A

PROJECT REPORT

ON

Assessment of Kinesiological Signals for Development for the Prediction of the Movement Activities for the Development of the Lower Limb Assistive Device

BY

Ms. Diksha Anoop Zutshi

Under the guidance of

Mrs. Sakshi Agarwal (Sc ‘D’) (DEBEL, DRDO)

IN FULFILMENT FOR THE DEGREE

OF

INTEGRATED MASTERS OF TECHNOLOGY IN BIOENGINEERING

SUBMITTED TO

Defence Bio-Engineering & Electro Medical Laboratory, DEBEL, DRDO

PREFACE

This report documents the work done during a yearlong Research Internship at the Defence Bioengineering and Electro Medical Laboratory, DRDO, Bangalore, under the supervision of Mrs. Sakshi Agarwal, Sc ‘D’.

The report first shall give an overview of the tasks completed during the period of internship with technical details. Then the results obtained shall be discussed and analysed chapter-vise. Various trials and protocols were worked upon and they shall be mentioned in the report.

I have tried my best to keep the report simple yet technically sound and covering the entire work which has been done. I hope I have succeeded in the attempt.

DECLARATION

I declare that this report reflects my original work about the subject in my own words. I have sufficiently cited and referenced the original sources, referred or considered in this work. I have not plagiarized or submitted the same work for the award of any other degree, plagiarism report is attached here with. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will lead to disciplinary action by the Institute.

Diksha

Diksha Anoop Zutshi

Date: 03-06-2022

Place: Defence Bioengineering and Electro Medical Laboratory, DRDO, Bangalore.

ACKNOWLEDGMENT

I would like to mention my deepest appreciations to my guide and mentor, Mrs. Sakshi Agarwal Sc ’D’, for always believing in me, for always giving me ample of opportunities to learn and grow, always having my best interests in mind and introducing me to the field that I now wish to pursue further. Her dynamism, vision, sincerity and motivation has deeply inspired me, teaching me the methodology of carrying out the research. It was a privilege and an honour to be able to carry research under her guidance. I would also like to thank Mr. Mohan Vellu for his guidance and support. I would like to thank (DEBEL), DRDO for providing all the resources to make this project happen. I would additionally like to thank Mr. Nagrajan, Dr. Manan , Dr. Daniel for all the help and enlightenment. I would like to thank Ms. Vinutha Sampat, SRF, for her guidance, support and for always giving an empathetic ear through it all. I would additionally like to thank Mr. Vinoth (JRF), Ms. Shikha (Apprentice), Ms. Rishika Basak, Mr. Jainam Verma, this wouldn’t have been possible without them.

I would like to express my deep and sincere gratitude to Dr. Renu Vyas, Head of the School, MIT School of Bioengineering, Sciences and Research, MIT-ADT University, for always seeing me forfeit for best of the opportunities, always guiding me with her experiences, being a good critique and pushing me always to think outside of the box.

ABSTRACT

Motion Intention Prediction is the forecast of the next action performed by the person based on the EMG signals, applications range from rehabilitative, industrial and defence purposes. The aim of conducting the study is to aid the soldiers with exoskeletons they are in control off, which will in turn assist them to increase their performance. The active exoskeletons have proven their reliance in terms of reducing the load which can be seen through decreased muscle activation amplitudes, metabolic activity, pressure in the FSR insoles, etc. With the intention prediction, exoskeletons can pick up the next course of actions and smoothen the process, without humans feeling any stress.

A minimum time window is selected in order to understand the change in the activity which is used by the algorithm to classify and differentiate between different activities. From the literature, about 30 to 100 ms long, time delay has been reported between the initiation of actual motion and the detection of EMG onset. From this 100 ms long window, one can decode the intention of motion, feeding this to the exoskeleton can smoothen the motion, hence, making the exoskeleton the master, assisting and not resisting the human body. Hence, boosting the physical strengths of the Javans.

Keywords: Motion Intention Prediction, Time Window, GAIT Cycle, Classification.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Evaluation of Efficiency of Exoskeleton

1.1: Introduction and Literature review:

An exoskeleton is a wearable gear that aids the performance of the operator, providing strength, speed, and endurance. They come under the class of articulated mechanical systems whose performance can only be estimated in close contact with the user. They are the orthoses that are designed for perfectly abled bodies augmenting the operator’s performance. Performance relies upon the factors such as selection of architecture, parameters, and the nature of the coupling to the human.

1.1.1. *Classification of Exoskeletons:*

They can be classified as active or passive, based on their operation and based on the body parts they are designed for. In an active exoskeleton, actuators are externally powered by electric, hydraulic, pneumatic, etc, sources. In a passive exoskeleton, unpowered materials such as springs, dampers, etc, are used, which store the energy from human motion and release it when required to aid support or motion. Based on application, they are mainly designed for industrial, rehabilitation, and military needs. Based on the specific body parts, they are designed for upper limbs, lower limbs, or a combination of parts. They have two operational modes: assistive and resistive. In the assistive mode, the exoskeleton provides power to support the movement of the human limb, while in the resistive mode it opposes motion/forces. EMG signals of primary muscle groups, with and without an exoskeleton, are analysed to assess the exoskeleton's effectiveness.

Upper limb exoskeletons are essentially being used for teleoperation and power amplification. The subject's weight is the main consideration in the lower limb exoskeleton, to analyse the extent of the subject's weight that can be sustained by the exoskeleton. The strain on the subject's limbs will be lower if the percentage weight sustained by the exoskeleton is higher.

During the 1960s and 1970s, the prime focus in this field was on the application of active exoskeletons for industry and medical purposes. It gradually shifted towards augmenting the strength of humans in the early 1990s. An evident shift has been seen in the focus in recent years towards rehabilitation and power assisting to overcome physical weakness. [1,2]

Evaluation of exoskeleton can be performed taking feedback of different biological signals such as: EMG, FSR, Metabolic output, etc. The springs in the exoskeleton make sure that they ground excess of pressure, hence giving lower force plate readings. Lower metabolic output signifies, lower energy expensed by the body while performing certain task and in case of EMG, lower muscle activation amplitudes signify lower efforts required to perform certain task.

1.1.2. *Electromyograph:*

A motor unit is the smallest functional unit of a muscle. It consists of the motor neuron, its axon and all the muscle fibres that are innervated by its branches. When motor units are activated, the corresponding muscle fibres contract. In general, an action potential is a transient variation of the membrane potential of a cell membrane of an excitable cell due to the activity of voltage gated ion channels present in the membrane. An electromyography (EMG) is a measurement of the electrical activity in muscles as a by-product of contraction

The EMG is generated when a motor neuron action potential from the spinal cord arrives at a motor end plate. Its arrival causes a release of ACh (Acetylcholine) at the synaptic cleft which causes a depolarization (Action Potential). This action potential electrically travels downward from the surface in a transverse tubule. This in turn causes a release of Ca^{++}, causing cross-bridge binding and the sarcomere of the muscle to contract. Electromyography (EMG) refers to the collective electric signal from muscles, which is controlled by the nervous system and produced during muscle contraction. The motor unit action potentials of all active motor units detectable under the electrode site are electrically superposed.

The electrical source is the muscle membrane potential of about -90mV. An EMG is the summation of action potentials from the muscle fibers under the electrodes placed on the skin. The more muscles that fire, the greater the amount of action potentials recorded and the greater the EMG reading. The resulting waveform obtained is called the electromyogram.

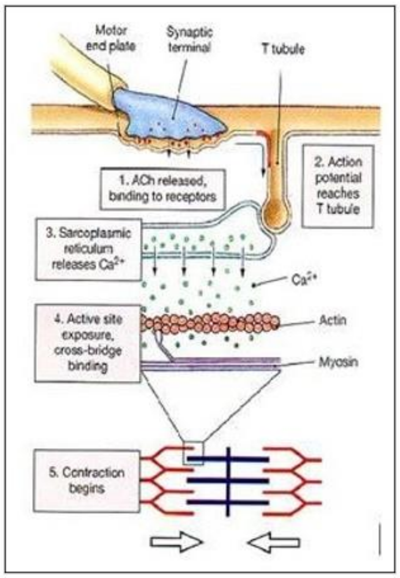

Motor end plate
Synaptic terminal
T tubule
1. ACh released, binding to receptors
2. Action potential reaches T tubule
3. Sarcoplasmic reticulum releases Ca^{2+}
Ca^{2+}
4. Active site exposure, cross-bridge binding
Actin
Myosin
5. Contraction begins

Figure 1.1: Muscle Contraction Process

Measured EMG potentials range between less than 50 µV and up to 20 to 30 mV, depending on the muscle under observation. EMG signals have frequencies in the range between 0-500Hz, and it is most prominent between 10-250Hz.

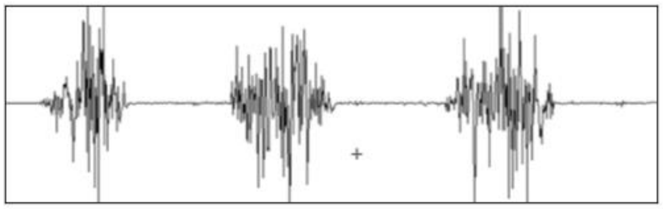


Figure 1.2: Raw EMG Signal

1.2: Methodology:

The aim is to analyse the performance of passive and active exoskeleton using EMG signals.

1.2.1. EMG Signal Acquisition:

Commercial EMG recording system (FREEEMG RT+, BTS Bioengineering, UK) was used. After the placement of the electrodes at the desired muscles, the system communicates with a computer through the USB receivers and can manage up to 20 probes simultaneously. Each probe is equipped with internal memory to ensure uninterrupted recording in case of temporary connection loss and to allow for acquisitions in wide spaces and open fields. For signal acquisition, the probes are directly attached to the pre-gelled electrodes, with no need for additional fastening.

Application of surface EMG electrodes requires proper skin preparation beforehand. In order to obtain a good quality EMG signal, the skin's impedance must be considerably reduced. For this purpose, the dead cells on the skin e.g., hair must be completely removed from the location where the EMG electrodes are to be placed. It is advisable to use an abrasive gel to reduce the dry layer of the skin.

1.2.2. Signal Filtration and processing:

The signals obtained using the surface electrodes and the wireless EMG transmitter contains noise which has to be eliminated in order to perform effective analysis. Thus, signal filtering techniques are employed.

Signal is visualized, filtered and analyzed using MATLAB. A high pass filter of 20Hz, a low pass filter of 500Hz and a linear envelope of 2Hz low pass was employed and then the resultant signal is rectified.

```
% Highpass filter
Highpass = 20;
Wo = Highpass/NyqFreq;
[D,C] = butter (4,Wo,'high');
% run filter
high_pass_filtdata = filtfilt(D,C,data);

% Lowpass filter
Lowpass = 500;
W1o = Lowpass /NyqFreq;
[D1,C1] = butter (4,W1o,'low');
% run filter
band_pass_filtdata = filtfilt(D1,C1, high_pass_filtdata);


% Linear envelope (2Hz lowpass)
LP = 2;
Wp = LP/NyqFreq;
[F,E] = butter (4,Wp,'low');
linear_data = filtfilt(F,E, band_pass_filtdata);
```

Data smoothening and rectification:

```
%Rectification
R=abs(linear_data);

%Moving average
M=movmean(R,500);

%RMS Analysis
RMS_RIGHT_TIB_ANT=rms(R);
```

1.2.3. Signal Analysis:

Many of the previous studies have validated that the use of the exoskeleton can reduce the load on the muscles, which can be seen in the reduced muscle sEMG activity.12% to 15% reduction in the activity was noticed in Erector Spinae using the back exoskeleton, using loads of 7.5 kgs to 15 kgs.[6] In one of the researches, a powered ankle exoskeleton has been used to show the load reduction on the muscle using the metabolic activity data.[7] A passive exoskeleton in trunk bending activity showed about a 44% reduction in the mean EMG amplitude for the upper back muscle activity and a 20% reduction for the bicep femoris activity.[3,4]

1.2.4. Experimentation:

1.2.4.1. Passive Exoskeleton:

The experiment protocol was designed based on the literature review and methodology. The EMG data for processing and analysis is acquired real time using a Freeemg1000 kit. The acquired data is later, analyzed using the EMG analyzer software. The muscles of interest –

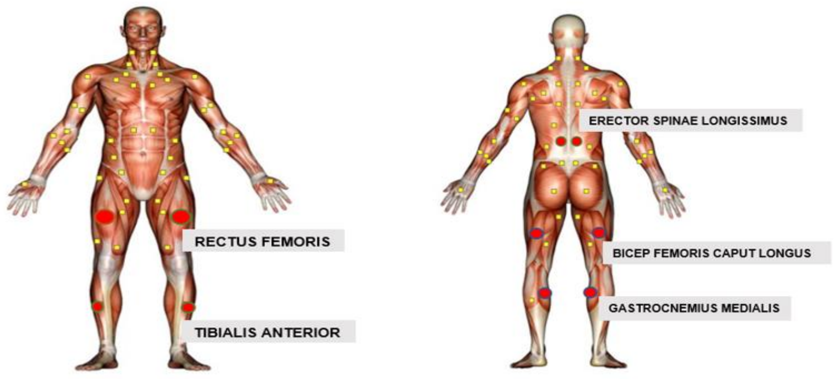


Figure 1.3: Muscles of Interest

The subject was made to wear comfortable clothing and the EMG electrodes were set up according to the set protocol for the 10 muscles of interest. The trials were set as follows-

Trials for Standing –

1. No load_No exoskeleton(reference)
2. Load_exoskeleton
3. No load_exoskeleton
4. Load_no exoskeleton

Similar 4 of these trials were planned in walking phase as well.

