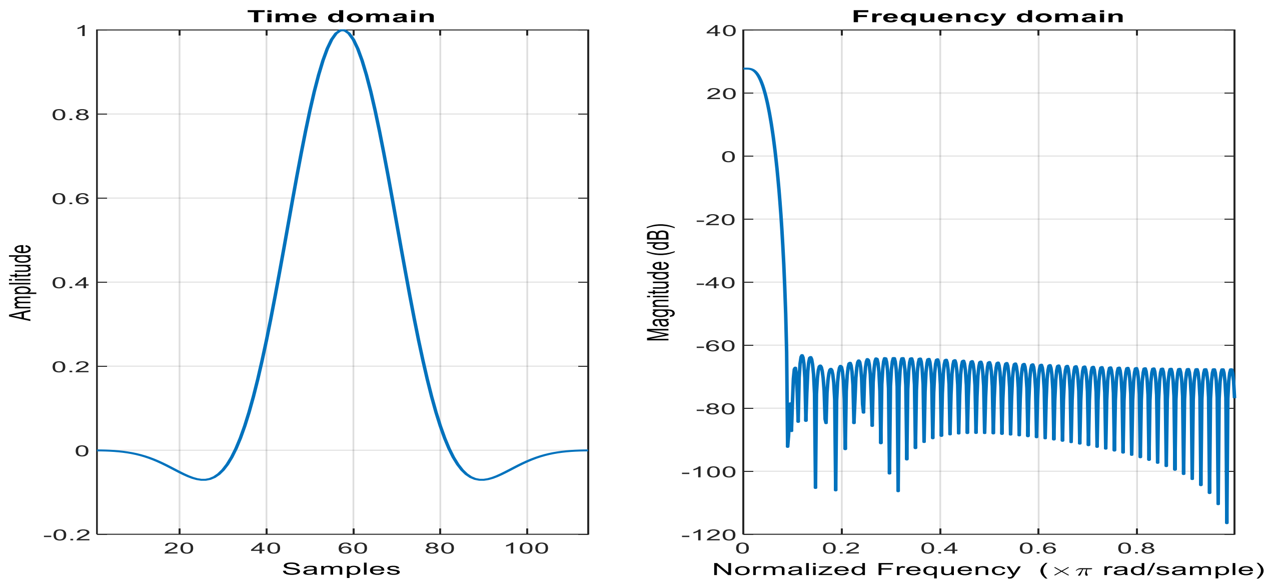
N = 114

F = 64

Fs = 32

% overlap = 0

Symmetric



Transient characteristics of the Flat top window for the above-mentioned parameters

1.3.3 Hamming Window

Hamming Window has one deciding factor

- Length (L) – frequency points per segment

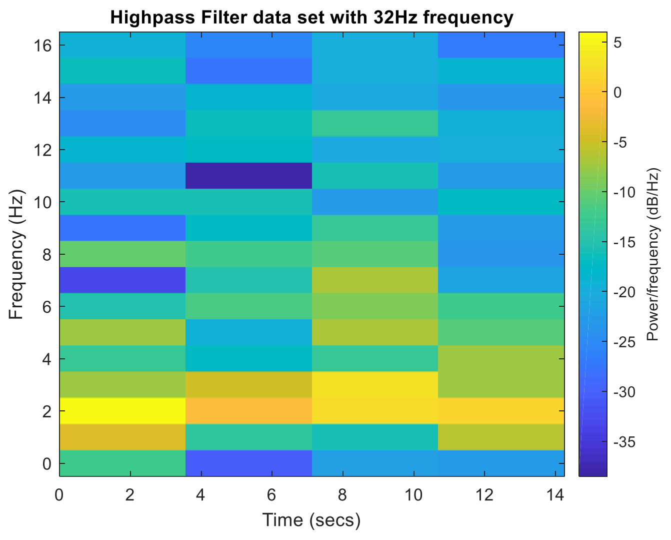
Function - spectrogram (input, hamming (L), 0, F, Fs)

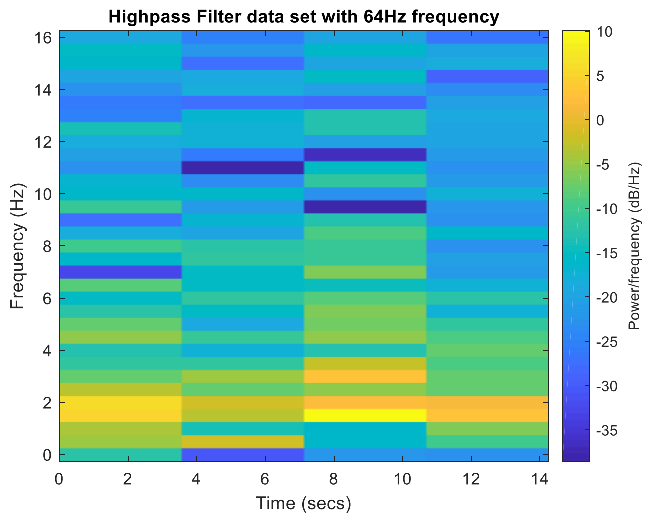
Plotting spectrograms with varying parameters using the flat top window:

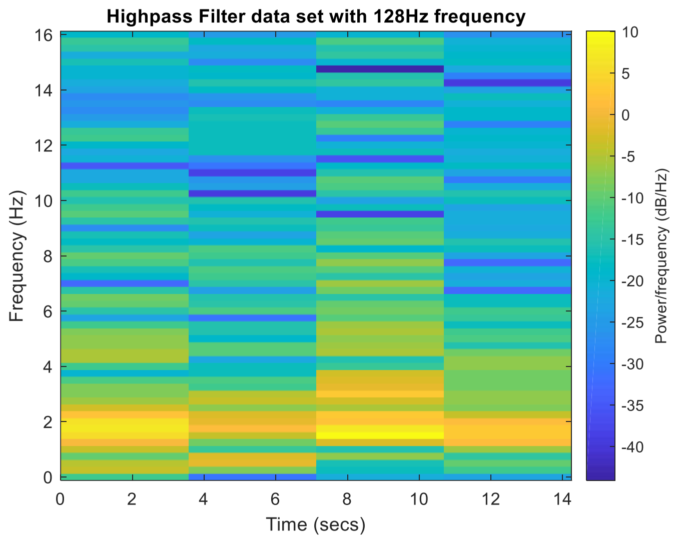
A. HAMMING WINDOW WITH CHANGING FREQUENCY (F)

```
spectrogram(filter_data,hamming(114),0,32,32,'yaxis')
title('Highpass Filter data set with 32Hz frequency')
figure(2),
spectrogram(filter_data,hamming(114),0,64,32,'yaxis')
title('Highpass Filter data set with 64Hz frequency')
figure(3),
spectrogram(filter_data,hamming(114),0,128,32,'yaxis')
title('Highpass Filter data set with 128Hz frequency')
```

Parameters with changing Frequency (F)	
Window	Hamming
Length (L)	114
% overlap	0
Sampling Frequency (Fs)	32







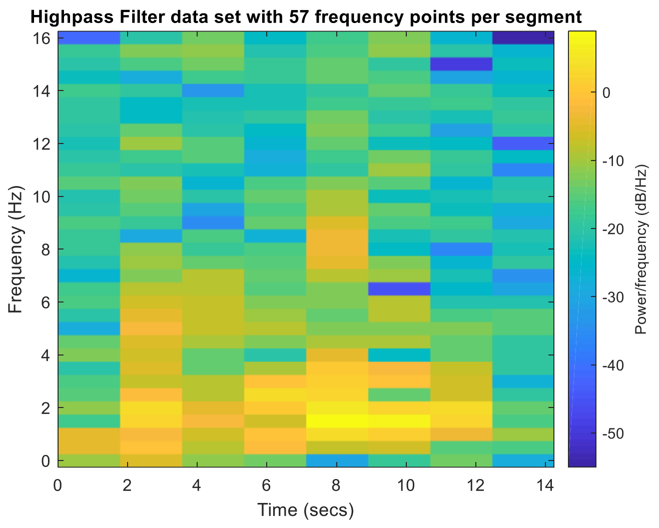
Interpretation

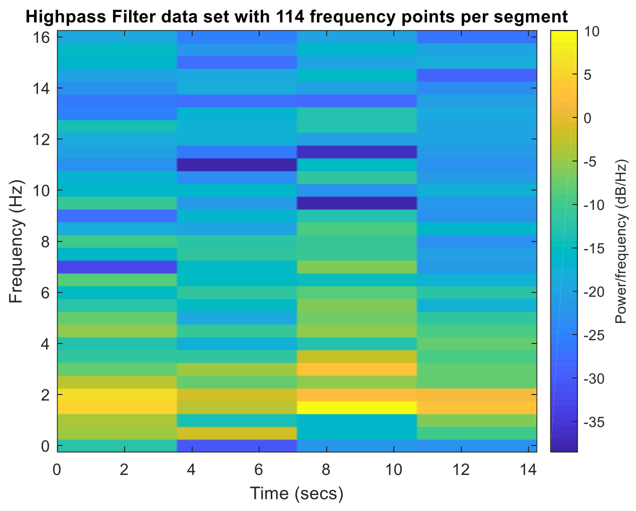
It can be observed from these spectrograms that irrespective of the window used, changing frequency increase the temporal resolution of the spectrogram i.e. the spectrogram has more information at a given frequency as the frequency increases and the information of any 2 adjoining frequency levels can be easily distinguished.

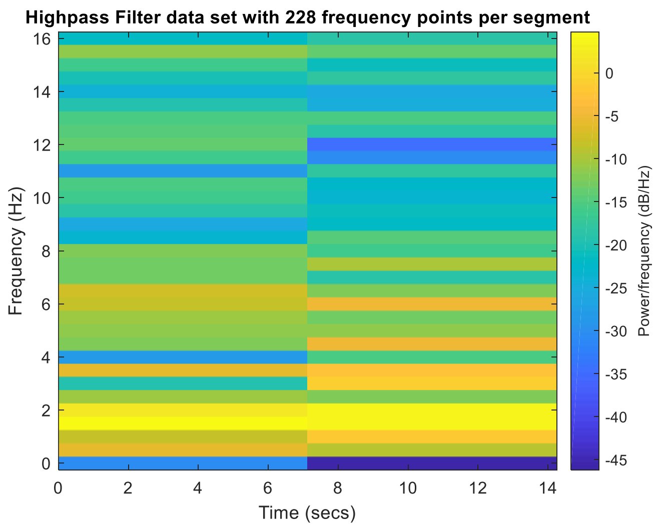
B. HAMMING WINDOW WITH CHANGING LENGTH (L)

```
spectrogram(filter_data,hamming(57),0,64,32,'yaxis')
title('Highpass Filter data set 57 frequency points per segment')
figure(2),
spectrogram(filter_data,hamming(114),0,64,32,'yaxis')
title('Highpass Filter data set 114 frequency points per segment')
figure(3),
spectrogram(filter_data,hamming(228),0,64,32,'yaxis')
title('Highpass Filter data set 228 frequency points per segment')
```

Parameters with changing Length (L)	
Window	Flat top
Frequency	64
% overlap	0
Sampling Frequency (Fs)	32

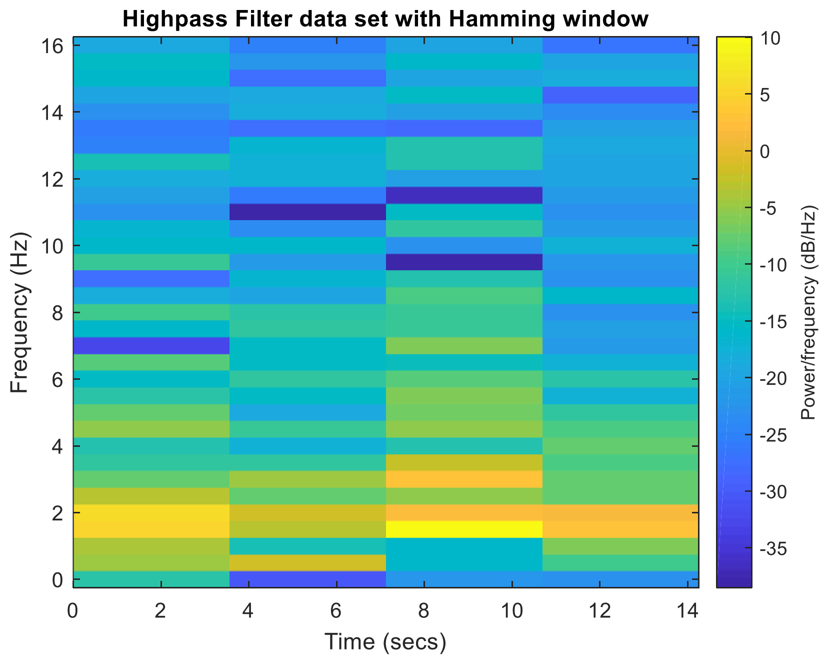






Interpretation

It can be observed from these spectrograms that irrespective of the window used, changing the window length decreases the number of segments/divisions for the same frequency and sample frequency and this decreases spectral resolution. However, it becomes blur at lower length or lower frequency points per segment.