In the walking phase the subject according to the set trial walked on the treadmill for 5 min with a speed of about 2Km/hr and the EMG signals were obtained accordingly and saved for further processing.

The surface electrodes are placed on the individual's upper and lower limbs according to the protocol. The collected EMG signals are wirelessly transmitted to the EMG ANALYSER kit. The individual is asked to stand in an upright position or walk according to the trial .

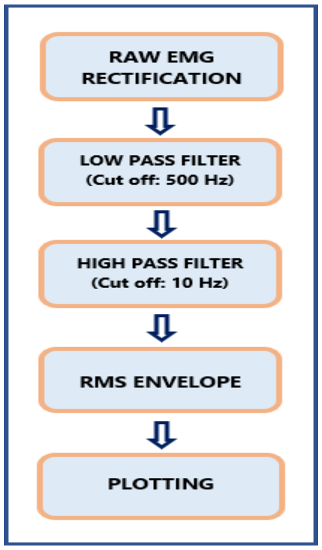

RAW EMG RECTIFICATION
LOW PASS FILTER (Cut off: 500 Hz)
HIGH PASS FILTER (Cut off: 10 Hz)
RMS ENVELOPE
PLOTTING

Figure 1.4: Protocol

1.2.4.2. Active Exoskeleton:

EMG signals from 10 different muscles namely – [left and right tibialis anterior, left and right gastrocnemius medialis, left and right rectus femoris, left and right erector spinae longissimus, left and right biceps femoris caput longus] were acquired in the signal visualization software and were bandpass filtered (10-500Hz) and rectified.

The system communicates with a computer through the USB receivers and can manage up to 20 probes simultaneously. Each probe is equipped with internal memory to ensure uninterrupted recording in case of temporary connection loss and to allow for acquisitions in wide spaces and open fields. For signal acquisition, the probes are directly attached to the pre-gelled electrodes, with no need for additional fastening.

The protocol was developed for three trial conditions, performed at the gap of 5 mins, which was repeated for 5 sessions: a. walking on a treadmill at 2km/hr for 5 minutes without the exoskeleton. b. walking on a treadmill at 2km/hr for 5 minutes with the exoskeleton turned off. c. walking on a treadmill at 2km/hr for 5 minutes with the exoskeleton turned on.

Each trial was set to a duration of T=300 s and within this duration, the EMG signals of all the 10 muscles were acquired on the analyser in real-time. The subject was also wearing the COSMED metabolic analyser for a detailed acquisition of the heart rate, stress test ECG, metabolic exercise and rest on its software.

1.3 : **Results and Discussions:**

1.3.1. Passive Exoskeleton:

Passive Exoskeleton divides the loads and shares between the adjacent muscles, hence, the results that we see aren't effective with respect to all the muscles. Here, no external energy is used, the energy generated by the body for movements are stored in its spring and later used to ease the movement.

The mechanism of passive exoskeleton includes a gas spring which is located at the hip joint. The spring compresses and releases during walking, this compression requires an extra effort given by the subject. Because of this reason a counter effect is observed in the EMG signals and thus EMG cannot be used as an effetive parameter for evaluating passive exoskeleton. [5]

MATLAB CODE : To plot EMG signals in No load No Exoskeleton condition over Load No Exoskeleton condition over Load Exoskeleton condition.

```
figure;
subplot(5,1,1);
plot(Time1, TIB_ANT_NL_NE,'-b');hold on;
plot(Time2, TIB_ANT_L_NE, '-r');hold on;
plot(Time3, TIB_ANT_L_E, '-g');xlabel('Time in s');ylabel('Amplitude');
title('Tibialis Anterior');legend('NL-NE','L-NE','L-E','Location','NorthEast');
```

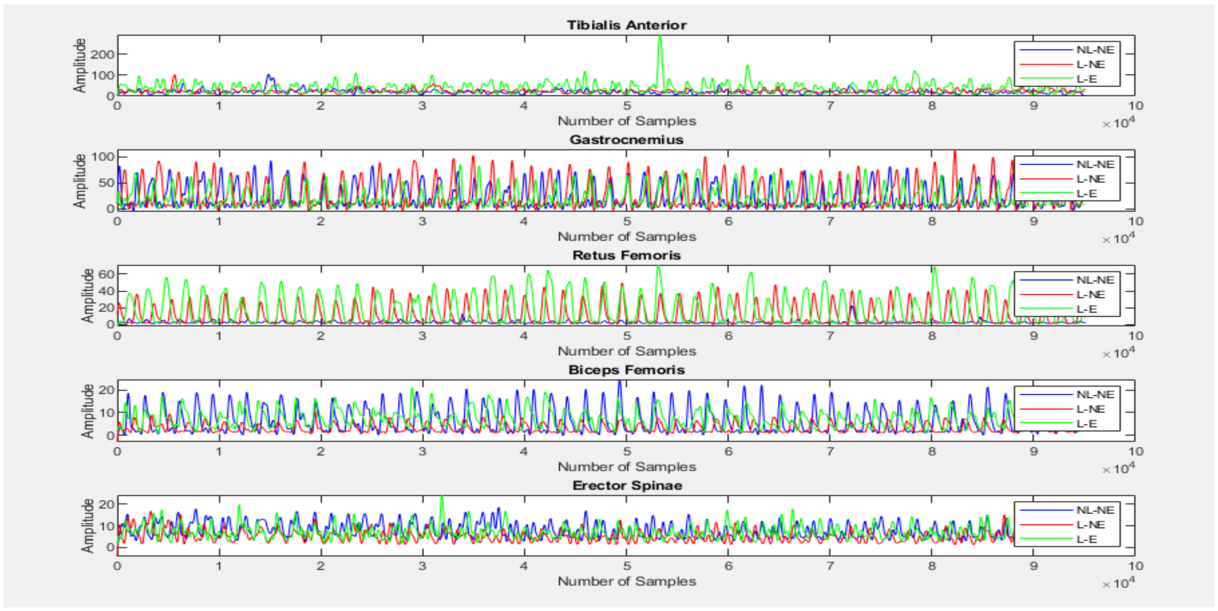


Figure 1.5: EMG Output for Varying Muscles Wearing the Passive Exoskeleton in three States

No load – No exoskeleton is the normal condition, load – no exoskeleton is where the candidate was made to carry the load without wearing the exoskeleton and load and exoskeleton is the condition where load was carried and exoskeleton was worn.

Here, we can observe its effectiveness with respect to Bicep Femoris and Gastrocnemius muscles.

1.3.2. Active Exoskeleton:

Active exoskeleton is the one with motors and is externally powered, its highly effective and huge reduction in muscle amplitude is observed after wearing it.

MATLAB CODE : To plot EMG signals while wearing No Exoskeleton over wearing Exoskeleton.

```
figure;
subplot(5,1,1);
plot( TIB_ANT_exo_on,'-b');hold on;
plot( TIB_ANT_exo_off, '-r');hold on;
xlabel('No. of samples');ylabel('Amplitude');
title('Tibialis Anterior');legend('EXO-on','EXO-off','Location','NorthEast');
```

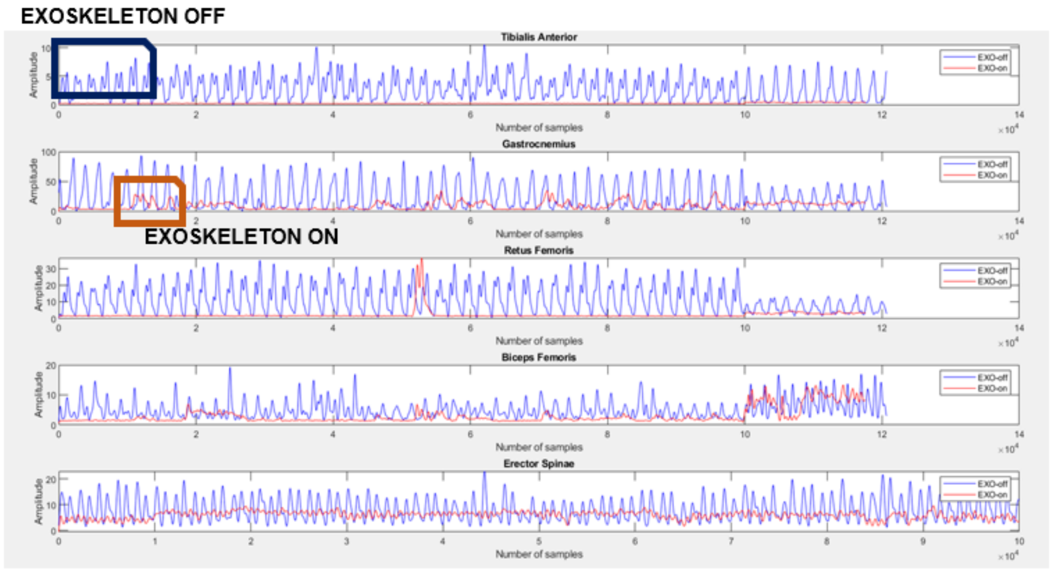


Figure 1.6: EMG output for varying muscles wearing the active exoskeleton in two states

The plot above is a time-domain representation, prompting the exoskeleton's switched off and on state in blue and red respectively. Here, it can be observed that there is a decrease in the amplitude, once the exoskeleton is worn.

The comparison between no exoskeleton and the exoskeletons on state, for all the considered muscles, for two subjects, is shown in fig 6, indicating the effectiveness of the exoskeleton in consideration, in reducing the effect of the load, which is prompted through amplitude values.

The calculated average RMS of the amplitudes for two subjects during switched-off and switched-on states of the exoskeleton are in the table below.

NAME OF THE MUSCLE	**RMS NO EXOSKELETON (Volts)**	**RMS EXOSKELETON ON (Volts)**
Tibialis Anterior	$16.64 * 10^{-6}$	$6.60 * 10^{-6}$
Gastrocnemius Medialis	$19.71 * 10^{-6}$	$10.75 * 10^{-6}$
Rectus Femoris	$5.20 * 10^{-6}$	$3.66 * 10^{-6}$
Bicep Femoris	$5.42 * 10^{-6}$	$3.91 * 10^{-6}$
Erector Spinae	$13.63 * 10^{-6}$	$8.58 * 10^{-6}$

Table 1.1: RMS Values

The decrease in RMS value observed in muscle activity in the switched-on state as compared to the no exoskeleton state prompts the effectiveness of the exoskeleton with respect to the muscles in consideration, which is, 60.3% for Tibialis Anterior, 45.45% for Gastrocnemius Medialis, 29.61% for Rectus Femoris, 27.85% Bicep Femoris Caput Longus, and 37.05% for Erector Spinae Longissimus.

From the above RMS analysis of the amplitudes in no exoskeleton and exoskeleton's switched on state, for two subjects, it can be observed that the Tibialis Anterior and Gastrocnemius Medialis are the major load-bearing muscles, as the most significant decrease in amplitudes is observed with respect to them, on the contrary, it is evident that the Bicep Femoris Caput Longus and Erector Spinae Longissimus show a much lower fall in amplitudes, hence proving their comparatively little involvement in load-bearing.

1.4 Conclusion:

It can be qualitatively concluded that the exoskeleton developed by Defence Bioengineering and Electromedical Laboratory (DEBEL), DRDO, Bangalore, has shown great potential and has considerably proven to be very efficient in reducing muscle activity, hence reducing the effect of load on the muscles, providing large scope for further quantitative studies and future development. However, it is observed that there is a huge delay between the user's activity and exoskeleton's response, and this needs to be addressed in order to increase the efficiency of the exoskeleton. From the literature it has been found that implementing machine learning and deep learning models to kinesiological signals can help bridge this gap. A study has been carried out in the subsequent chapters for implementing human motion prediction through machine learning applications.

1.5 References:

1. Spada S, Ghibaudo L, Carnazzo C, Di Pardo M, Chander DS, Gastaldi L, Cavatorta MP. Physical and virtual assessment of a passive exoskeleton. InCongress of the International Ergonomics Association 2018 Aug 26 (pp. 247-257). Springer, Cham.
2. Agarwal P, Narayanan MS, Lee LF, Mendel F, Krovi VN. Simulation-based design of exoskeletons using musculoskeletal analysis. InInternational Design Engineering Technical Conferences and Computers and Information in Engineering Conference 2010 Jan 1 (Vol. 44113, pp. 1357-1364).
3. Bosch T, van Eck J, Knitel K, de Looze M. The effects of a passive exoskeleton on muscle activity, discomfort and endurance time in forward bending work. Applied ergonomics. 2016 May 1;54:212-7.
4. Cram JR. The history of surface electromyography. Applied psychophysiology and biofeedback. 2003 Jun;28(2):81-91.
5. Bosch T, van Eck J, Knitel K, de Looze M. The effects of a passive exoskeleton on muscle activity, discomfort and endurance time in forward bending work. Applied ergonomics. 2016 May 1;54:212-7.

Chapter 2

Fatigue Analysis

2.1: Introduction and Literature review:

Motor unit potential (MUP) is the culmination of multiple muscle fibres (MFs). All the MFs within a motor unit being homogenous generates similar-sized action potentials. MUPs recorded from surface electrodes are lower in amplitude as compared to intra-muscular recordings. Compound Motor Action Potential (CMAP) is the culmination of all the MUPs, that is the summation of all the motor units agitated within the muscle, hence being the compound derivative signal of all MFs. As the stimulus increases so does the signal.[1]

As the output requirement of the muscle increases, that is the work demanded by the muscle, so do the active motor units that fire, it is called the Recruitment Process, which is governed by Henneman's size principle, depending on the size of the action potential. The firing of different motor units is independent of each other.

Motor neurons are recruited based on their excitabilities in an orderly fashion from low to high axonal velocity. Motor neurons belonging to a motor nucleus or multiple motor nuclei, organize themselves according to the hierarchy of their excitation by synaptic excitation, creating a motor pool or task group. Motor units (motor neurons and muscle fibres) are recruited from slower to faster, fatigue-resistant to more susceptible to fatigue. [2]

2.1.1. Major upper limb muscles in load carrying and sustaining:

Flexor Carpi Radialis: It is a thin and superficial muscle located in the forearm, it helps in flexion and abduction of the hand and wrist, it also helps in the dynamic stabilization of the scaphoid.

Bicep Brachii Caput Brevis: It is responsible for the movement of the elbow, radio-ulnar, and glenohumeral joints. It is present towards the anterior side of the arm.

Triceps Brachii Caput Lateralis: It plays a major role in extension at the elbow joint and extension along with abduction at the shoulder. It is an arm muscle constituting three heads. [3]

2.1.2. Muscle Contractions:

Isometric contractions are the ones where there is no change in muscle length and joint angles, it constitutes varying tension and energy, for example, planks. In Isotonic contractions, the force is generated against a static resistance by varying the length of the muscle, for example, such contractions can be seen in dumbbell bicep curls. It consists of two kinds of contractions, namely concentric and eccentric. EMG signals from these contractions are further used to control assistive robots, exoskeletons. [4]

2.1.3. Fatigue Analysis:

The increased load can be seen as the increase in the EMG amplitude. In the last 25 years, sEMG has been used as a tool to estimate the degree of muscle fatigue. Muscle fatigue can be widely divided into three headings, central fatigue, fatigue of the neuromuscular junction, and muscle fatigue. Muscle fatigue or local muscle fatigue is sometimes called neuromuscular fatigue.

The struggle to perform a task at a certain intensity as the induction of fatigue lowers the capacity temporarily. (Knowlton et al., 1951) Addresses the increase in the amplitude of EMG during fatigue induction. Kogi and Hakamada (1962) analyzed the frequency of the EMG power spectrum for fatigue analysis and found a shift towards lower frequencies at the induction of fatigue.

Muscle contraction and concentration of lactic acid are directly proportional, the level of force exerted by the muscle and the type of contractions also determine the overall concentration of lactate. An increase in lactate concentration leads to a change in pH which holds as a prime reason for induction of fatigue. This results in a decrease in muscle fiber conduction velocity and changes in the shape of the motor unit action potential (MUAP) waveform. As it is widely known that sEMG is generated from all the MUAPs as an interference pattern, hence the changes are visible in the waveform as the increase in amplitude and power spectrum shift towards lower frequencies as a decrease is observed in median frequency. [5,6]

2.2: Methodology:

Here, our aim is to analyse the muscle fatigue EMG signals.

2.2.1. Acquisition protocol:

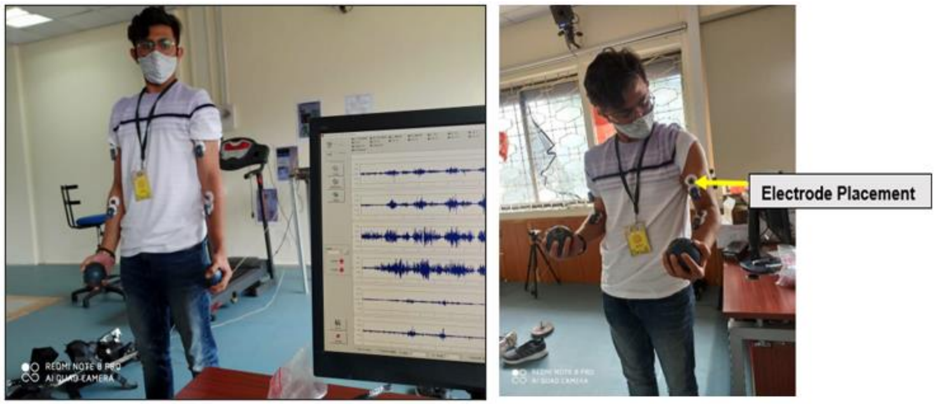


Figure 2.1: Electrode Placement

For reference isotonic exercise were conducted with a subject to analyse the change in the signal when fatigue is induced in the muscle. The 3 muscles of biceps and triceps were selected and EMG electrode were placed.

After setting up the acquisition software the subject was made to hold a metal ball of 3 kg and was made to perform dumbbell bicep curl exercise until maximum volatile contraction is attained and the muscles feel the maximum tension.

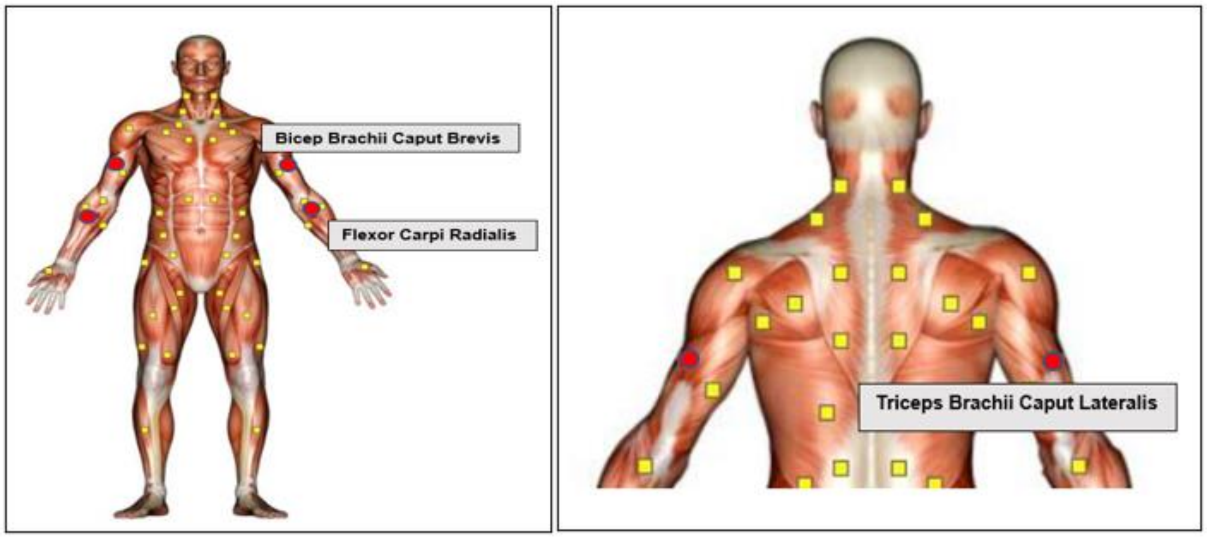


Figure 2.2: Muscles for EMG Acquisition

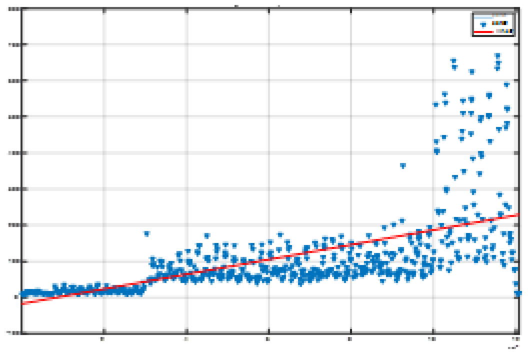
2.3: Results and Discussions:

As it is widely known that sEMG is generated from all the MUAPs as an interference pattern, hence, the changes are visible in the waveform as the increase in amplitude and power spectrum shift towards lower frequencies as a decrease is observed in median frequency, when fatigue is induced

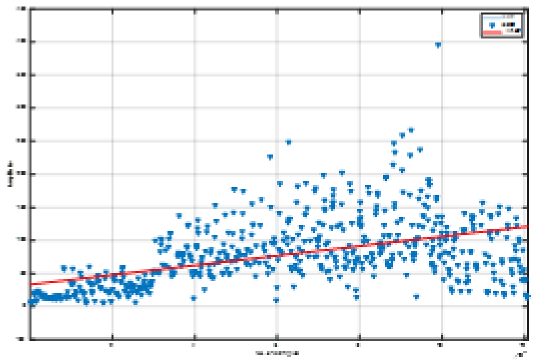
2.3.1. Time Domain Analysis:

MATLAB CODE FOR TIME DOMAIN ANALYSIS:

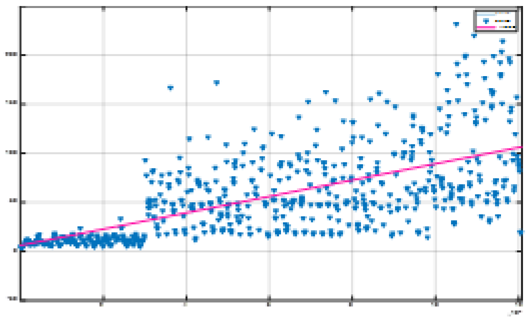
```
medFilt = dsp.MedianFilter(200);
y = medFilt(exo_off);
findpeaks(y)
```



Flexor Carpi Radialis



Bicep Brachii Caput Brevis



Tricep Brachii Caput Lateralis

Figure 2.3: Fatigue Analysis

Here, these plots show a trend of amplitudes plotted against number of samples.
The increasing trend in the regression plot hence proving the onset of muscle fatigue with respect to time.

2.4: Conclusion:

It can be successfully concluded that we have observed and analysed the fatigue, induced by the rigorous exercise of biceps and triceps.

2.5: References:

1. Raghavan M, Fee D, Barkhaus PE. Generation and propagation of the action potential. Handbook of clinical neurology. 2019 Jan 1;160:3-22.
2. Cope TC, Sokoloff AJ. Orderly recruitment among motoneurons supplying different muscles. Journal of Physiology-Paris. 1999 Jan 1;93(1-2):81-5.
3. Nazmi N, Abdul Rahman MA, Yamamoto SI, Ahmad SA, Zamzuri H, Mazlan SA. A review of classification techniques of EMG signals during isotonic and isometric contractions. Sensors. 2016 Aug;16(8):1304.
4. Cifrek M, Medved V, Tonković S, Ostojić S. Surface EMG based muscle fatigue evaluation in biomechanics. Clinical biomechanics. 2009 May 1;24(4):327-40.
5. Petrofsky JS. Frequency and amplitude analysis of the EMG during exercise on the bicycle ergometer. European Journal of Applied Physiology and Occupational Physiology. 1979 Mar;41(1):1-5.

Chapter 3

Electromotive Delay

3.1: Introduction and Literature review:

Electromechanical Delay is the interval between the EMG onset and the onset of the resulting change in the mechanical variable.[1] It's the interval between the receiving the at the motor end plate and chance of the electrical activity across the sarcolemma, it's also measured as the time interval between the change in the electrical activity and the movement. EMD interval is also referred to as Electro-Mechanical Response Time.[2] It's the lag between the onset of myoelectric activity and tension development in a muscle contraction. It consists of the time required for the conduction of action potentials along the sarcolemma and t-tubules, the release of C^{2+} from sarcoplasmic reticulum and the subsequent formation of the cross bridges between myosin and actin filament and development of tension in contractile components and stretching in series elastic components. [3] It's the delay between the muscle electrical simulation and the onset of muscle fascicle motion, it consists of electro-chemical processes and it consists of delay between onset of muscle fascicle motion, onset of myotendinous junction motion and the force production.[4]

EMD also varies with muscle elastic properties. Strenuous exercise accompanies increased body temperatures and changes in contractile and elastic properties of the muscles and hence the EMD is altered. EMD is suggested to play an important role in the evaluation of the muscle's coordination, its found that the interval is increased by 20ms in fatigued muscles because of impaired muscle conductive, contractive and elastic properties and increased body temperature.[3] It was observed that EMD is shorter in an eccentric exercise than concentric because of rate of change of length of series of elastic elements of the muscle.

In an experiment the forearm was moved back and forth before the maximal voluntary contraction. In the concentric condition the series of elastic contractions had to catch up to and exceed the passive velocity of the shortening of the muscle before the transducer detected any change in the force. The passive lengthening of the muscle added to the velocity stretch of the series of elastic component. Hence, the detectable force was reached sooner in eccentric than in concentric contractions. These factors which leads to shortening of the time in the detection of the force, is responsible for shortening of EMD interval. Factor responsible: (i) Fast movements: They activate motor units with fast force rise times, hence have shorter EMD intervals than slow movements. (ii) Muscles that comprises of fast twitch fibres yield shorter EMD intervals. (iii) In eccentric contraction, there is a possibility of stretch reflexes augmenting the activation alpha neuron pool, which results in manifestation of shorter EMD intervals. (iv) Increased muscle stiffness because of compression and consequent restriction of the fluid flow in the transverse tubules. This presence of viscosity should shorten the EMD delay as presence of viscosity stiffens the muscles/ tendon/ bone linkage as a function of rate of change of length.[5]

3.2: Methodology:

The conducted experiment in DEBEL, DRDO, GAIT lab, where the lag between the EMG onset and the onset of the acceleration was captured and measured for different lower body muscles in consideration.

3.2.1. Muscles in consideration:

- Tibialis Anterior
- Gastrocnemius Medialis
- Rectus Femoris
- Sartorius
- Vastus Lateralis
- Vastus Medialis
- Semitendinosus
- Bicep Femoris

3.2.2. Systems used:

For EMG acquisition: FREEEMG RT+BTS Bioengineering, UK

For the acquisition of acceleration: G sensors: BTS Bioengineering

3.2.3. Experiment protocol:

In total 5 subjects were prepared, with trials performed twice on each of them. The subjects were made to wear comfortable clothing and the EMG electrodes were set up according to the set protocol for the 8 muscles of interest. The trials were set as follows-

The subject was asked to walk and stop on commands in a straigh line,10 trials were conducted, for about 2 mins each, with the stops in every 20 secs.