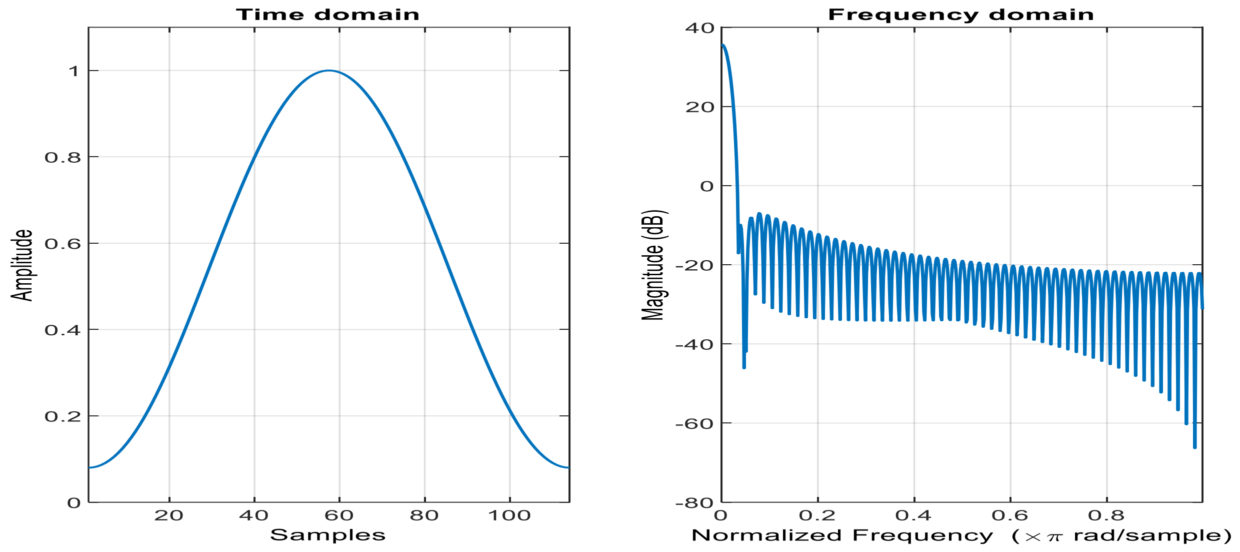
After changing all the parameters for the Hamming window, we can plot the optimal spectrogram as shown here with the following parameters :

N = 114

F = 64

Fs = 32

% overlap = 0



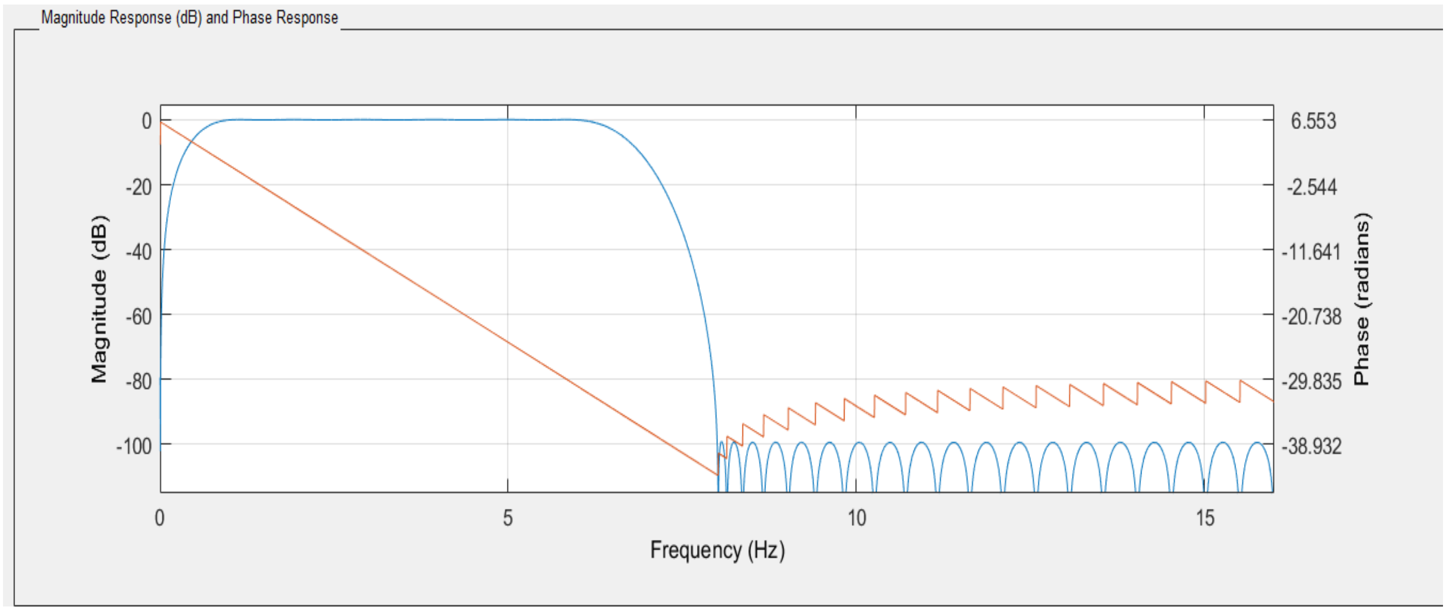
<u>Transient characteristics of the Hamming window for the above-mentioned parameters</u>

2. FILTER:

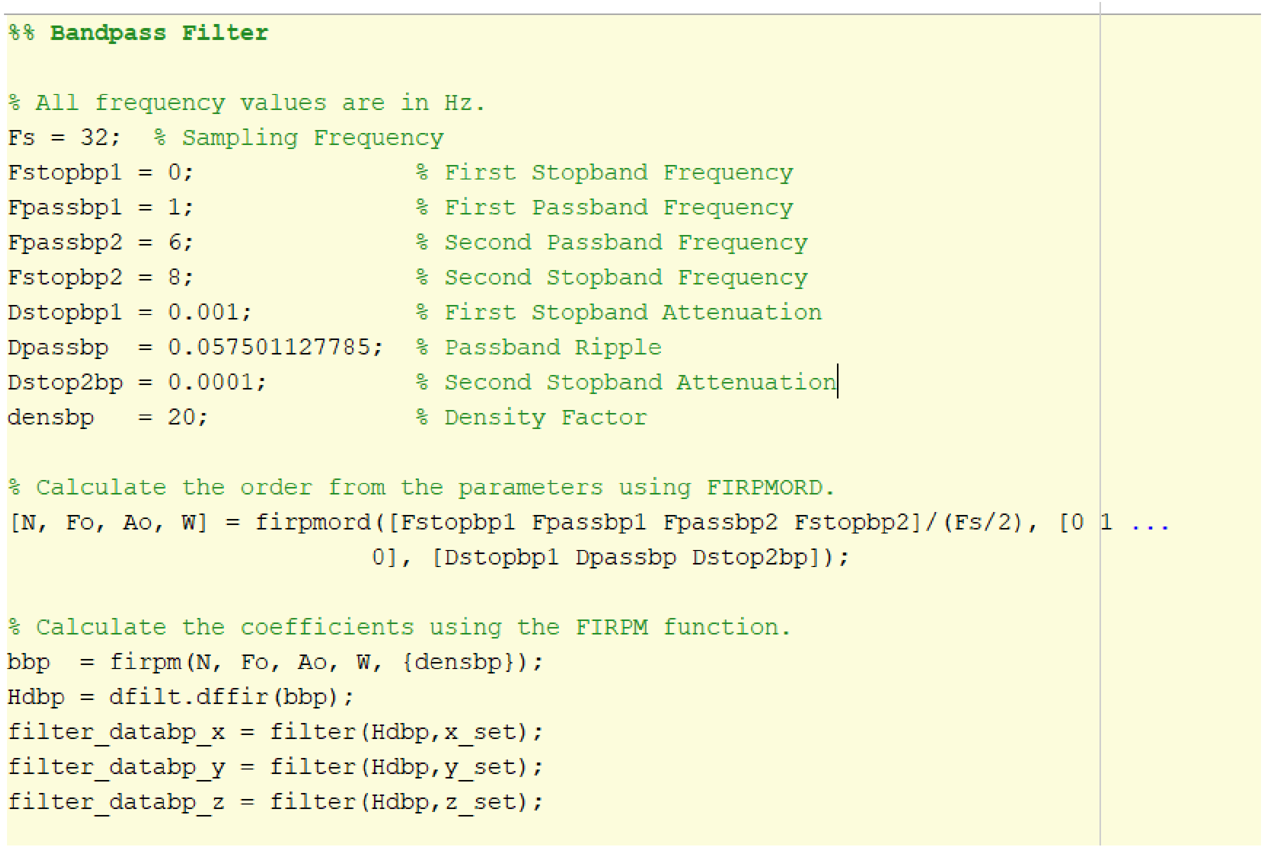
2.1 Bandpass Filter

A bandpass filter is a combination of lowpass and Highpass filter that passes frequencies within a certain range and rejects (attenuates) frequencies outside. It can be observed from the given data and the spectrograms plotted that any signal above ~ 6Hz has very low power or energy and hence can be discarded. It can also be noted that due to the presence of noise or unwanted signals below 2 Hz, any signal below 2 Hz becomes unwanted and hence can be eliminated. Also, it should be noted that the frequency of an average person walking is around 2 Hz and may vary for climbing and descending stairs, hence an average of 2 Hz is chosen as first pass frequency. Maximum plausible frequency for descending stairs (without skipping any stair) is around 6 Hz considering the age given in the data set. Therefore, the final design parameters with their values is shown in the following table,

Design Parameter	Value
Response Type	Bandpass
Design Method	FIR - Equiripple
Stable	Yes
Filter Order	63 (minimum)
Density Factor	20
Sampling Frequency (Fs)	32 Hz
First stop frequency (Fstop1)	0
First pass frequency (Fpass1)	1 Hz
Second pass frequency (Fpass2)	6 Hz
Second stop frequency (Fstop2)	8 Hz



Magnitude and Phase response of a bandpass filter with afore mentioned parameters



```
%% Bandpass Filter

% All frequency values are in Hz.
Fs = 32;  % Sampling Frequency
Fstopbp1 = 0;               % First Stopband Frequency
Fpassbp1 = 1;               % First Passband Frequency
Fpassbp2 = 6;               % Second Passband Frequency
Fstopbp2 = 8;               % Second Stopband Frequency
Dstopbp1 = 0.001;           % First Stopband Attenuation
Dpassbp  = 0.057501127785;  % Passband Ripple
Dstop2bp = 0.0001;          % Second Stopband Attenuation
densbp   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fstopbp1 Fpassbp1 Fpassbp2 Fstopbp2]/(Fs/2), [0 1 ...
                         0], [Dstopbp1 Dpassbp Dstop2bp]);

% Calculate the coefficients using the FIRPM function.
bbp  = firpm(N, Fo, Ao, W, {densbp});
Hdbp = dfilt.dffir(bbp);
filter_databp_x = filter(Hdbp,x_set);
filter_databp_y = filter(Hdbp,y_set);
filter_databp_z = filter(Hdbp,z_set);
```

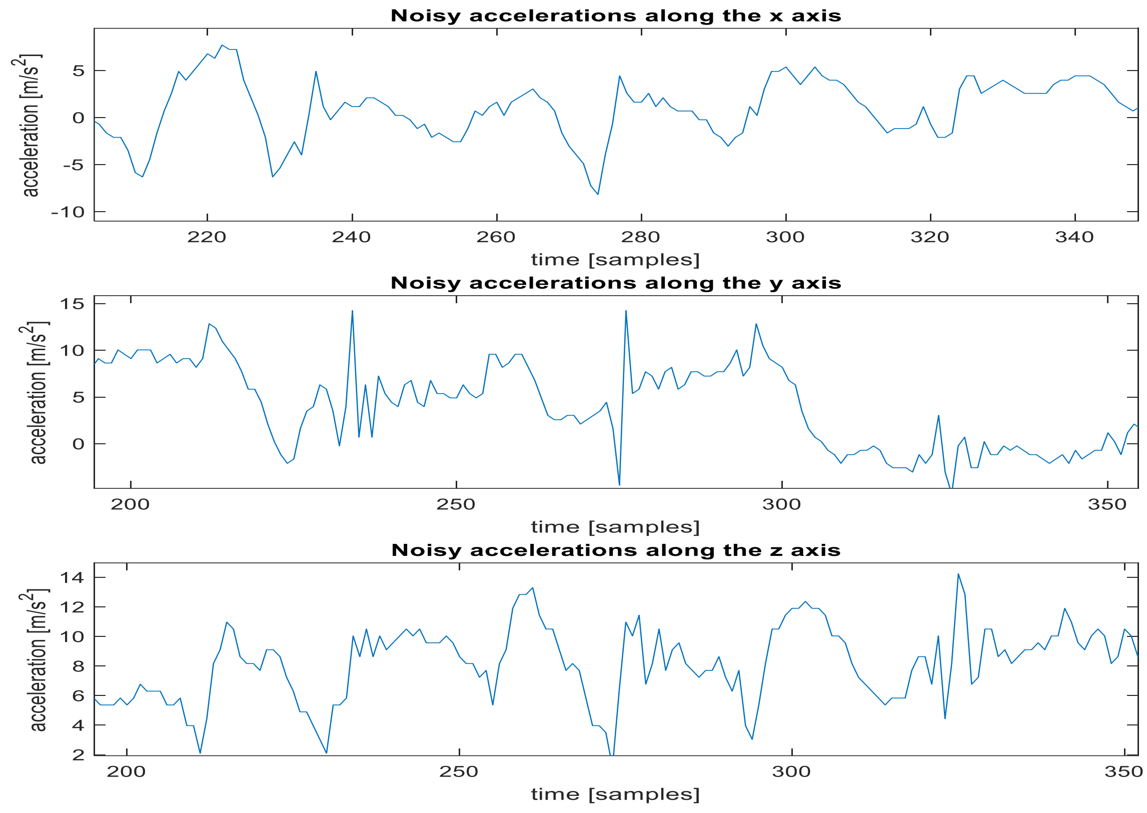
MATLAB Code for Bandpass filter

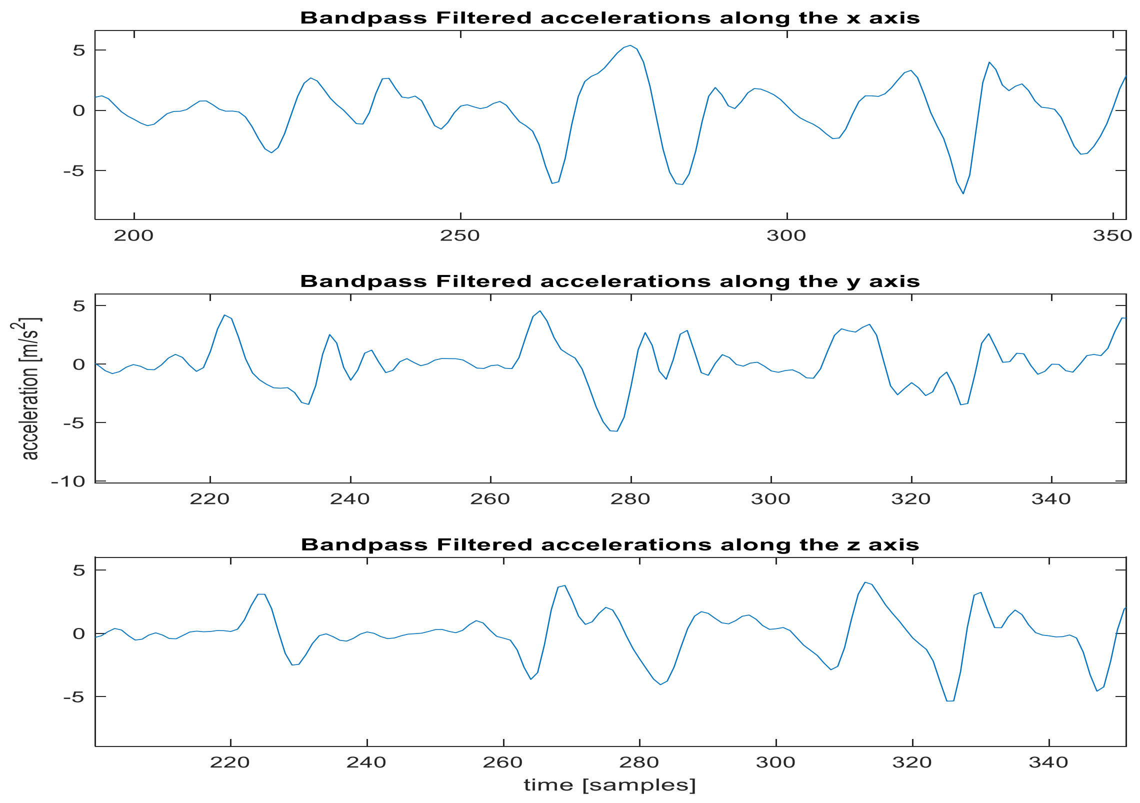
2.2 Comparision

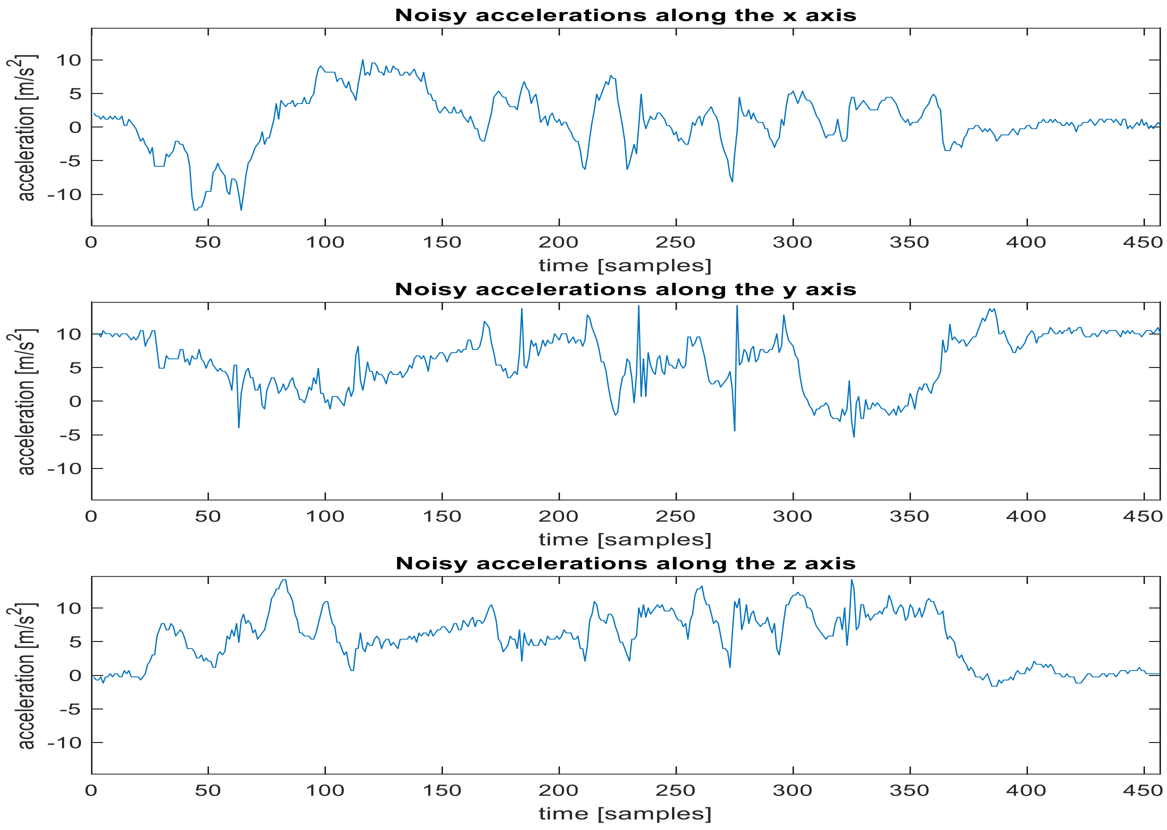
Three cases are presented below showing the comparision between noisy and filtered data for the similar activity and performed by the same human. Both zoomed in and zoomed out versions have been represented to make a better comparision and analysis.