In alternative trials with a person, a person is first asked to initiate the walk with the same leg which has the g-sensor attached, therefore, accelerometer will first record the swing phase and then the stance phase and then initiate the second trail with the opposite leg, here the accelerometer will record the stance phase first and then the swing phase.

G-sensor is placed little below the knee, above the shank area.

There’s virtually no lag in the recording of both the signals as it can be simultaneously be acquired from BTS Software.

3.4: Results and Discussions:

Lower values of accelerometer imply the stance phase and higher values imply the swing phase, if the walk is started with the leg which has the g-sensor attached and vice-versa if the walk is initiated with the leg opposite to the leg with g-sensor attached.

3.3.1. EMD interval with respect of stance and swing phase:

Here, the leg same as to the G-sensor is in consideration:

The plot above is EMG for Tibialis Anterior and Gastrocnemius Medialis, Tibialis being red and Gastrocnemius being blue.

The plot below shows acceleration signal with respect to the EMG

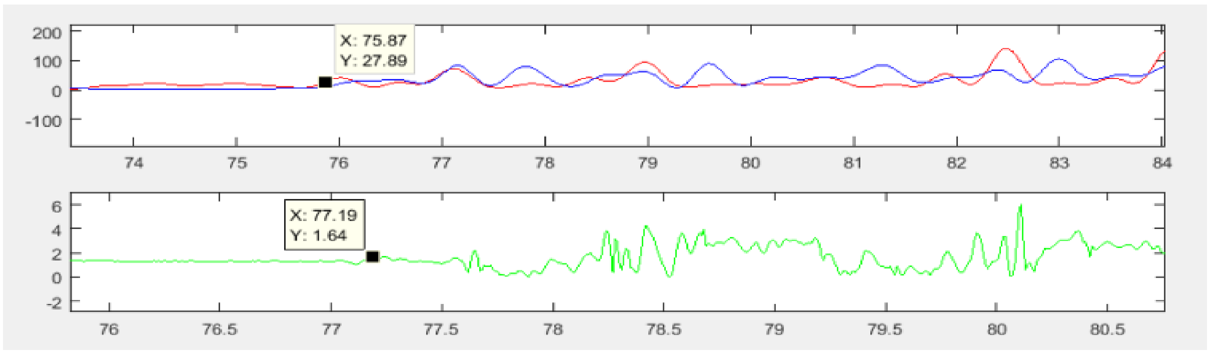


Figure 3.1: EMD Interval

Here, we can observe that Tibialis starts before Gastrocnemius, suggesting the leg in consideration was in swing phase initially.

Which can even be confirmed with a high and then a low in the accelerometer signal.

The time delay between EMG and Acceleration signals is observed about 132ms.

3.4: Conclusion:

It can be positively said that we have observed and analyzed the electromechanical response time for various lower limb superficial muscles in consideration. Having found EMD for different muscles, we know using this as intension of action, we can encode it and train classification model to predict the next action. Hence, making the exoskeleton in-charge assisting and not resisting the human body in anyway and boosting the Javan's strengths.

3.5: References:

1. Wolf W, Staude G, Appel U, Dengler R. Is the electromechanical delay an experimental artifact?
2. Winter EM, Brookes FB. Electromechanical response times and muscle elasticity in men and women. European journal of applied physiology and occupational physiology. 1991 Aug;63(2):124-8.

3. Zhou S, Carey MF, Snow RJ, Lawson DL, Morrison WE. Effects of muscle fatigue and temperature on electromechanical delay. Electromyography and clinical neurophysiology. 1998 Mar 1;38:67-74.

4. Lacourpaille L, Hug F, Nordez A. Influence of passive muscle tension on electromechanical delay in humans. PloS one. 2013 Jan 4;8(1):e53159.

5. Norman RW, Komi PV. Electromechanical delay in skeletal muscle under normal movement conditions. Acta Physiologica Scandinavica. 1979 Jul;106(3):241-8

Chapter 4

Introduction to Motion Intention Prediction

4.1: Introduction and Literature review:

For the recognition of motion and activities is to analyse the bio-signals acquired through on body sensors. This technique represents a powerful tool for figuring out the patterns in the biosignals.[1] Motion planning algorithms are used to compute collision-free paths. To develop reliable planning algorithm that can handle environmental uncertainties and dynamic motions. In order to compute reliable motion trajectories, its important to gather the state of the humans to predict the motion.[2]

The exoskeleton control systems are divided in three categories: high-level, mid-level and the low-level. The high-level control systems are used to monitor the movement and recognize the intension. The middle-level control systems convert the human intensions to control trajectories. The low-level control systems are adopted to achieve precise control over the actuators. The efficiency of the exoskeleton is determined by the accuracy and the timeliness of the human intension recognition. s-EMG has been incorporated to increase this efficiency. Current crucial challenges faced with this method: 1) Time Delay 2) s-EMG being non-stationary.

The movement is captured after it already happened due to the delay induced by the sensors, hence, it is difficult to have exact match between the exoskeleton and the human movement. The research conducted in the field, considers a certain acceptable amount of delay. One of the studies conducted by Young et al. used a 300ms window, for s-EMG and motion signals to obtain high classification accuracy in the motion recognition. In some other related studies, researchers have tried to strike a balance between accuracy and delay, as the algorithm delay and the mechanical system delay will make the actual motion the actuator unable to track the designed trajectory. In order to deal with this time delay, HeHuang proposed a locomotion-mode prediction method during mode transitions. By predicting the movement mode, the artificial knee controller can switch the control parameter (such as impedance) in time. The prosthesis can perform seamless and safe locomotion-mode transitions. Another researcher, Pew et al., proposed a predictive model that uses IMU and EMG signals to predict turn intensions 400ms in advance with 95% accuracy. Though these algorithms can only predict in transition state and can't predict in the stable state of walking.

The non-stationary nature of s-EMG increases the uncertainties in the motion intension recognition algorithms, this refers to the change in the muscle characteristics due to individualistic differences, muscle fatigue and human adaption. These differences manifest as the mutations in the s-EMG because of individualistic anatomical differences. Muscle fatigue and human adaptions are shown as slow chances in the EMG distribution over time. In order to mitigate the issues with non-stationary characteristics, research needs to b focused on feature engineering of s-EMG. Spanias et al. used log-likelihood metric to detect s-EMG disturbances, the classifier in use detected and discarded the disturbances. The reliability of the classification algorithms can be guaranteed by giving up on the unqualified s-EMG.[3]

4.2: Methodology:

The conducted experiment in DEBEL, DRDO, GAIT lab, where walking trials were taken for 10 subjects with G-sensor and s-EMG sensors and was used to characterize and predict swing and stance phases of a GAIT cycle using Artificial Neural Networks. The data was prepared using pre-processing and feature engineering steps.

In one of the past researches, ANN was used to predict for robotic hand. The myoelectric controller was designed to provide prediction before the real execution of the hand grasp task. therefore, exploiting the EMG signal temporal window going from muscle activation to kinematic effective movement, i.e., the electromechanical delay phase. [4]

The functional task is measured in the 100ms window. The task classifier architecture was based on sequence of ANNs, data which was feature engineered, under three labels was provided as inputs:

1. Swing Phase

2. Stance Phase

3. Double Support Phase

The data was thresholded with respect to classes:

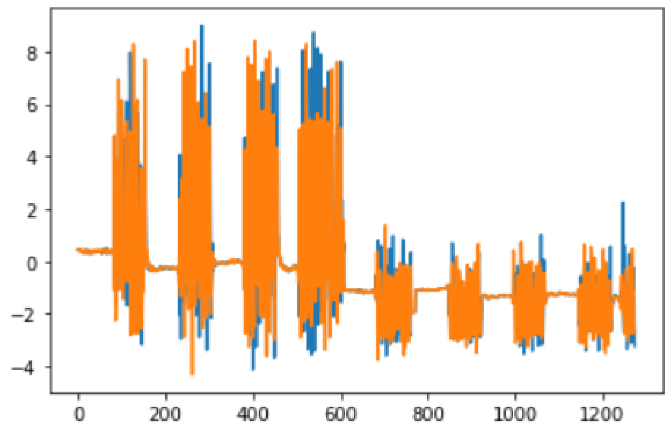


Figure 4.1: Two acceleration signals plotted together.

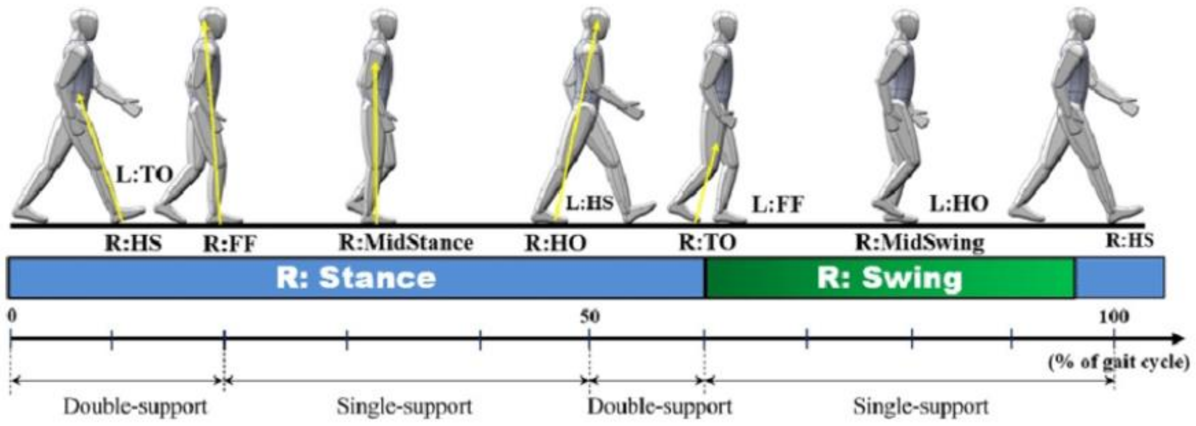
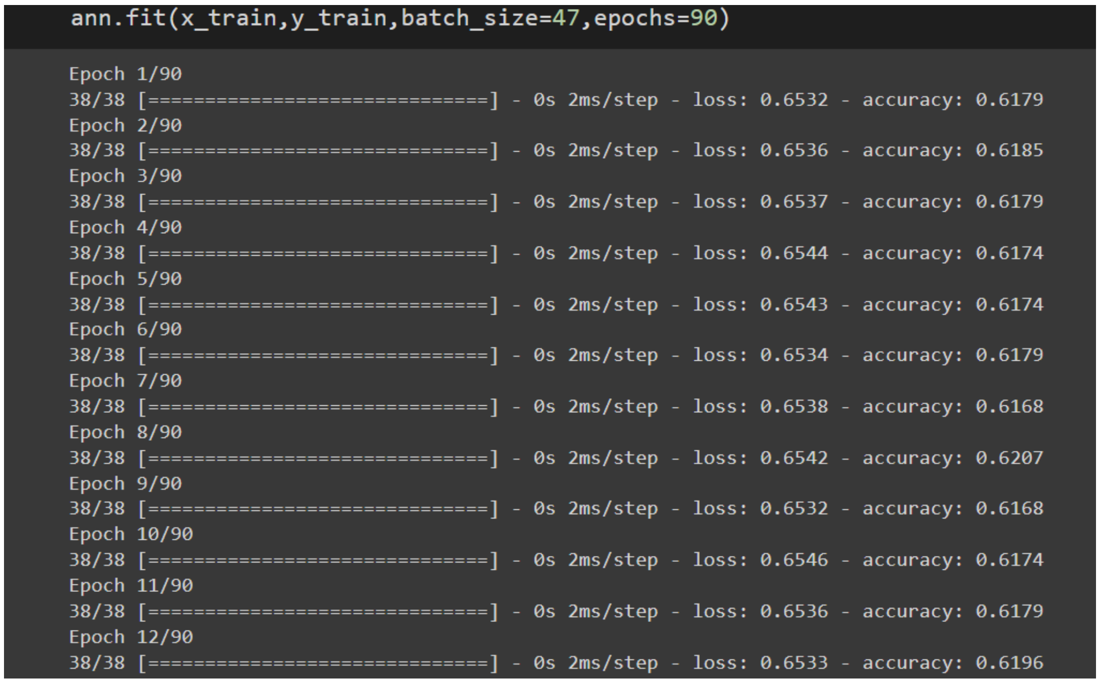


Figure 4.2: Complete GAIT Cycle

4.3: Results and Discussions:



```
ann.fit(x_train,y_train,batch_size=47,epochs=90)

Epoch 1/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6532 - accuracy: 0.6179
Epoch 2/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6536 - accuracy: 0.6185
Epoch 3/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6537 - accuracy: 0.6179
Epoch 4/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6544 - accuracy: 0.6174
Epoch 5/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6543 - accuracy: 0.6174
Epoch 6/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6534 - accuracy: 0.6179
Epoch 7/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6538 - accuracy: 0.6168
Epoch 8/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6542 - accuracy: 0.6207
Epoch 9/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6532 - accuracy: 0.6168
Epoch 10/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6546 - accuracy: 0.6174
Epoch 11/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6536 - accuracy: 0.6179
Epoch 12/90
38/38 [==============================] - 0s 2ms/step - loss: 0.6533 - accuracy: 0.6196
```

Figure 4.3: ANN Model

Model Accuracy without incorporation of Pre-processing and data cleaning steps: 61%

4.4: Conclusion:

Once we incorporate feature engineering and pre-processing and use clustering method for data binning into classes, we will see an increase in model accuracy.