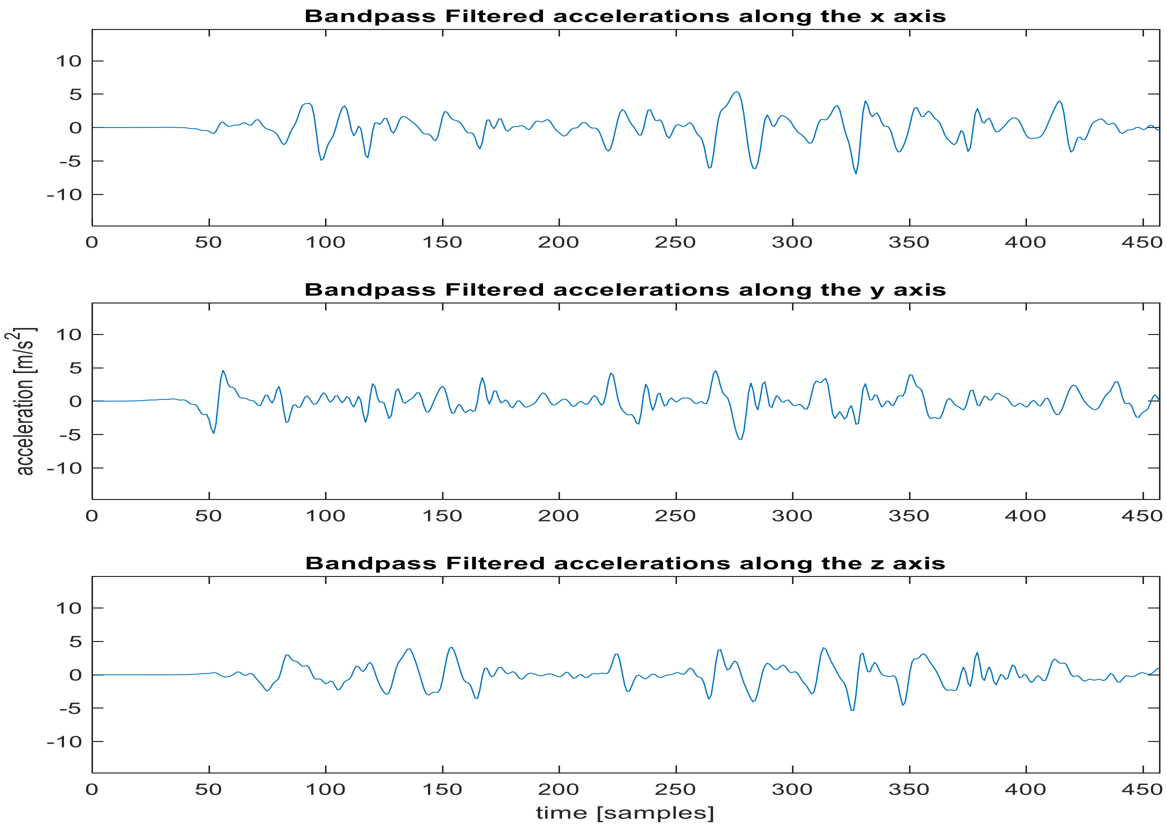
I. **CLIMB_STAIRS**

File name - **<u>Accelerometer-2011-05-30-22-00-32-climb_stairs-m2</u>**



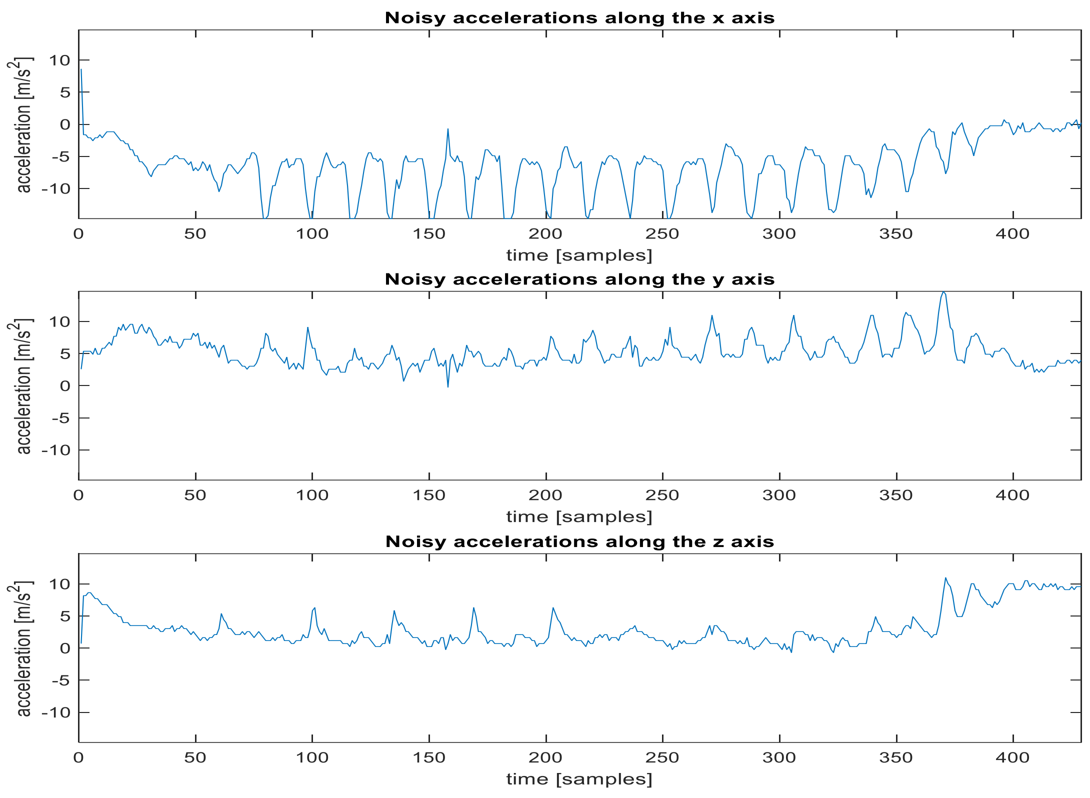


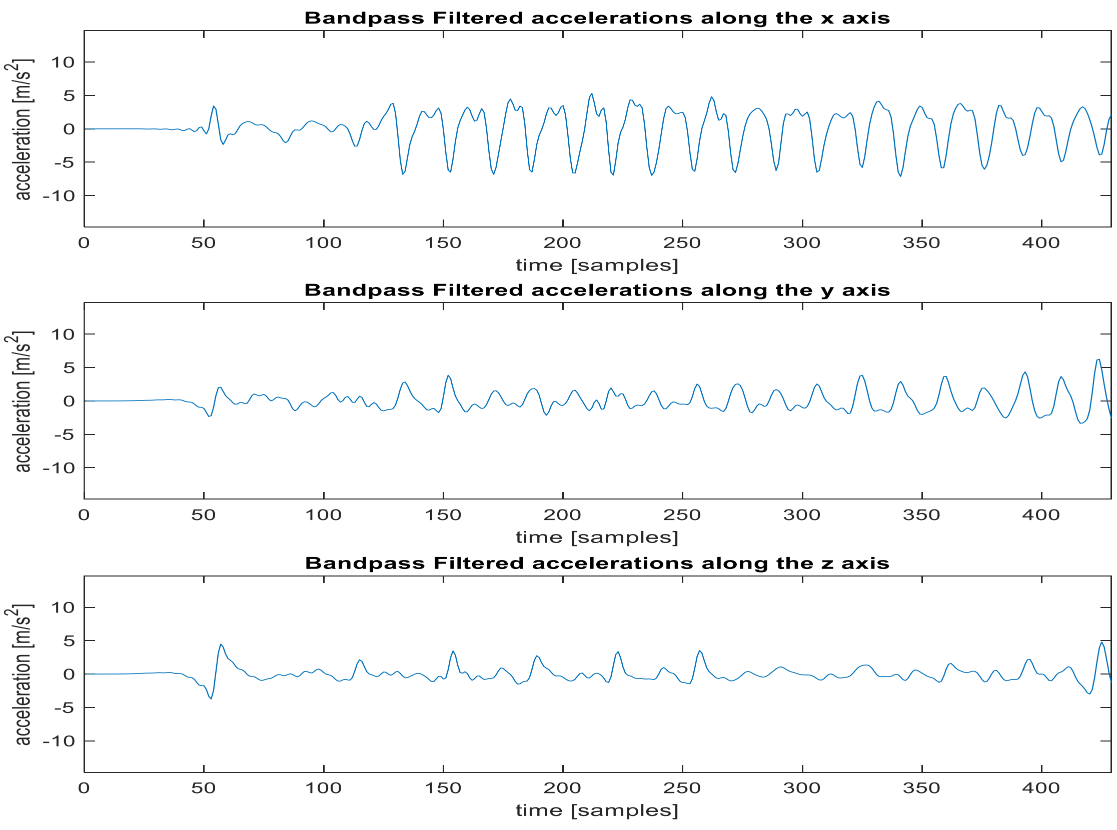

Noisy accelerations along the x axis
Noisy accelerations along the y axis
Noisy accelerations along the z axis
acceleration [m/s^2]
time [samples]

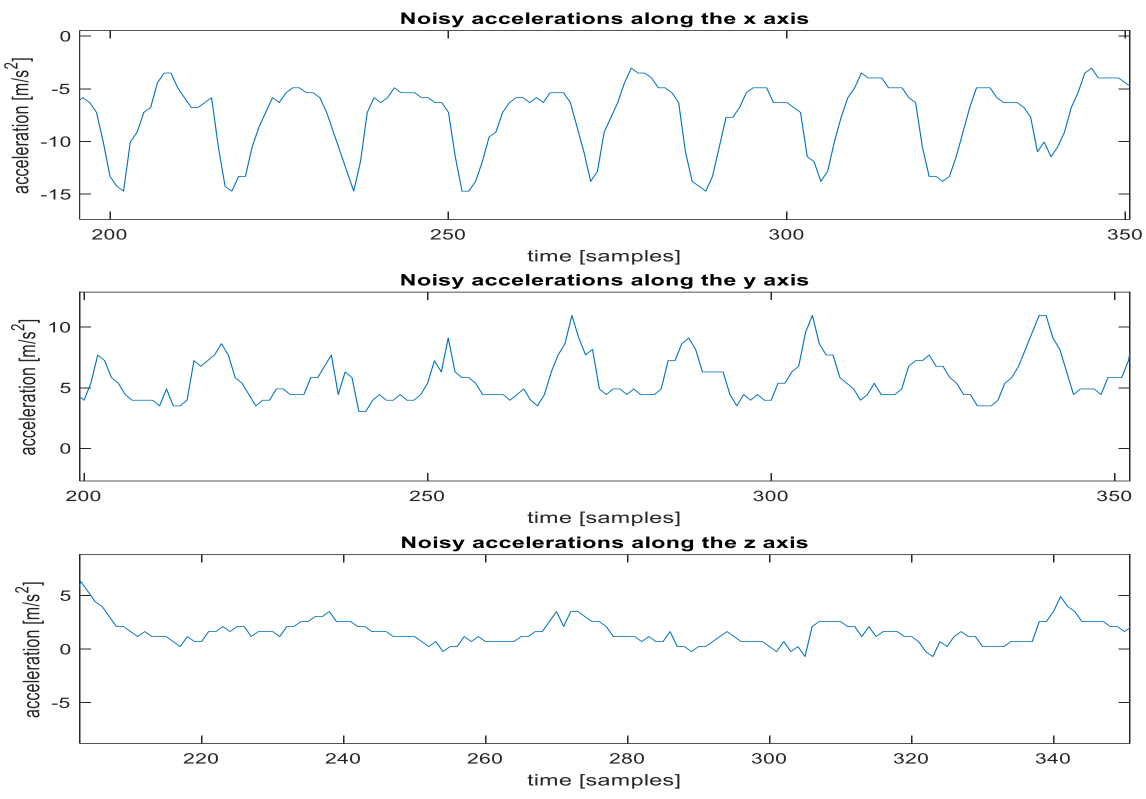

Bandpass Filtered accelerations along the x axis
Bandpass Filtered accelerations along the y axis
Bandpass Filtered accelerations along the z axis
acceleration [m/s^2]
time [samples]

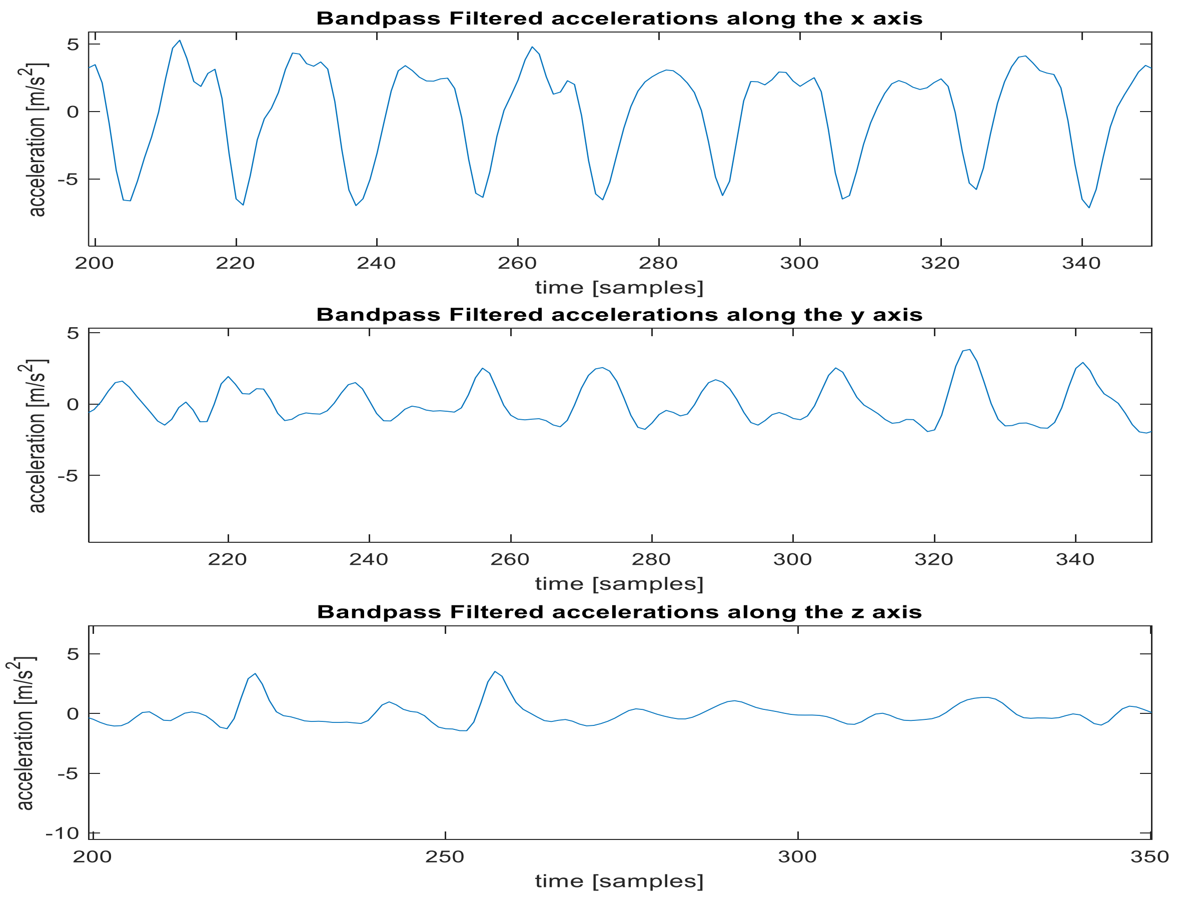
II. DESCEND_STAIRS

File Name **- <u>Accelerometer-2011-05-30-21-53-35-descend_stairs-m2</u>**



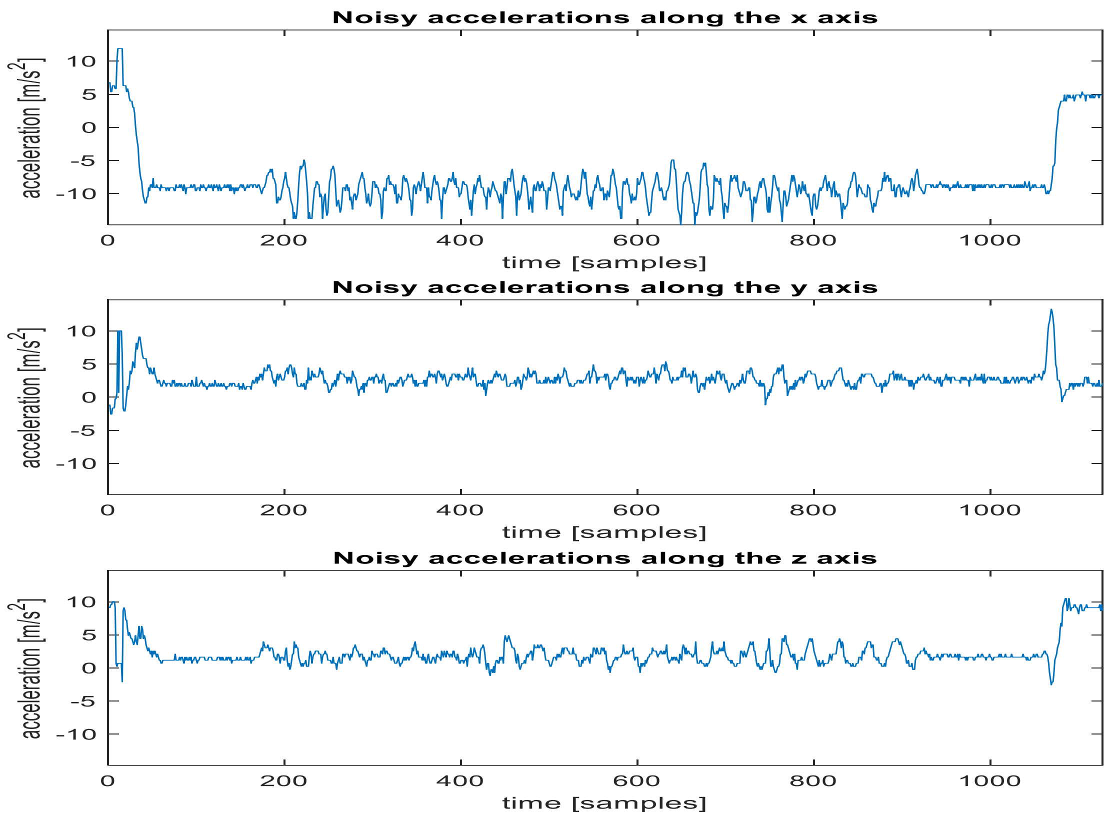


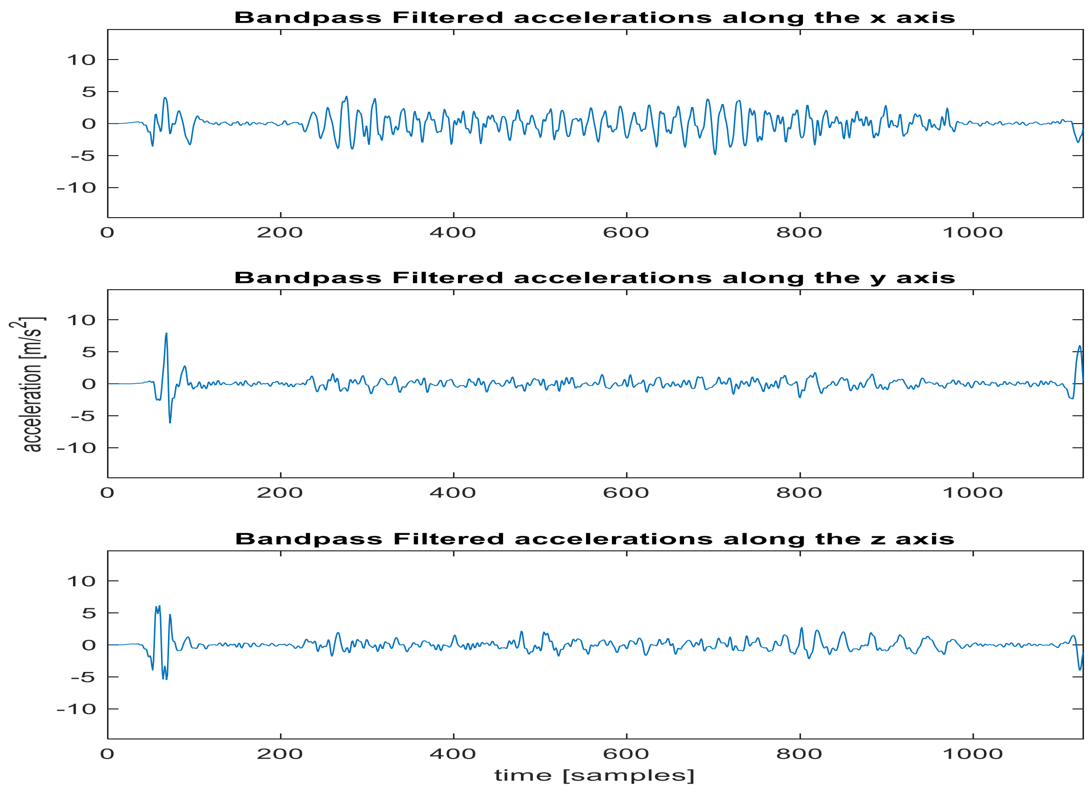

Noisy accelerations along the x axis
acceleration [m/s²]
time [samples]
Noisy accelerations along the y axis
acceleration [m/s²]
time [samples]
Noisy accelerations along the z axis
acceleration [m/s²]
time [samples]

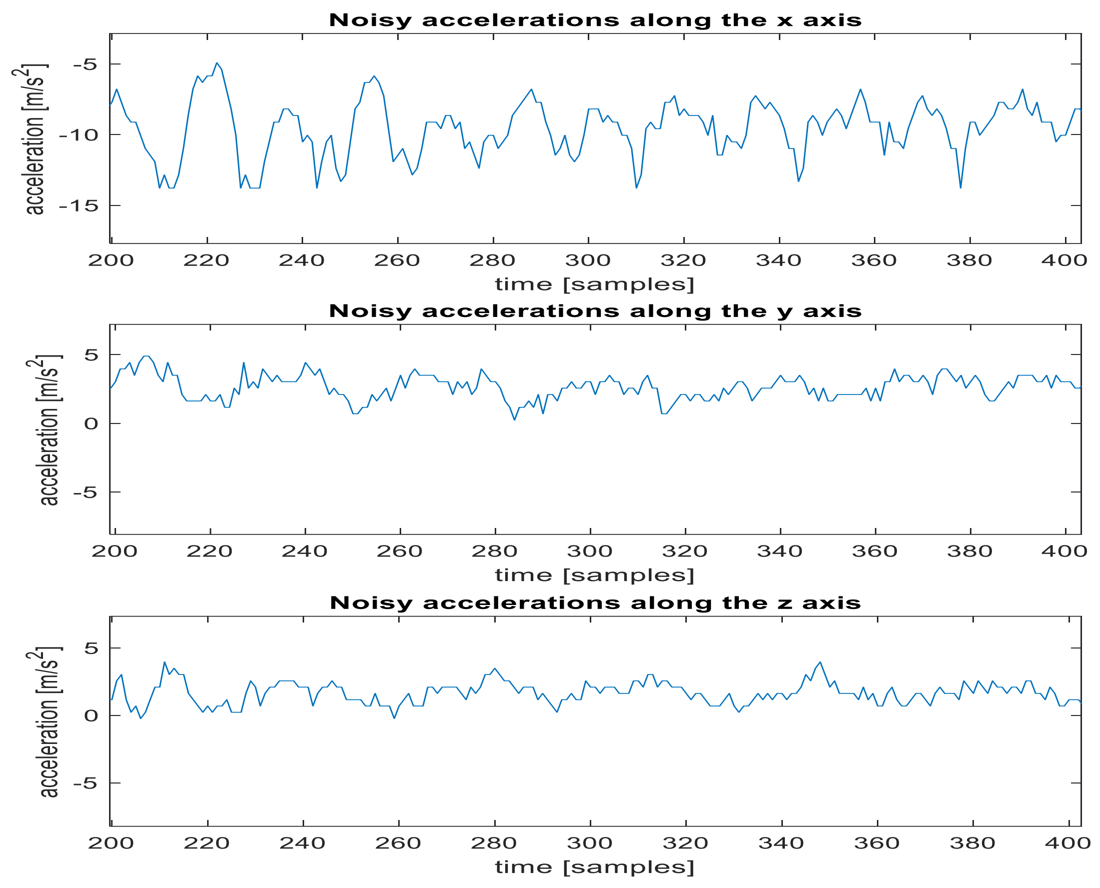

Bandpass Filtered accelerations along the x axis
acceleration [m/s²]
time [samples]
Bandpass Filtered accelerations along the y axis
acceleration [m/s²]
time [samples]
Bandpass Filtered accelerations along the z axis
acceleration [m/s²]
time [samples]

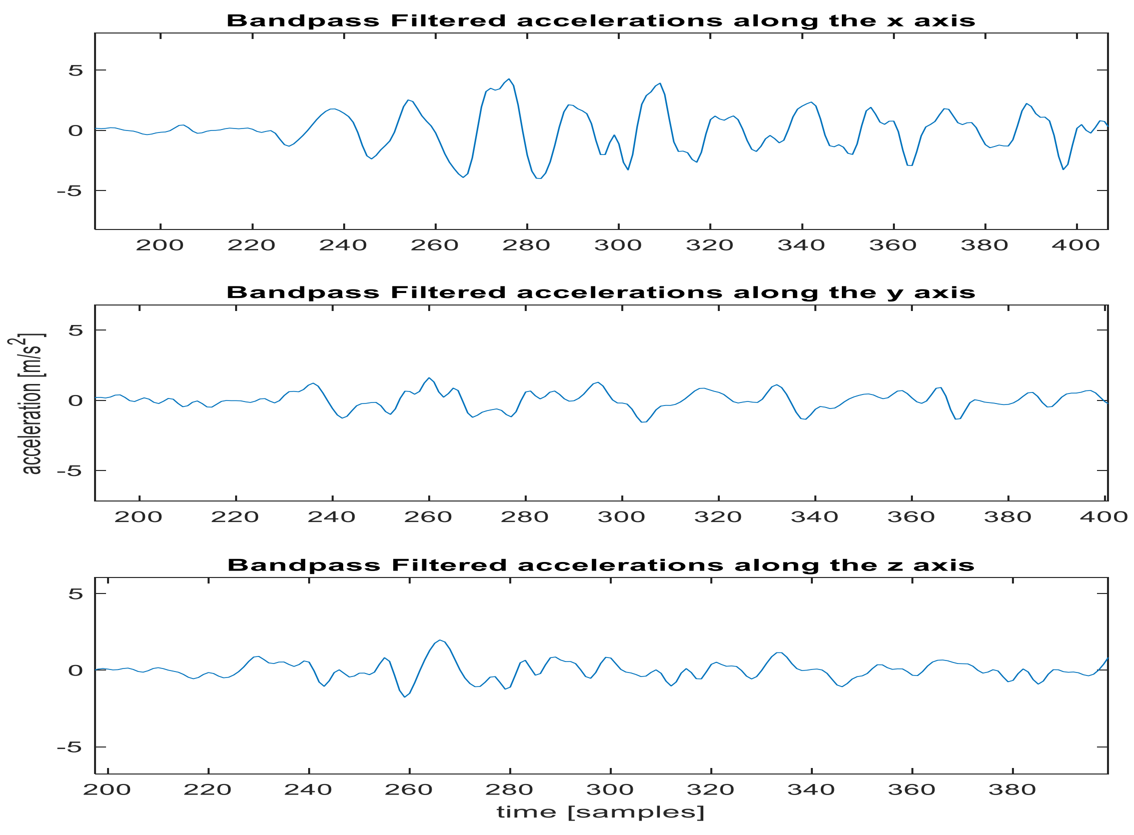
III. **WALK**

File Name **- <u>Accelerometer-2012-06-11-11-38-12-walk-m1</u>**






Noisy accelerations along the x axis
acceleration [m/s^2]
-5
-10
-15
200 220 240 260 280 300 320 340 360 380 400
time [samples]
Noisy accelerations along the y axis
acceleration [m/s^2]
5
0
-5
200 220 240 260 280 300 320 340 360 380 400
time [samples]
Noisy accelerations along the z axis
acceleration [m/s^2]
5
0
-5
200 220 240 260 280 300 320 340 360 380 400
time [samples]


Bandpass Filtered accelerations along the x axis
5
0
-5
200 220 240 260 280 300 320 340 360 380 400
Bandpass Filtered accelerations along the y axis
acceleration [m/s^2]
5
0
-5
200 220 240 260 280 300 320 340 360 380 400
Bandpass Filtered accelerations along the z axis
5
0
-5
200 220 240 260 280 300 320 340 360 380
time [samples]

It can be observed that after applying the bandpass filter that the:

a) The filtered signal has become smoother and less noisy. It has no kinks or sudden rise and drop.
b) The acceleration always starts from 0 acceleration indicating absence of any noise or DC gain that was present in the noisy signal due to multiple and unavoided reasons.
c) The amplitude of the signal has reduced to a nominal and plausible range unlike the noisy signal.
d) The filtered signal ends at 0 indicating that the person comes to rest unlike the noisy signal where there always exists some noise and acceleration can never be 0.
e) Mean of the acceleration in any axis for the filtered signal is 0 however, nothing can be said about the mean for the noisy signal.

3. CLASSIFIER

3.1 Structure

The type of classifier used is Bagged Tree, with the learning method as ensemble.

Bagging is a bootstrap ensemble method that creates individuals for its ensemble by training each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing, with replacement, *N* examples - where *N* is the size of the original training set; many of the original examples may be repeated in the resulting training set while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set.

An ensemble model is a technique that combines predictions from multiple machine learning algorithms together to make accurate predictions than any individual model.