4.5: References:

1. Badawi AA, Al-Kabbany A, Shaban HA. Sensor Type, Axis, and Position-Based Fusion and Feature Selection for Multimodal Human Daily Activity Recognition in Wearable Body Sensor Networks. Journal of Healthcare Engineering. 2020 Jun 7;2020.
2. Park JS, Park C, Manocha D. Intention-Aware Motion Planning Using Learning Based Human Motion Prediction. InRobotics: Science and Systems 2017 Jul.
3. Ding Z, Yang C, Wang Z, Yin X, Jiang F. Online Adaptive Prediction of Human Motion Intention Based on sEMG. Sensors. 2021 Jan;21(8):2882.
4. Gandolla M, Ferrante S, Ferrigno G, Baldassini D, Molteni F, Guanziroli E, Cotti Cottini M, Seneci C, Pedrocchi A. Artificial neural network EMG classifier for functional hand grasp movements prediction. Journal of International Medical Research. 2017 Dec;45(6):1831-47

Chapter 5

Classification of Walking with different weights using EMG and IMU Data

5.1: Introduction and Literature review:

5.1.1. Types of machine learning:

These models can learn patterns and use them further to take decisions and make predictions for unseen data. These models can be taught to recognize objects, for this, such models need to be trained and optimized. These models involve automating the repetitive decisions making and

repetitive evaluations tasks, providing consistent results. The resulting function includes rules and data structures hence, creating a trained machine learning model.
Types of Machine Learning Model: Supervised Model, Unsupervised Model and Reinforcement Model.

Supervised Model:
Here, both, input and target variables are provided as input to the model and model learns to map patterns between the input and target variables. They can be of three types: Classification and Regression. Classification Models, the target of prediction is a class or category label, that input variable would fall under, e.g.: mail being spam or not. Regression Models, here the model is responsible to evaluate and understand the relationship between input variables. Here, it predicts numeric labels.
Unsupervised Model:
Here, the model is responsible for extracting relationships between the input variables. Outputs aren't provided in the training stage; it solely depends on the model in hand to find corelations in the data provided. This algorithm tries to organize that structure the data in the process of creating labels. They can be of two types, Clustering and Density Estimation. In clustering, model segments the data based on similarities into multiple groups or clusters. Density Estimation includes submarining the distribution of the data, in the process reducing the dimensions so as to determine the exact information required or to see the underlying data pattern.
Reinforcement Model: This uses regiment training process on the machine learning model, providing a set of actions, parameters and end values. It learns from the patterns and previous results and begins to adapt its course with respect to a situation in order to achieve best possible results. [1,5]

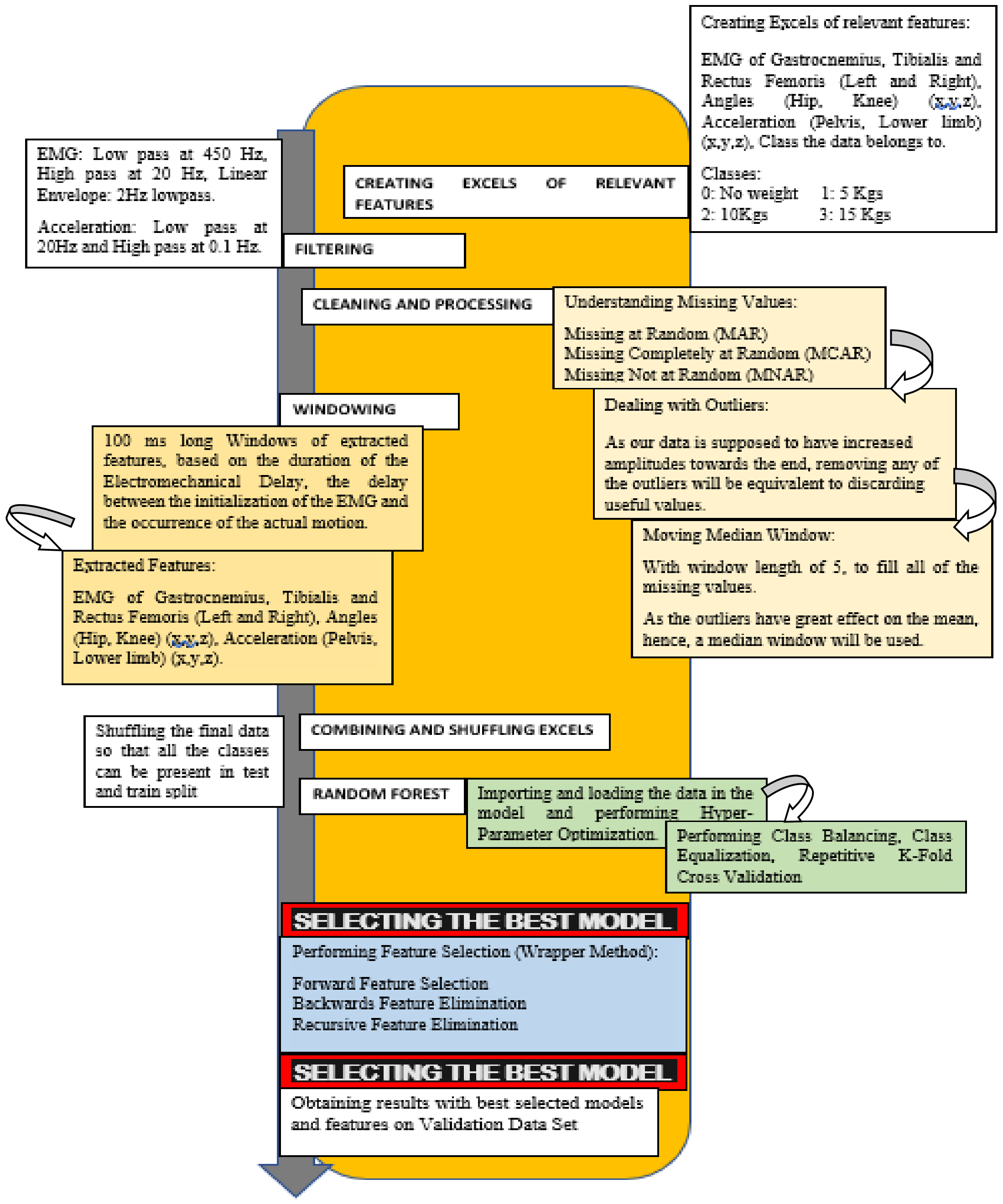
5.1.2. Classification and its types:

Classification models are fed with input and the target variables These models use input training data to predict the probability of it to fall under certain class. It does so by understanding patterns connecting the input variables to the classes in the training phase and when they obtain previously unseen data in testing phase, these patterns are used to predict the classes to which these unseen variables would belong.

Types: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbour and Naïve Bayes.
Logistic Regression: It works as binomial classification. It consists of a sigmoid function that generates probability of the output label and compares it to the defined threshold, e.g., spam or not.
Decision Tree: Here, branches are built in hierarchical approach act with if-else logic, the dataset is partitioned into multiple subsets and final decision is executed at the leaves.
Random Forest: It is the collection of decision trees. Here, the results are aggregated from multiple predictors, it's the ensemble technique. Bagging technique is utilized here as well, allowing every tree to be trained on the random sampled original dataset.
Support Vector Machine (SVM): It classifies the data points based on the positioning and distances from the border between different classes, called as hyper-plane which maximizes the distances between the classes.
K-Nearest Neighbour (KNN): This model represents each data point in n dimensional space which is defined as n features. It calculates the distances between surrounding points and then assigns class labels based on least of the distances.
Naïve Bayes: This uses the prior knowledge to calculate conditional probabilities assuming each of the feature are independent of each other. [6]

Referring to the research papers, 'Sensor Type, Axis, and Position-Based Fusion and Feature Selection for Multimodal Human Daily Activity Recognition in Wearable Body Sensor Networks', authored by, Abeer A. Badawi, Ahmad Al-Kabbany, and Heba A. Shaban, and HuGaDB: Human Gait Database for Activity Recognition from Wearable Inertial Sensor Networks, authored by, Roman Chereshnev and Attila Kert´esz-Farkas, where Human Gait Database (HuGaDb) was used, consisting of the data from the gyroscopes, the accelerometers and the sEMG sensors was used, to classify activities such as, 'Sitting, Walking, Running, Walking, Stairs, Bicycling and Elevator ride', with extracted features being mean, variance, skewness, kurtosis, RMS, etc, using machine learning algorithms such as Random Forest, Decision Tree, Support Vector Machine and K-Nearest Neighbours. Observed from the results, Random Forest algorithm proved to be the most accurate with respect to the data in hand. [7,8]

5.2: Methodology



Creating Excels of relevant features:
EMG of Gastrocnemius, Tibialis and Rectus Femoris (Left and Right), Angles (Hip, Knee) (x,y,z), Acceleration (Pelvis, Lower limb) (x,y,z), Class the data belongs to.
Classes: 0: No weight 1: 5 Kgs 2: 10Kgs 3: 15 Kgs
EMG: Low pass at 450 Hz, High pass at 20 Hz, Linear Envelope: 2Hz lowpass.
Acceleration: Low pass at 20Hz and High pass at 0.1 Hz.
CREATING EXCELS OF RELEVANT FEATURES
FILTERING
CLEANING AND PROCESSING
Understanding Missing Values:
Missing at Random (MAR)
Missing Completely at Random (MCAR)
Missing Not at Random (MNAR)
WINDOWING
Dealing with Outliers:
As our data is supposed to have increased amplitudes towards the end, removing any of the outliers will be equivalent to discarding useful values.
100 ms long Windows of extracted features, based on the duration of the Electromechanical Delay, the delay between the initialization of the EMG and the occurrence of the actual motion.
Moving Median Window:
With window length of 5, to fill all of the missing values.
As the outliers have great effect on the mean, hence, a median window will be used.
Extracted Features:
EMG of Gastrocnemius, Tibialis and Rectus Femoris (Left and Right), Angles (Hip, Knee) (x,y,z), Acceleration (Pelvis, Lower limb) (x,y,z).
Shuffling the final data so that all the classes can be present in test and train split
COMBINING AND SHUFFLING EXCELS
RANDOM FOREST
Importing and loading the data in the model and performing Hyper-Parameter Optimization.
Performing Class Balancing, Class Equalization, Repetitive K-Fold Cross Validation
SELECTING THE BEST MODEL
Performing Feature Selection (Wrapper Method):
Forward Feature Selection
Backwards Feature Elimination
Recursive Feature Elimination
SELECTING THE BEST MODEL
Obtaining results with best selected models and features on Validation Data Set

Figure 5.1: Process Methodology

5.2.1. Creating excels of relevant features:

EMG of Gastrocnemius, Tibialis and Rectus Femoris (Left and Right), Angles (Hip, Knee) (x,y,z), Acceleration (Pelvis, Lower limb) (x,y,z), Class the data belongs to from 5 Subjects with 4 trials per class.

Classes:

0-------------No Weight

1-------------5kgs

2-------------10kgs

3-------------15kgs

5.2.2. Filtering:

EMG: Low pass at 450 Hz, High pass at 20 Hz, Linear Envelope: 2Hz lowpass.

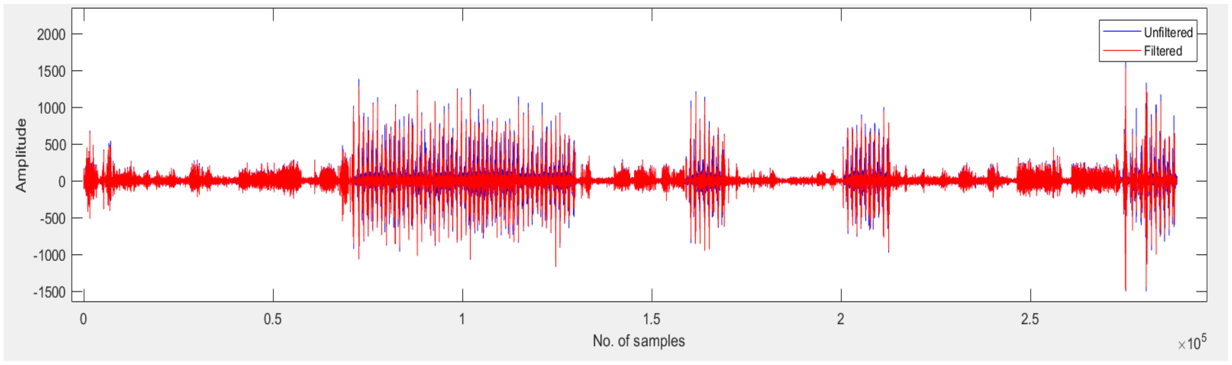


Figure 5.2: EMG Signal

Acceleration: Low pass at 20Hz and High pass at 0.1 Hz.

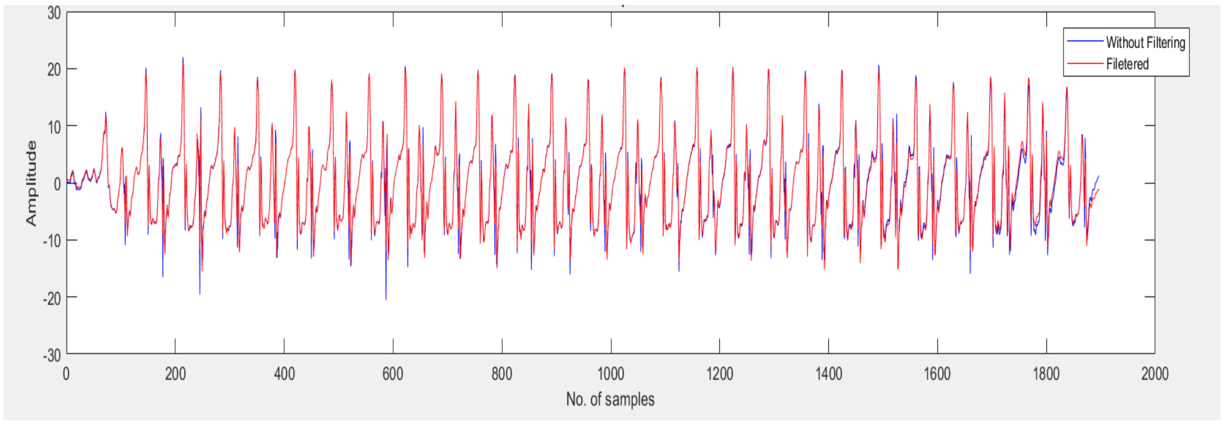


Figure 5.3: Acceleration Signal

5.2.3. Cleaning and Processing:

5.2.3.1. Understanding Missing Values:

Missing values can be of three types, Missing at Random, Missing Completely at Random and Missing not at Random.

a) Missing at Random (MAR): Values are missing because of some other variables, e.g.: in BP data, values can be missing because of the age, not a lot of young people get their BP checked as compared to the older ones.
b) Missing Completely at Random (MCAR): Missing data is unrelated.
c) Missing Not at Random (MNAR): Missing data is related to the variable itself, e.g.: Some people don't want to reveal their age or salary in the surveys.

With respect to the data in hand, the only reason the values can be missing is because of incomplete downloading, while conducting the trials. So, it can be classified as MAR (Missing at Random).

5.2.3.2. Dealing with Outliers:

EMG:

Understanding the nature of outliers: As our data is supposed to have increased amplitudes towards the end, removing any of the outliers will be equivalent to discarding useful values.

Outliers have great effect on mean and standard deviation.

Outliers have no effect on median and inter-quartile range

Histogram:

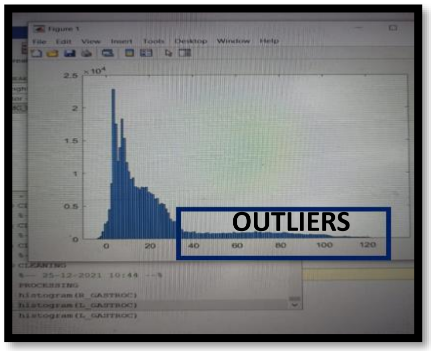
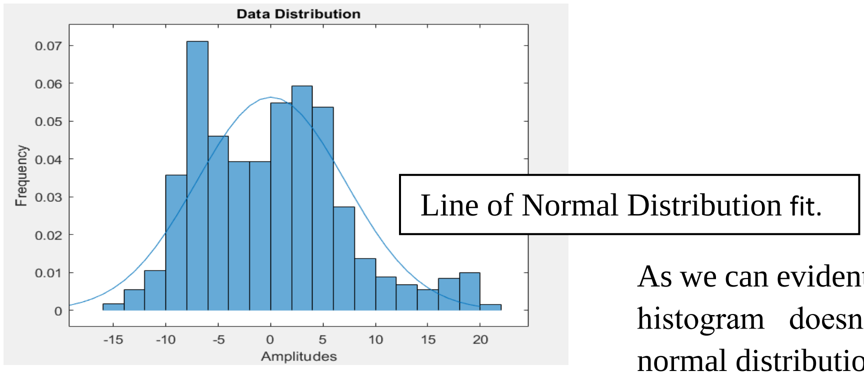


Figure 5.4: Histogram

ACCELERATION:

It represents the distribution of data, identifies presence of outliers, skewness in the data, modality of the data (unimodal, bimodal, multimodal), etc. Mode is the measure of the central tendency, representing the most frequently occurring values in the dataset.

Histogram:



As we can evidently see, the histogram doesn’t fit the normal distribution

Figure 5.5: Normal Distribution Fit

In the plot above, it can be clearly spotted that it is a bimodal plot.

QQ (Quantile-Quantile) Plot:

It is the plot of two data quantiles set against each other. It is used to understand if two datasets come from similar data distribution background. It is a graphical method of analysing the probability distributions of two populations by plotting their respective quantiles against each other. Quantile is defined as the percentage or the part of points lying below a certain value, e.g.: 40% quantile is the point at which 40% data will fall below and 60% will fall above that mentioned value.

Here, the quantiles of standard normal distribution data are set against the quantiles of data in hand to understand the deviations between the two.

Using QQ plot many aspects of data distribution can be analysed simultaneously, such as, outlier detection, shift in scale, change in symmetry, etc.

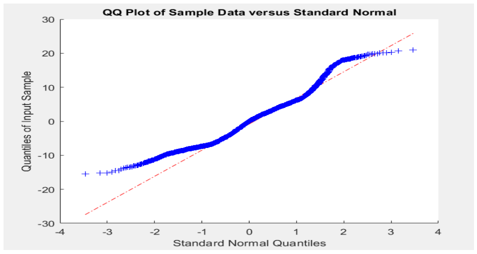


Figure 5.6: Distribution is positively skewed.

Skewness tells us about the direction of outliers. Here, the outliers are on the positive side.
There are methods to transform skewed data to normal distribution: Power Transformation, Log Transformation and Exponential Transformation.
With the data in consideration, the outliers would only be produced by a person's faulty GAIT at a point or a mis-step, not removing them will help the model take into account the human error.

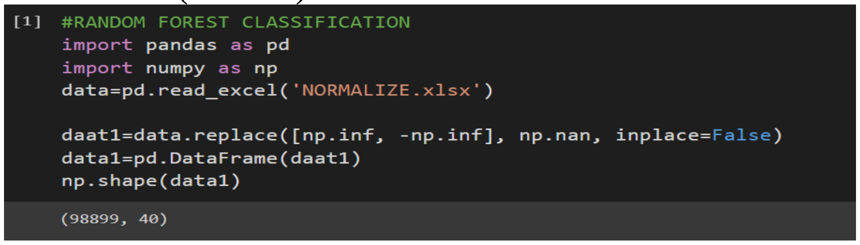
5.2.3.3. Moving Median Window:
Outliers have a great effect on the Mean; hence, the Moving Median Window is be used for filling of missing values., with the window length of 5.

5.2.4. Windowing and Feature Engineering:
Window Length: 100ms: It is the time required to differentiate between activities. It is based on the duration of EMD (Electro Motive Delay), the delay between the initialization of EMG and the occurrence of actual motion.
Extracted Features: Mean & Standard Deviation of EMG of Gastrocnemius, Tibialis and Rectus Femoris (Left and Right), Angles (Hip, Knee) (x,y,z), Acceleration (Pelvis, Lower limb) (x,y,z).

5.2.5. Combining and Shuffling Excels:
Combining datasets of individual classes (Walking without weight, Walking with 5 Kgs, Walking with 10 Kgs and Walking with 15 Kgs) and shuffling to make sure of equal division in Train and Test split.
Training Dataset: 5 subjects

```
(98899, 40)
```

98899: Rows
40: Columns (Features)



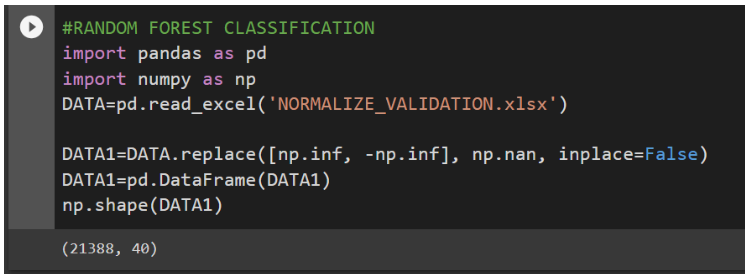
```
[1] #RANDOM FOREST CLASSIFICATION
    import pandas as pd
    import numpy as np
    data=pd.read_excel('NORMALIZE.xlsx')

    daat1=data.replace([np.inf, -np.inf], np.nan, inplace=False)
    data1=pd.DataFrame(daat1)
    np.shape(data1)

    (98899, 40)
```

Figure 5.7: Dataset Split

Validation Dataset: 1 Subject Data

```
(21388, 40)
```

21388: Rows
40: Columns (Features)



```
#RANDOM FOREST CLASSIFICATION
import pandas as pd
import numpy as np
DATA=pd.read_excel('NORMALIZE_VALIDATION.xlsx')

DATA1=DATA.replace([np.inf, -np.inf], np.nan, inplace=False)
DATA1=pd.DataFrame(DATA1)
np.shape(DATA1)
```

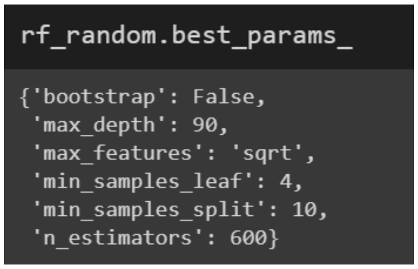
```
(21388, 40)
```

Figure 5.8: Dataset Split

5.2.6.1. Hyper Parameter Optimization:

Hyperparameters are configurations that cannot be learnt from the regular data that we provide to the algorithm; these are inbuilt to the algorithm and each algorithm has its own predefined set of hyperparameters. Hyperparameters are often tuned for increasing model accuracy.

Selected Hyper-Parameters:



```
rf_random.best_params_
```

```
{'bootstrap': False,
 'max_depth': 90,
 'max_features': 'sqrt',
 'min_samples_leaf': 4,
 'min_samples_split': 10,
 'n_estimators': 600}
```

Figure 5.9: Hyper-Parameter Optimization

Bootstrap: A method of replacing and statistically resampling the original dataset in turn creating multiple training datasets.

Max_depth: It controls the growth height of the trees in the random forest. Increasing this parameter plays a significant role in increasing the accuracy of the prediction algorithm, until overfitting is induced.

Max_features: Random Forest Algorithm chooses a subset of data at random and runs the prediction for best the suitable features.

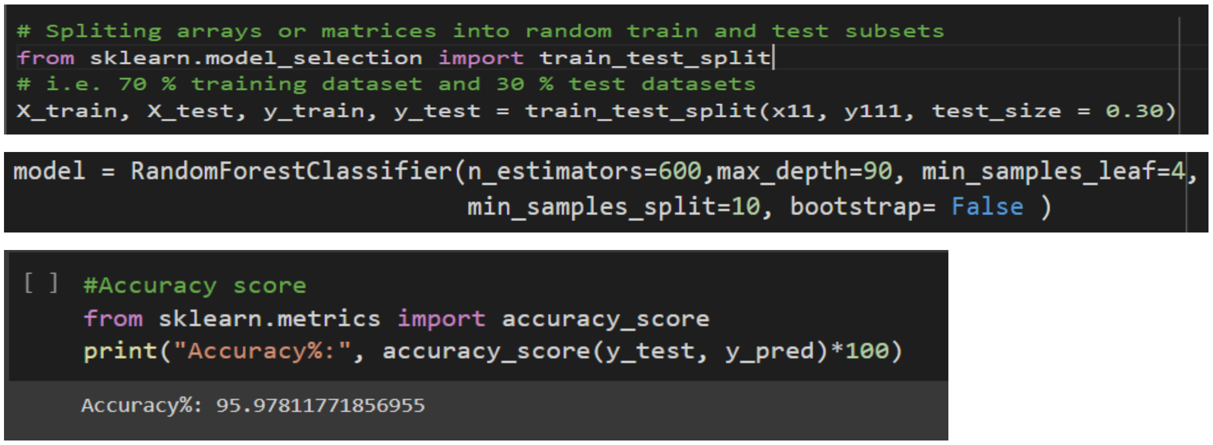
Min_samples_leaf: It identifies the least number of samples post-split that a node shall hold, to avoid over-fitting.

Min_samples_split: It identifies the least number of samples on an internal node for it to split further. If this value happens to be extremely low, then it will eventually lead to over-fitting of the model. Increasing this value, we can decrease the value of total number of splits, limiting the model parameter, hence reducing the induced over-fitting. On an average it is set between 2 to 6.

n_estimators: This identifies the suitable number of decision tress in the random forest model classifier.