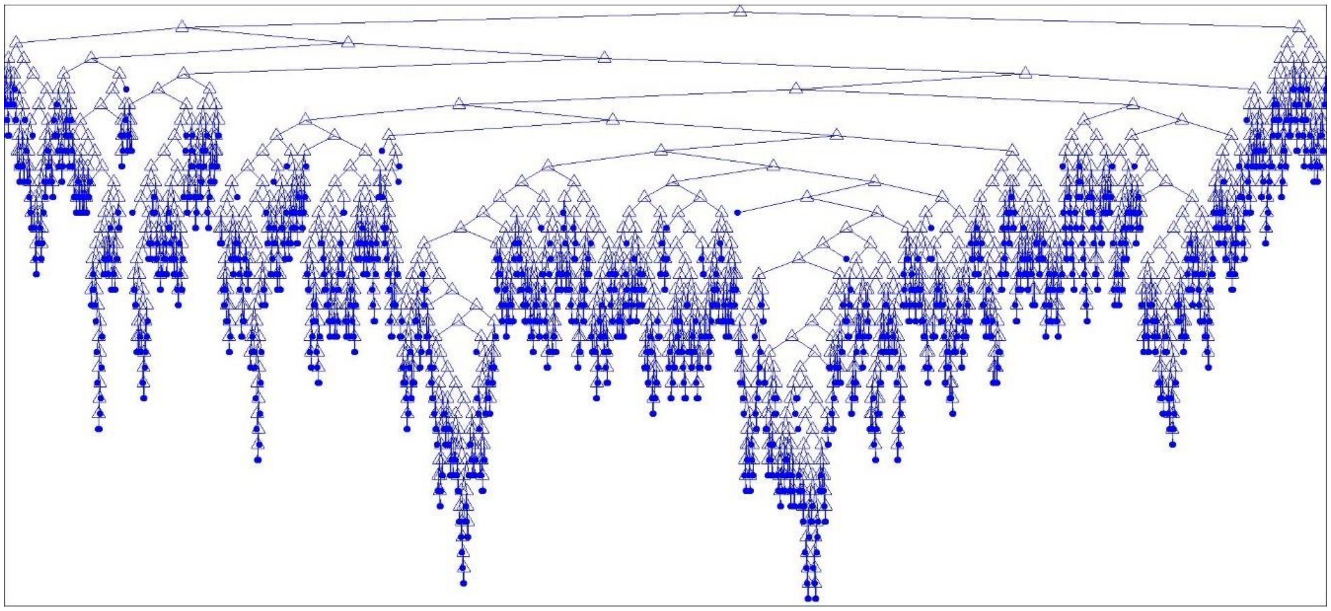
Conventional decision tree algorithm is a high variance algorithm and is data sensitive, i.e. the resulting decision tree could change with minimum changes to training data. Thus, the concept of bagging applied to trees become useful which allows to reduce the variance on a high variance algorithm. Bagging applied to decision trees removes the concern of overfitting data.

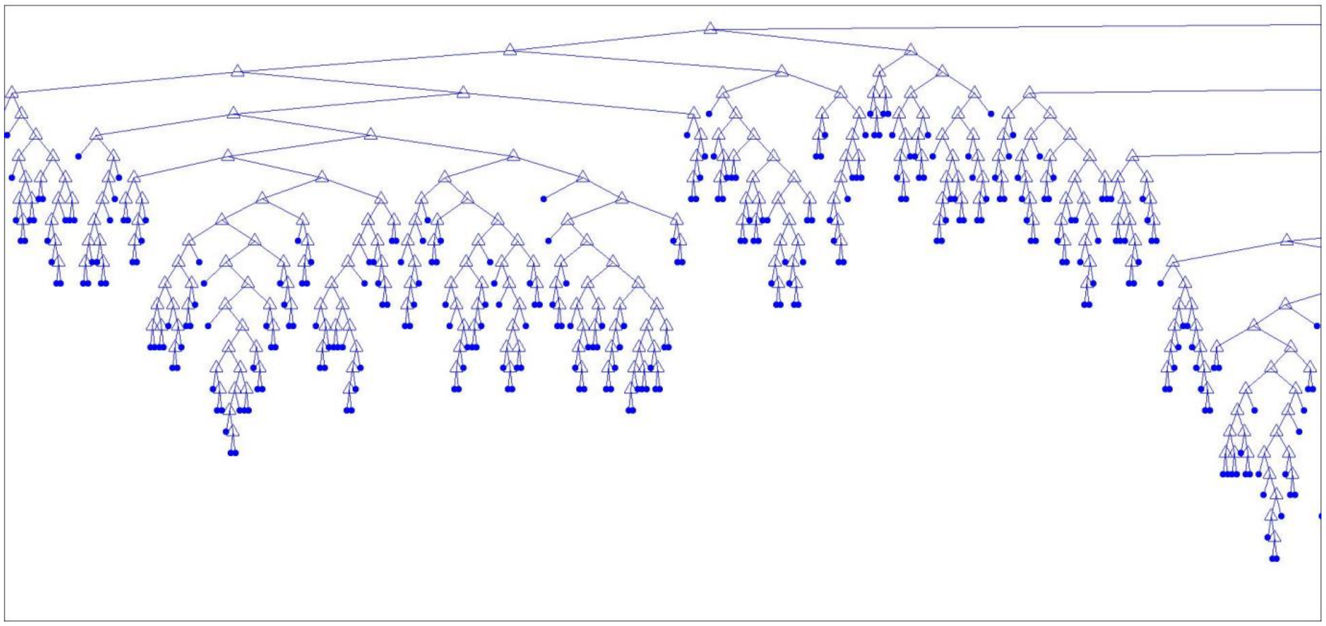
In a bagged tree classifier, the entire population P of size N is divided into B samples of size n.

1. Create Multiple Data Sets:
   - Bagging generates B new training sets of size n by sampling from P uniformly and with replacement.
   - A construction tree is created.

2. Build Multiple Classifiers:
   - Classifiers are built on each data set.
   - The B models are fitted by assigning a class to each terminal node, and the class attached to each case is stored, couple with predictor values for each observation.

3. Combine Classifiers:
   - The predictions of all the classifiers are combined using voting for classification that leads to a robust model.


Architecture of the Classifier


Structure of main Tree

3.2 Procedure

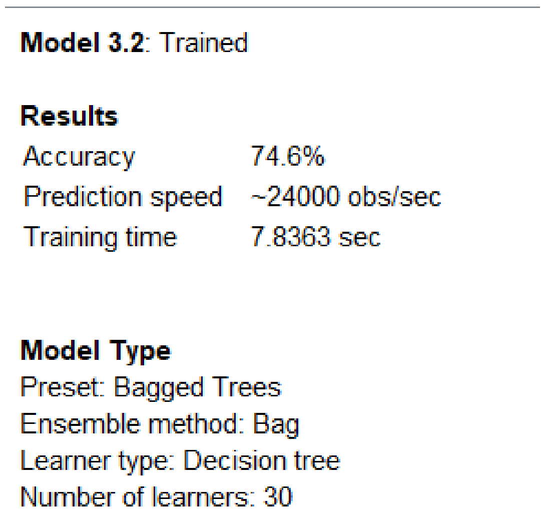
Procedure for training the classifier was as follows:

1) The noisy data provided was filtered using a High-pass Filter to remove the DC component.
2) Filtered data was further smoothened using median filtering.
3) From the processed data, approximately 7500 observations were chosen randomly by exporting the processed data to excel and choosing through random selection.
4) The selected data was trained on a Classifier in MATLAB, using Structure type as Bagged Tree and learner method as Ensemble.
5) The Validation process chosen was 5-fold Cross-Validation.

6) The model was trained across structures which included Decision Trees, Support Vector machines, Ensemble classifiers and Nearest Neighbor classifier.
7) The model with highest accuracy was chosen- Bagged Tree using Ensemble.
8) The number of learners chosen for the model in Step 7 was 30.

3.3 Evaluation

3.3.1 Trained Data

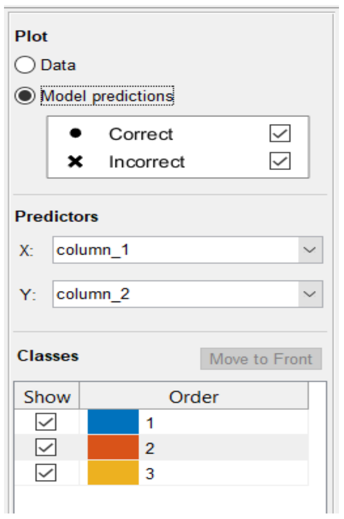


Model 3.2: Trained

Results

Accuracy	74.6%
Prediction speed	~24000 obs/sec
Training time	7.8363 sec

Model Type
Preset: Bagged Trees
Ensemble method: Bag
Learner type: Decision tree
Number of learners: 30



Plot
Data
[✓ Model predictions]

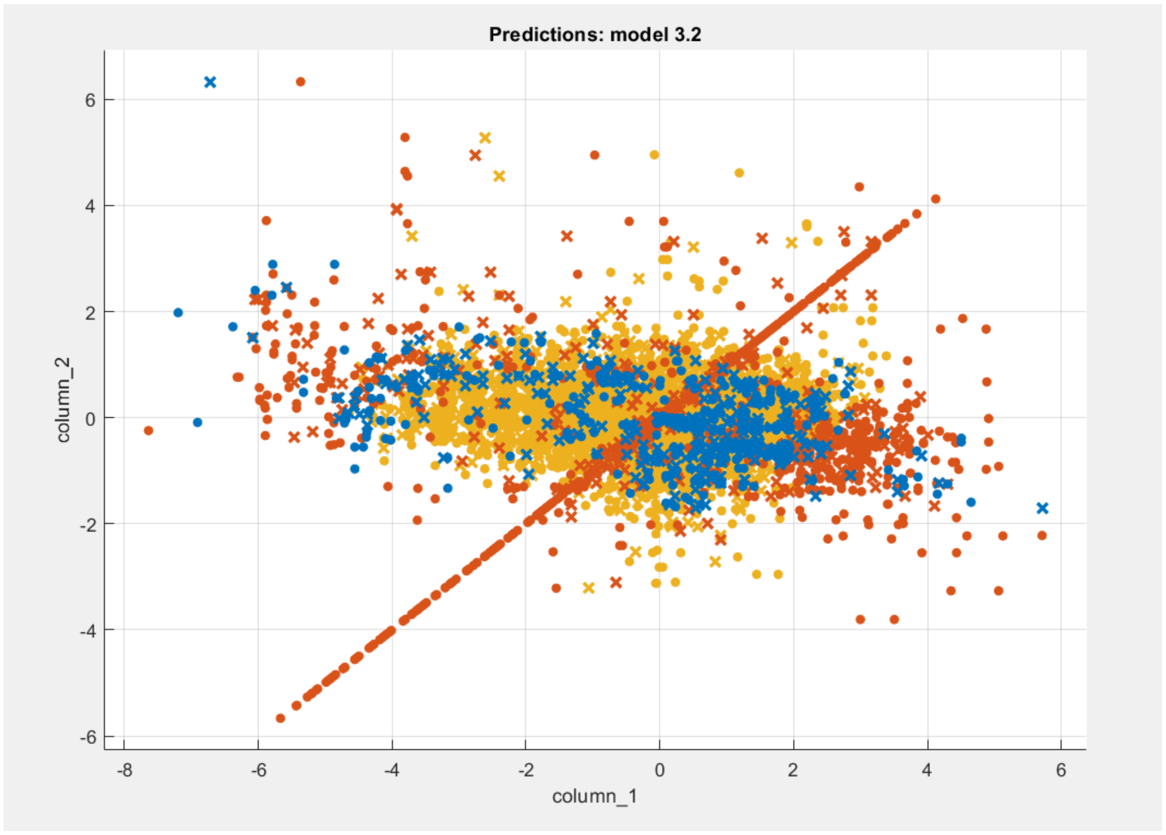
● Correct ✓
✖ Incorrect ✓

Predictors
X: column_1
Y: column_2

Classes

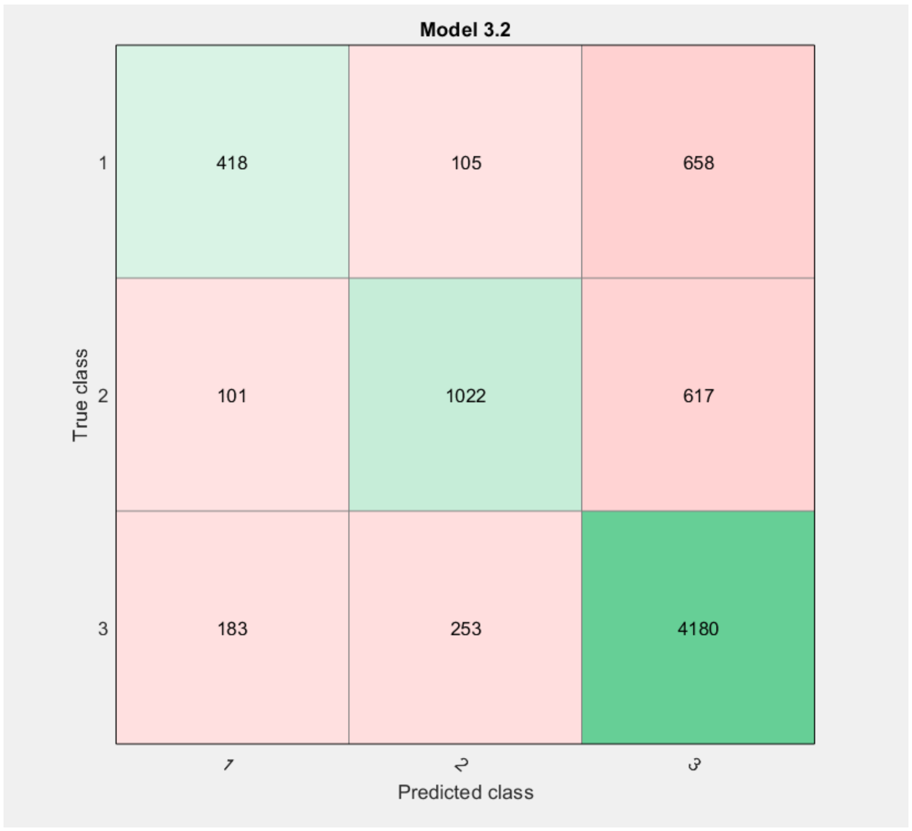
Show	Order
✓	🟦 1
✓	🟧 2
✓	🟨 3

Classifier accuracy on trained data



Performance of the trained classifier on the trained data

This classification learning structure allows us to make predictions for models that include principle component analysis (PCA)