5.2.6.2. Understanding Divisions between the Classes and Repeated K-Fold Cross Validation: Class Equalization and Class Balancing:

Training Data:

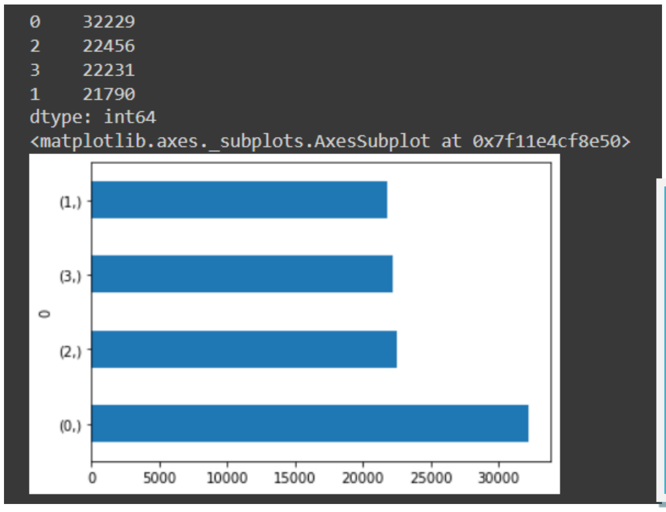


```
# Spliting arrays or matrices into random train and test subsets
from sklearn.model_selection import train_test_split
# i.e. 70 % training dataset and 30 % test datasets
X_train, X_test, y_train, y_test = train_test_split(x11, y111, test_size = 0.30)
```

```
model = RandomForestClassifier(n_estimators=600,max_depth=90, min_samples_leaf=4,
                               min_samples_split=10, bootstrap= False )
```

```
[ ] #Accuracy score
    from sklearn.metrics import accuracy_score
    print("Accuracy%:", accuracy_score(y_test, y_pred)*100)

    Accuracy%: 95.97811771856955
```

Figure 5.10: Hyper-Parameterized Model



```
0    32229
2    22456
3    22231
1    21790
dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f11e4cf8e50>
```

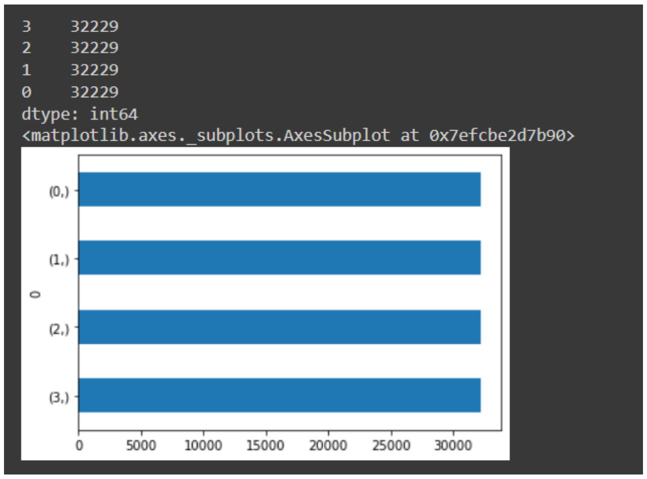
Classes and rows representing them.

Classes	Rows
0	32229
1	21790
2	22456
3	22231

Accuracy obtained: 95.97%, on training data after implementing hyper-optimized parameters.

Figure 5.11: Class Distribution

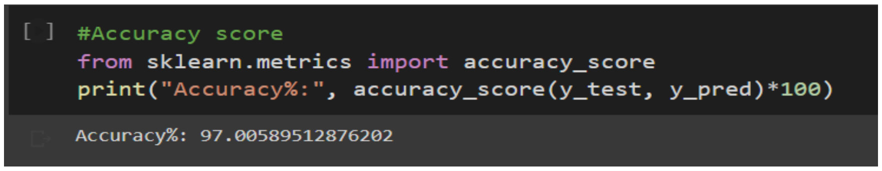
a. <u>Class Equalization</u>: Implementing Class Equalization technique.



```
3    32229
2    32229
1    32229
0    32229
dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7efcbe2d7b90>
```

Figure 5.12: Class Equalization

Equalizing the number of rows bellowing to each class to 32229, which is equivalent to the highest number of rows belonging to a particular class (Class 0), it is a form of over sampling using SMOTE algorithm. It helps removes the bias; the algorithm isn't more trained with respect to few classes.



```
#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy%:", accuracy_score(y_test, y_pred)*100)

Accuracy%: 97.00589512876202
```

Figure 5.13: Hyper-Parameterized and Class Equalized Model

Accuracy obtained implementing hyper-parameter optimized model with class equalization technique is 97.005%.

K-Fold Repetitive Cross Validation:

Cross – Validation: Training the model with the subset of data and then evaluating it using the complimentary subset of data. It's the method of dividing the data.

Types: LOOCV and K-Fold Cross Validation.

LOOCV: Leave one out cross validation: Leaves only one datapoint for testing and iterates over each data point.

K-Fold Cross Validation: We split the dataset into different folds (K) and perform training on all except one (K-1). We iterate K times with different subsets. This might give rise to random sampling, that is, division of classes between subsets is irregular.

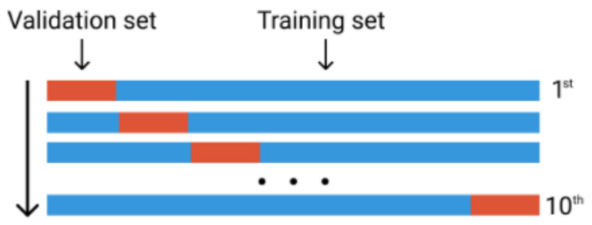


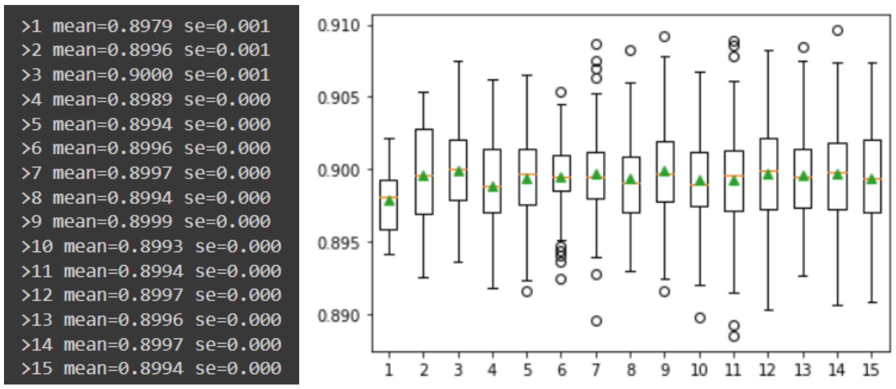
Figure 5.14: K-Fold Repetitive Cross Validation

Usually selected folds (K): 10 (NON-OVERLAPPING)

Stratified sampling in K-Fold Cross Validation makes sure the training dataset contains all the classes in equal proportions, as its present in the original dataset.

<u>Repeated Stratified K Fold Cross Validation</u>: It repeats the cross-validation procedure 'n' number of times and we utilize the mean of accuracy.

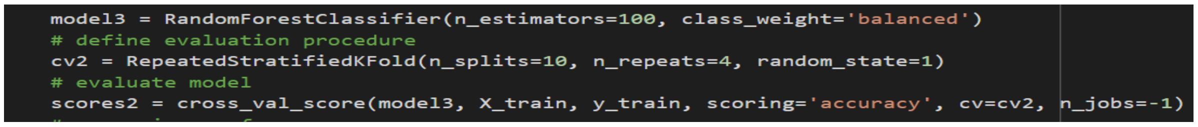
Confirming the number of repeats: Appropriate number of repeats for Repeated Stratified K Fold Cross Validation is given by the highest mean of accuracy and lowest standard error.



```
>1 mean=0.8979 se=0.001
>2 mean=0.8996 se=0.001
>3 mean=0.9000 se=0.001
>4 mean=0.8989 se=0.000
>5 mean=0.8994 se=0.000
>6 mean=0.8996 se=0.000
>7 mean=0.8997 se=0.000
>8 mean=0.8994 se=0.000
>9 mean=0.8999 se=0.000
>10 mean=0.8993 se=0.000
>11 mean=0.8994 se=0.000
>12 mean=0.8997 se=0.000
>13 mean=0.8996 se=0.000
>14 mean=0.8997 se=0.000
>15 mean=0.8994 se=0.000
```

Figure 5.15: Hyper-Parameter Optimization of Repeated Stratified K Fold Cross Validation

Maximum Number of Useful Repeats = 3



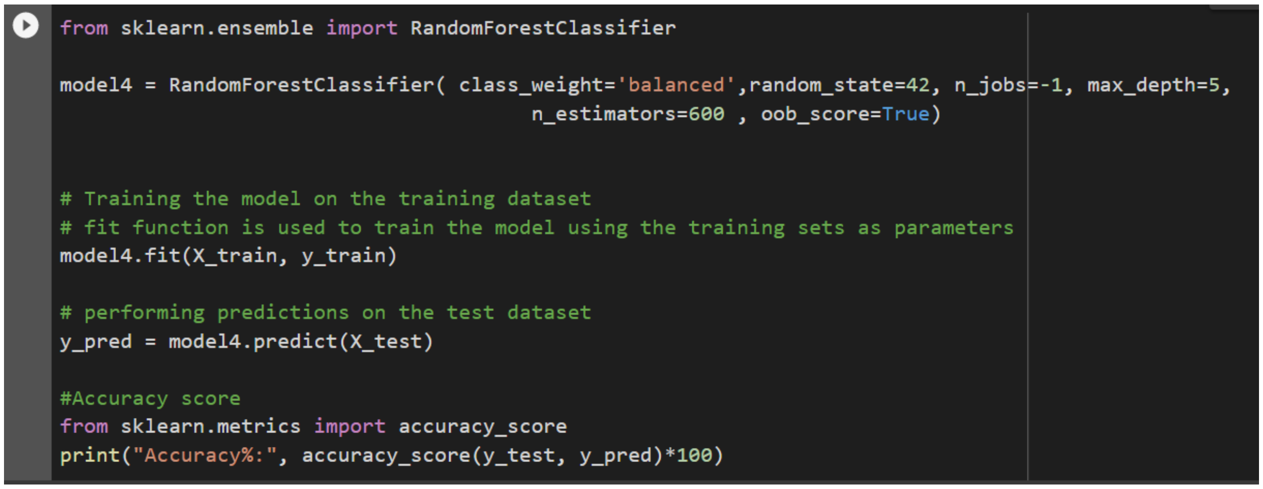
```
    model3 = RandomForestClassifier(n_estimators=100, class_weight='balanced')
    # define evaluation procedure
    cv2 = RepeatedStratifiedKFold(n_splits=10, n_repeats=4, random_state=1)
    # evaluate model
    scores2 = cross_val_score(model3, X_train, y_train, scoring='accuracy', cv=cv2, n_jobs=-1)
```

```
Accuracy: 91.97
```

Accuracy obtained implementing hyper-parameter optimized model with class equalization along with CV technique is 91.97%.

b. <u>Class Balancing</u>: Implementing Class Balancing technique.

Using weight balancing technique, higher weightage is given to the features with lower value counts and lower weightage to the features with higher value counts, hence, creating equalization. Weights can be changed manually or by using the option to simply balance it automatically.

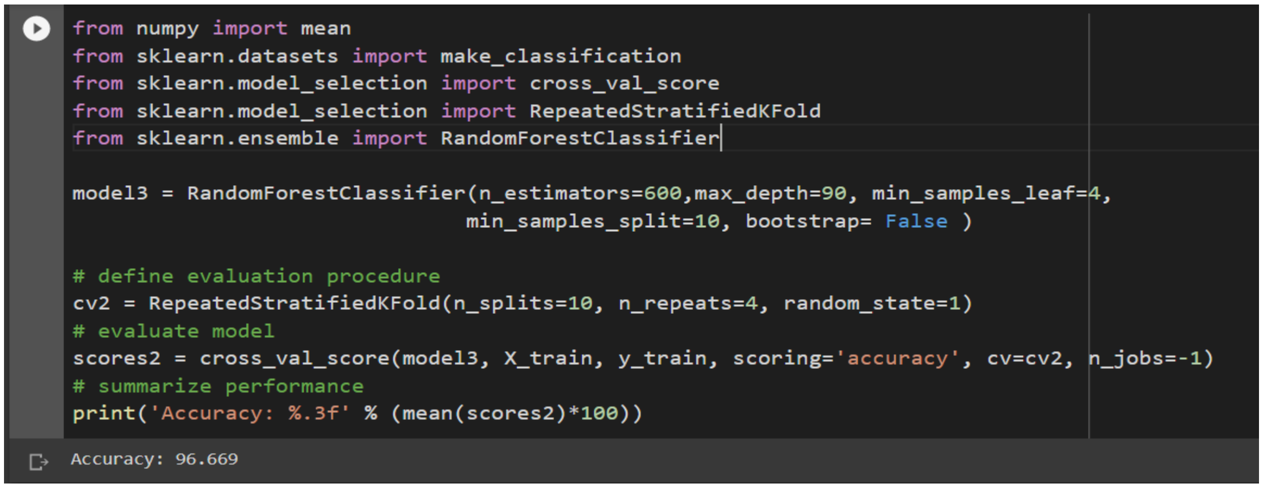


```
from sklearn.ensemble import RandomForestClassifier

model4 = RandomForestClassifier( class_weight='balanced',random_state=42, n_jobs=-1, max_depth=5,
                                       n_estimators=600 , oob_score=True)


# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
model4.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = model4.predict(X_test)

#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy%:", accuracy_score(y_test, y_pred)*100)
```

Figure 5.16: Hyper-Parameter Optimized and Class Balanced Model

Accuracy obtained implementing hyper-parameter optimized model with class balancing technique is 68.18%.

K-Fold Repetitive Cross Validation:



```
from numpy import mean
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.ensemble import RandomForestClassifier

model3 = RandomForestClassifier(n_estimators=600,max_depth=90, min_samples_leaf=4,
                                min_samples_split=10, bootstrap= False )

# define evaluation procedure
cv2 = RepeatedStratifiedKFold(n_splits=10, n_repeats=4, random_state=1)
# evaluate model
scores2 = cross_val_score(model3, X_train, y_train, scoring='accuracy', cv=cv2, n_jobs=-1)
# summarize performance
print('Accuracy: %.3f' % (mean(scores2)*100))
```

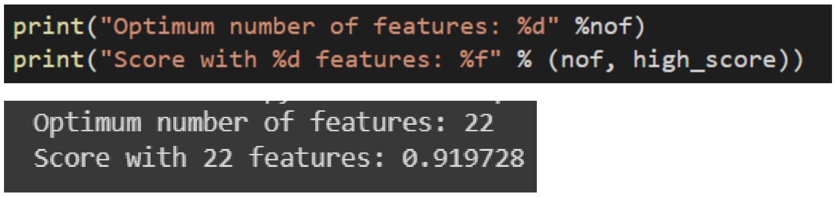
```
Accuracy: 96.669
```

Figure 5.17: Hyper-Parameter Optimized and K-Fold Repetitive Cross Validation Model

Accuracy obtained implementing hyper-parameter optimized model with class balancing along with CV technique is 96.66%.