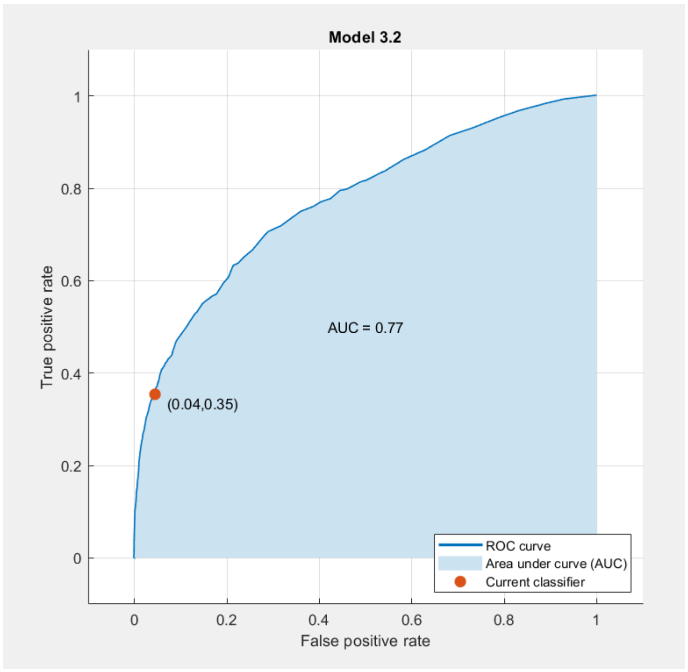


True class \ Predicted class	1	2	3
1	418	105	658
2	101	1022	617
3	183	253	4180

Confusion Matrix for the trained data

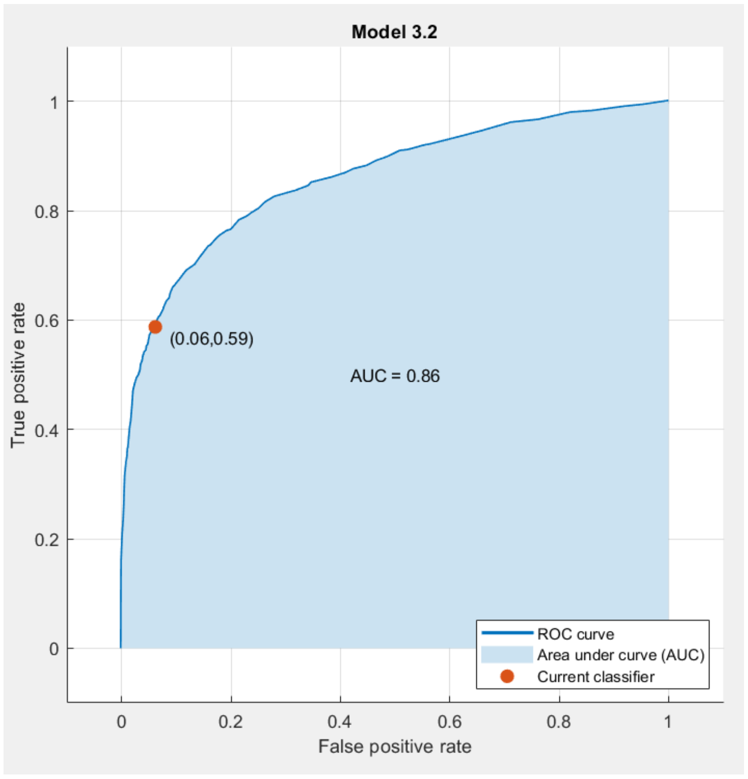
The Receiver Operating Curve (ROC) for different classes is shown below. ROC is plotted between the True positive rate and the False negative rate. Area under the curve marks the accuracy performance of the classifier.

i. Climbing Stairs



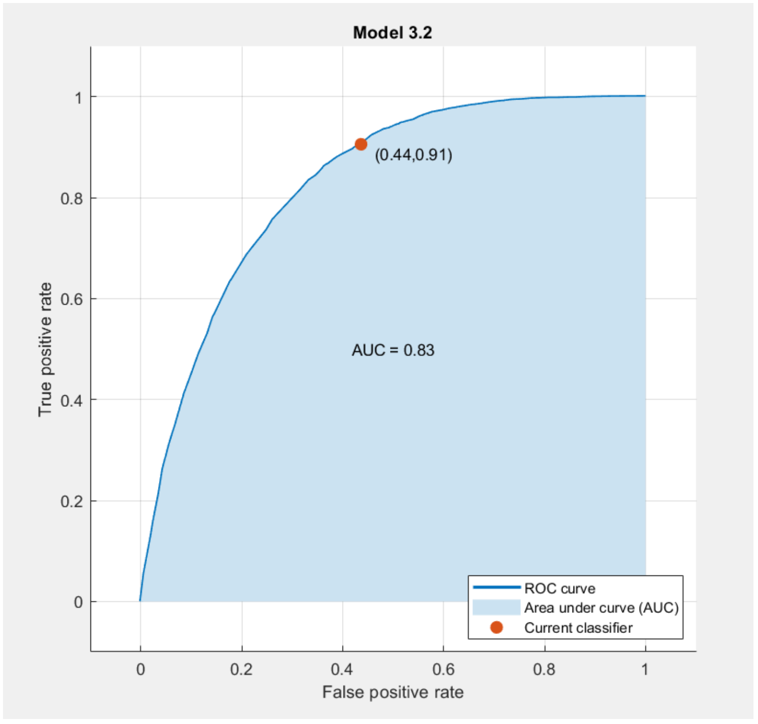
The area under the curve (AUC) is 77% indicating the accuracy of the classifier for this activity.

ii. Descending Stairs



The area under the curve (AUC) is 86% indicating the accuracy of the classifier for this activity.

iii. Walking



The area under the curve (AUC) is 83% indicating the accuracy of the classifier for this activity

3.3.1 Raw Data

Raw data of around 2000 values was compromised and the confusion matrix can be represented as shown below-

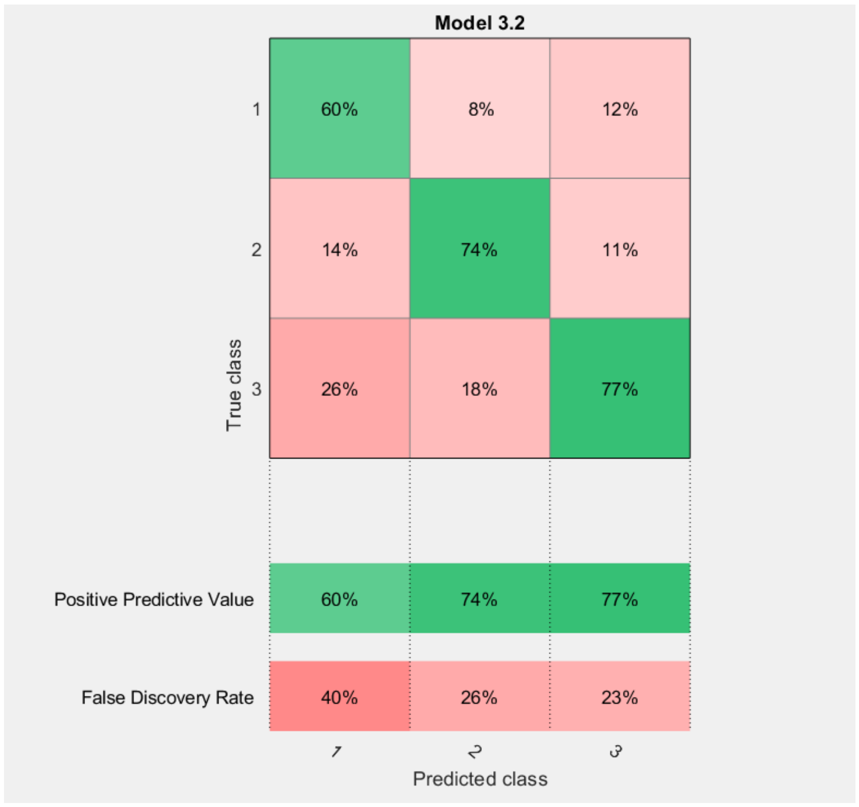
		1	**2**	**3**
True Class	1	30	79	205
	2	81	110	387
	3	63	93	844

Prediction accuracy on the raw data is about 52%

4. ANALYSIS

4.1 False Positive

False positive is a type I error where a false detection is made i.e. the event did not occur or the condition was not satisfied still the result was true or positive.

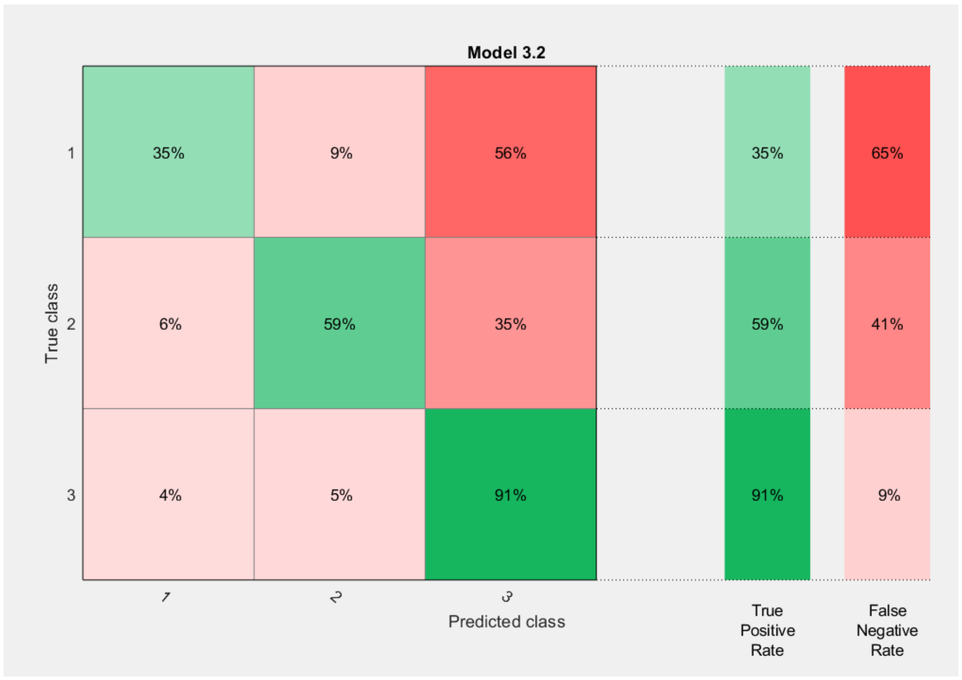


<u>False Positive Value and False discovery rate</u>

It can be inferred that Positive detection for Class 3 (Walking) is highest followed by Class (2) followed by Class (1). It means that the possibility of detecting a walking activity correctly is highest while detecting a climbing stairs activity is the lowest.
Also, it can be said that the chances of false detection or false discovery rate is highest while climbing stairs and least while walking implying that there are 40% chances that the human was not climbing stairs while the model said so while only 23% chances that the human was no walking while the model said so.

4.2 False Negative

False positive is a type II error where a miss detection is made i.e. the event did occur or the condition was satisfied but the result was still false or negative i.e. the system missed that event.



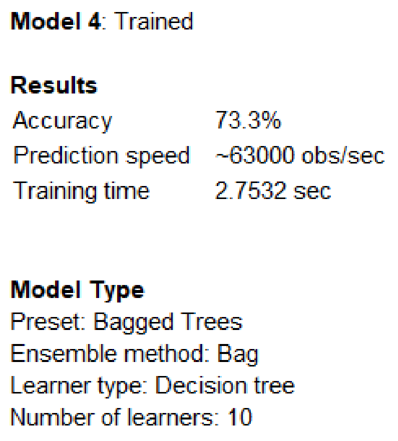
True Positive Rate and False Negative rate

It can be inferred that the True positive rate of class 3 (walking) is the highest while class 1 (climbing stairs) is the lowest implying that the possibility of correctly measuring an event is highest in walking in least in climbing stairs.
Also, it can be said that there is a 91% chance that the system correctly reads an event and 9% chance it missed the true event in case of walking whereas there is only 35% chance that the system will accurately read the event of climbing stairs implying that 65% of the total times, the system will miss the climbing event even if the event actually happened.

4.3 Tuning

Ensemble is a method which consists of many weak learners which are trained to generate a single strong learner. Thus, one of the ways to tune the classifier used (Bagged Ensemble Classifier) is to change the number of learners by either pruning the weak learners or adding more learner to improve the final learner.

Effect of Decreasing Learner (from 30 to 10)
After decreasing the number of learners, it can be observed that the False Positive and False negative attains new values and the overall efficiency or accuracy of the model decreases from 74.6% to 73.3%.

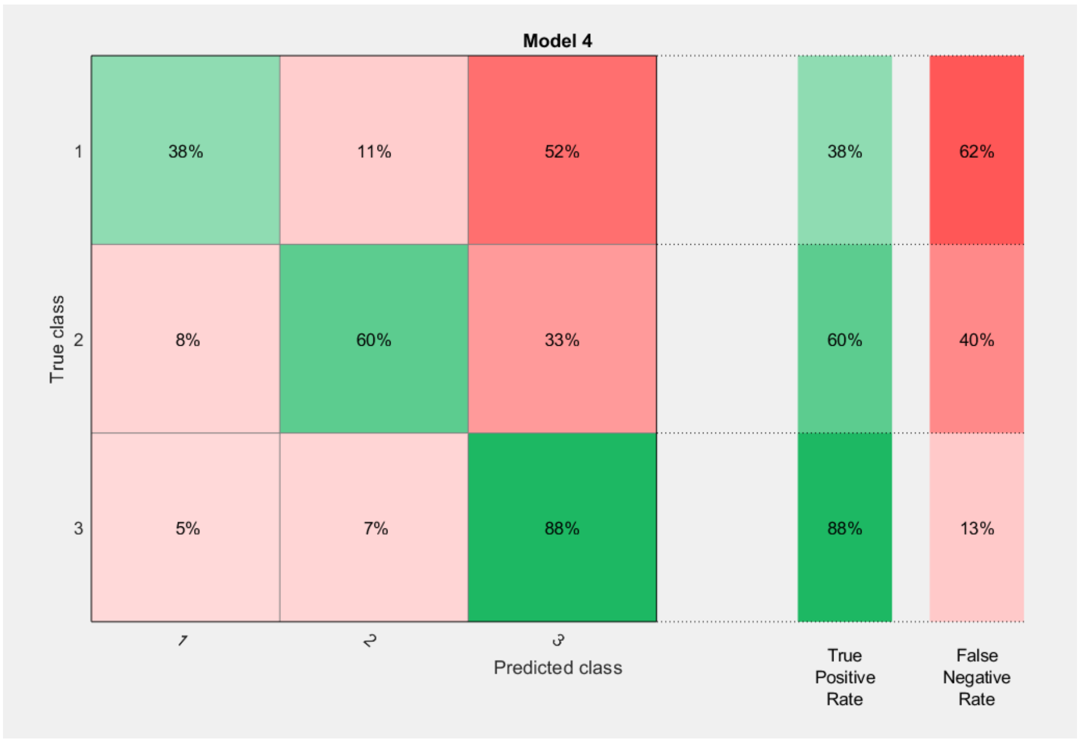


Model 4: Trained

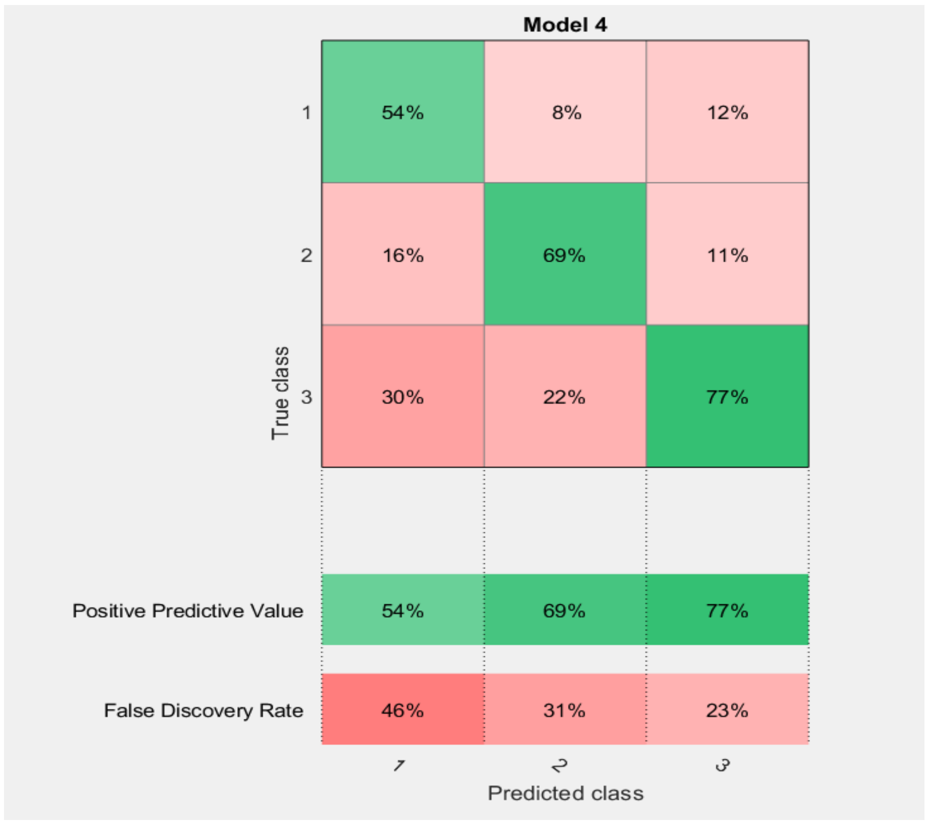
Results

Accuracy	73.3%
Prediction speed	~63000 obs/sec
Training time	2.7532 sec

Model Type
Preset: Bagged Trees
Ensemble method: Bag
Learner type: Decision tree
Number of learners: 10



True Positive Rate and False Negative rate



True positive Value and False Discovery Rate

Interpretation

It can be observed that as the number of learners reduce, the false negative rate decreases for climbing stairs and descending stairs by 3% and 1% respectively however, for walking it increases by 4%.

Also, as the number of learners reduce, the false positive rate increases for climbing stairs and descending stairs by 6% and 5% respectively however, it remains same for walking.

Effect of Increasing Learner (from 30 to 50)

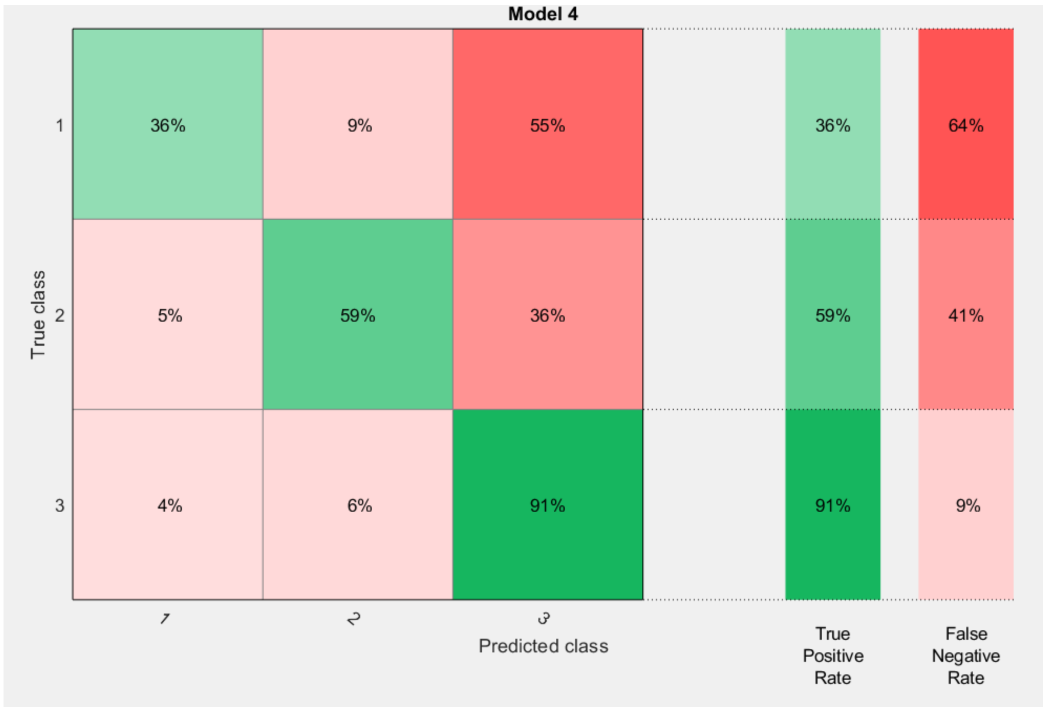
Model 4: Trained

Results

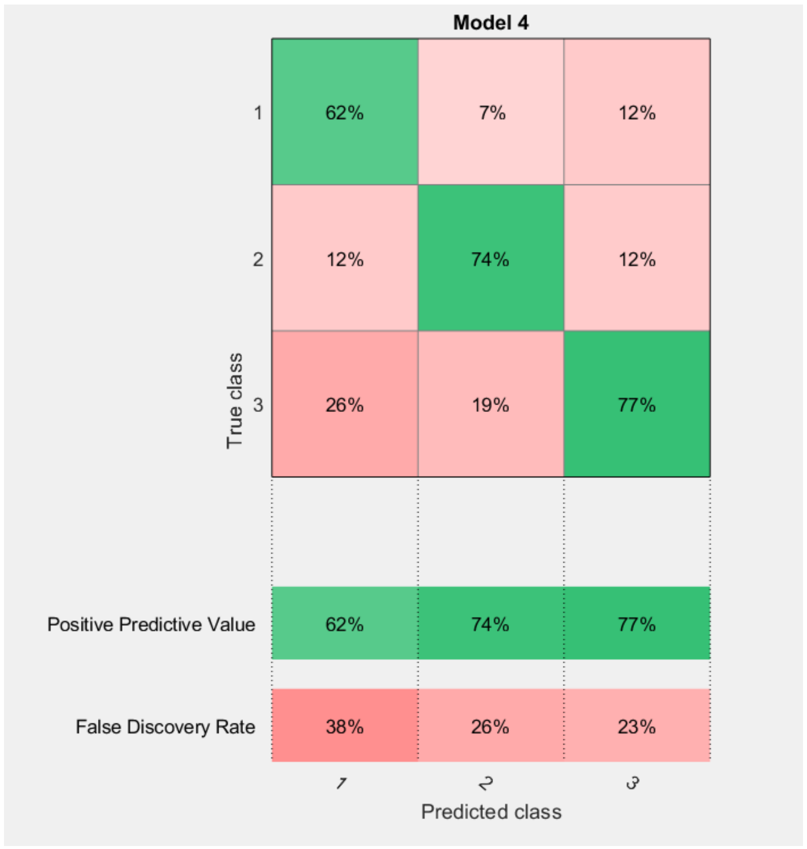
Accuracy	74.8%
Prediction speed	~15000 obs/sec
Training time	14.746 sec

Model Type

Preset: Bagged Trees
Ensemble method: Bag
Learner type: Decision tree
Number of learners: 50



True Positive Rate and False Negative rate



Interpretation

It can be observed that as the number of learners increase, the false negative rate decreases for climbing stairs by 1% while remains same for descending stairs and walking.

Also, as the number of learners increase, the false positive rate decreases by 2% for climbing stairs and remains same for walking and descending stairs.

True positive Value and False Discovery Rate

Thus, it can be concluded that by increasing /decreasing the number of learners, there has been a shift in false positive and false negative rates. However, the trade-off during this shift is the model prediction accuracy which increased for the former case and decreased for the latter case. This is as expected since increasing the number of learners should make the accuracy of our prediction model stronger.

APPENDIX

Code for training the Classifier

```
%function [trainedClassifier, validationAccuracy] =
% Convert input to table
inputTable = array2table(class, 'VariableNames', {'column_1',
'column_2', 'column_3', 'column_4'});

predictorNames = {'column_1', 'column_2', 'column_3'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_4;
isCategoricalPredictor = [false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the
classifier.
template = templateTree(...
    'MaxNumSplits', 7536);
classificationEnsemble = fitcensemble(...
    predictors, ...
    response, ...
    'Method', 'Bag', ...
    'NumLearningCycles', 30, ...
    'Learners', template, ...
    'ClassNames', [1; 2; 3]);

% Create the result struct with predict function
predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
predictorNames);
ensemblePredictFcn = @(x) predict(classificationEnsemble, x);
trainedClassifier.predictFcn = @(x)
ensemblePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.ClassificationEnsemble = classificationEnsemble;
trainedClassifier.About = 'This struct is a trained model exported
from Classification Learner R2018a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a
new predictor column matrix, X, use: \n  yfit = c.predictFcn(X)
\nreplacing ''c'' with the name of the variable that is this struct,
e.g. ''trainedModel''. \n \nX must contain exactly 3 columns because
this model was trained using 3 predictors. \nX must contain only
predictor columns in exactly the same order and format as your
training \ndata. Do not include the response column or any columns
you did not import into the app. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map'')),
''appclassification_exportmodeltoworkspace'')">How to predict using
an exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
% Convert input to table
inputTable = array2table(class, 'VariableNames', {'column_1',
'column_2', 'column_3', 'column_4'});

predictorNames = {'column_1', 'column_2', 'column_3'};
predictors = inputTable(:, predictorNames);
response = inputTable.column_4;
isCategoricalPredictor = [false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationEnsemble,
'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] =
kfoldPredict(partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError');
yfit = trainedClassifier.predictFcn(trainedData)
C = confusionmat(realvalues,yfit)
```