THE BEST MODEL OBTAINED: IMPLEMENTING HYPER-PARAMETER OPTIMIZED MODEL WITH CLASS EQUALIZATION TECHNIQUE GIVING AN ACCURACY OF 97.005%.

5.2.6.3. Feature Selection (Wrapper Method): Forward Feature Selection, Backward Elimination Recursive Feature Elimination.



```
print("Optimum number of features: %d" %nof)
print("Score with %d features: %f" % (nof, high_score))
```

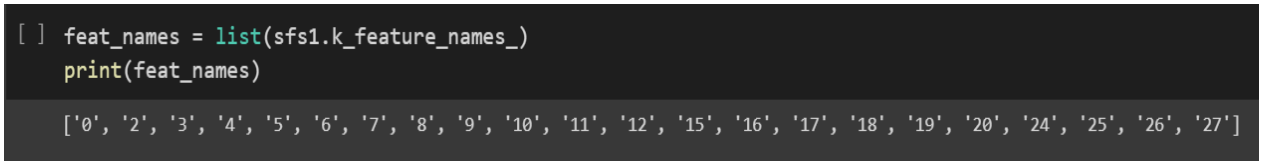
```
Optimum number of features: 22
Score with 22 features: 0.919728
```

Figure 5.18: Optimum Number of Features

a. Forward Feature Selection:

Here, features are selected one by one based on p value and significance level. Features are selected into a null model based on the minimum p value and highest significance level.

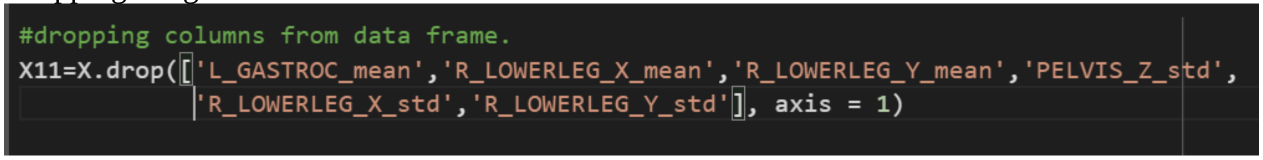
Significant Features:



```
[ ] feat_names = list(sfs1.k_feature_names_)
    print(feat_names)

    ['0', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '15', '16', '17', '18', '19', '20', '24', '25', '26', '27']
```

Figure 5.19: List of Significant Features from Forward Feature Selection

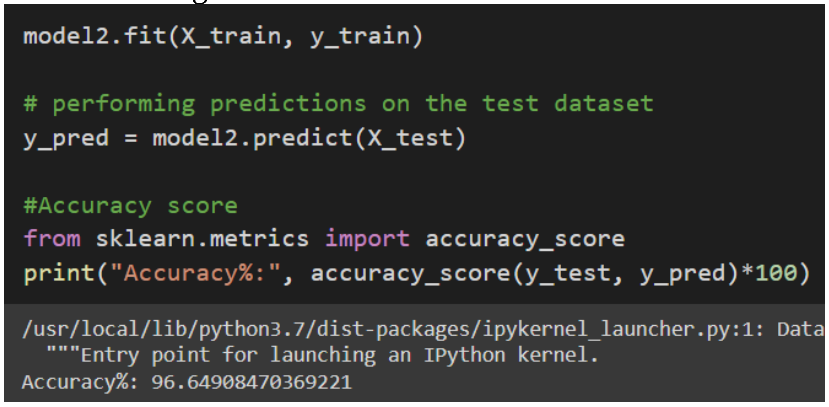
Dropping Insignificant Features:



```
#dropping columns from data frame.
X11=X.drop(['L_GASTROC_mean','R_LOWERLEG_X_mean','R_LOWERLEG_Y_mean','PELVIS_Z_std',
           'R_LOWERLEG_X_std','R_LOWERLEG_Y_std'], axis = 1)
```

Figure 5.20: Dropping Insignificant Features

Model Building:



```
model2.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = model2.predict(X_test)

#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy%:", accuracy_score(y_test, y_pred)*100)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: Data
  """Entry point for launching an IPython kernel.
Accuracy%: 96.64908470369221
```

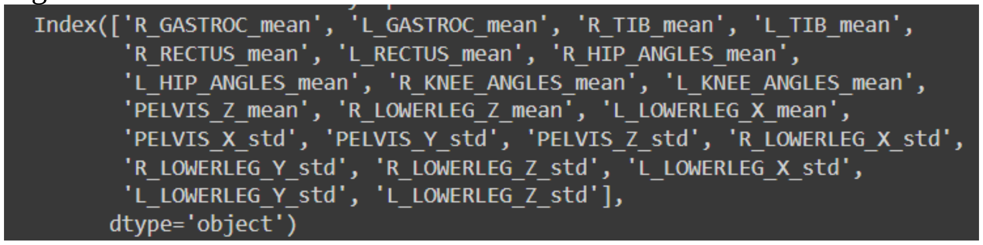
Figure 5.21: Model Building with hyper-parameter optimized, class equalization technique and Forward Feature Selection Technique

Accuracy obtained implementing hyper-parameter optimized model with class equalization technique and Forward Feature Selection Technique is 96.64%.

b. Recursive Feature Elimination:

It's basically a backward selection of the predictors. This technique begins by building a model on the entire set of predictors and computing an importance score for each predictor. The least important predictor(s) are then removed, the model is re-built, and important scores are computed again. In practice, the analyst specifies the number of predictor subsets to evaluate as well as each subset's size.

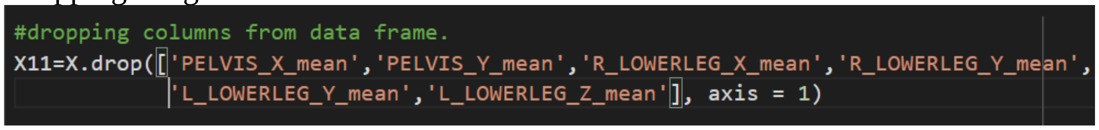
Significant Features:



```
Index(['R_GASTROC_mean', 'L_GASTROC_mean', 'R_TIB_mean', 'L_TIB_mean',
       'R_RECTUS_mean', 'L_RECTUS_mean', 'R_HIP_ANGLES_mean',
       'L_HIP_ANGLES_mean', 'R_KNEE_ANGLES_mean', 'L_KNEE_ANGLES_mean',
       'PELVIS_Z_mean', 'R_LOWERLEG_Z_mean', 'L_LOWERLEG_X_mean',
       'PELVIS_X_std', 'PELVIS_Y_std', 'PELVIS_Z_std', 'R_LOWERLEG_X_std',
       'R_LOWERLEG_Y_std', 'R_LOWERLEG_Z_std', 'L_LOWERLEG_X_std',
       'L_LOWERLEG_Y_std', 'L_LOWERLEG_Z_std'],
      dtype='object')
```

Figure 5.22: List of Significant Features from Recursive Feature Elimination

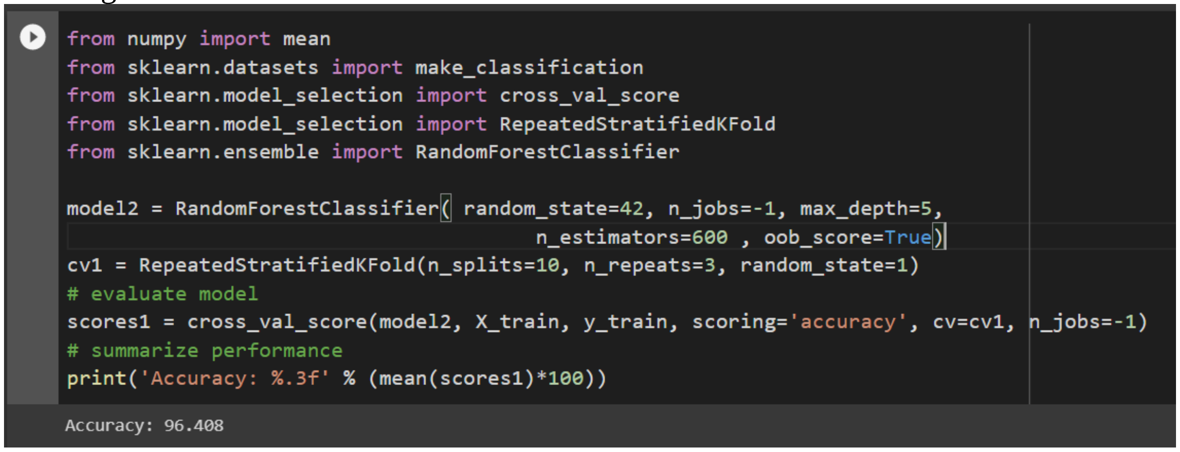
Dropping Insignificant Features:



```
#dropping columns from data frame.
X11=X.drop(['PELVIS_X_mean','PELVIS_Y_mean','R_LOWERLEG_X_mean','R_LOWERLEG_Y_mean',
            'L_LOWERLEG_Y_mean','L_LOWERLEG_Z_mean'], axis = 1)
```

Figure 5.23: Dropping Insignificant Features

Building Model:



```
from numpy import mean
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.ensemble import RandomForestClassifier

model2 = RandomForestClassifier( random_state=42, n_jobs=-1, max_depth=5,
                                          n_estimators=600 , oob_score=True)
cv1 = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate model
scores1 = cross_val_score(model2, X_train, y_train, scoring='accuracy', cv=cv1, n_jobs=-1)
# summarize performance
print('Accuracy: %.3f' % (mean(scores1)*100))

Accuracy: 96.408
```

Figure 5.24: Model Building with hyper-parameter optimized, class equalization technique and Recursive Feature Elimination Technique

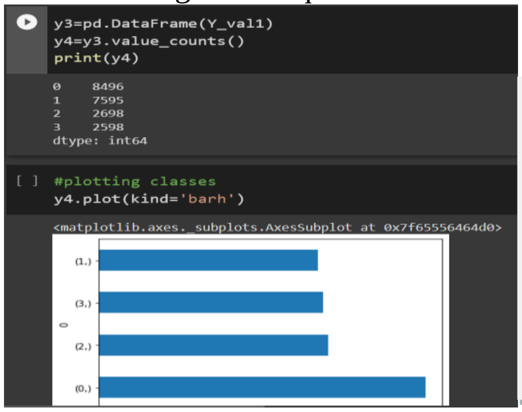
Accuracy obtained implementing hyper-parameter optimized model with class equalization technique and Recursive Feature Elimination Technique is 96.40%.

THE BEST MODEL OBTAINED: IMPLEMENTING HYPER-PARAMETER OPTIMIZED MODEL WITH CLASS EQUALIZATION TECHNIQUE AND FORWARD FEATURE SELECTION GIVING AN ACCURACY OF 97.005%.

5.2.7. Best model parameters are implemented on Validation Dataset:

a. Performing Class Equalization on Validation Dataset: `(21388, 40)`

21388: Rows
40: Columns (Features)



```
y3=pd.DataFrame(Y_val1)
y4=y3.value_counts()
print(y4)

0    8496
1    7595
2    2698
3    2598
dtype: int64
```

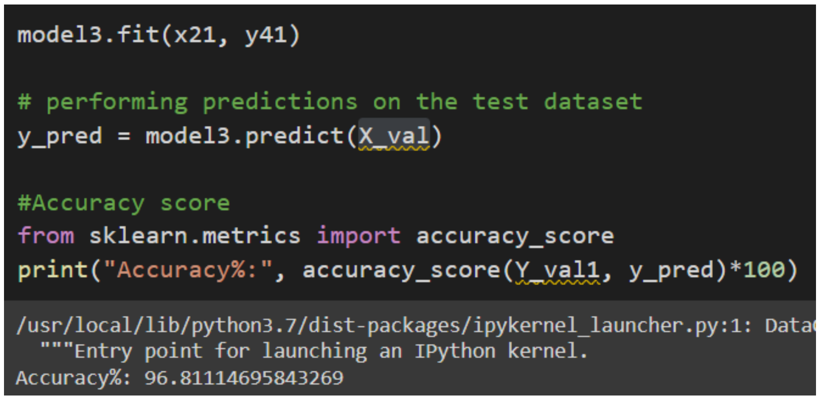
```
[ ] #plotting classes
    y4.plot(kind='barh')

<matplotlib.axes._subplots.AxesSubplot at 0x7f65556464d0>
```

Classes and rows representing them.

Classes	Rows
0	8496
1	7595
2	2698
3	2598

Figure 5.25: Class Distribution

Running the hyper-parameter optimised model with class equalized training and validation datasets with CV after selection of most significant features using Forward Feature Selection.



```
model3.fit(x21, y41)

# performing predictions on the test dataset
y_pred = model3.predict(X_val)

#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy%:", accuracy_score(Y_val1, y_pred)*100)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: Data
  """Entry point for launching an IPython kernel.
Accuracy%: 96.81114695843269
```

Figure 5.25: Model Building with hyper-parameter optimized model with class equalization technique and Forward Feature Selection Technique

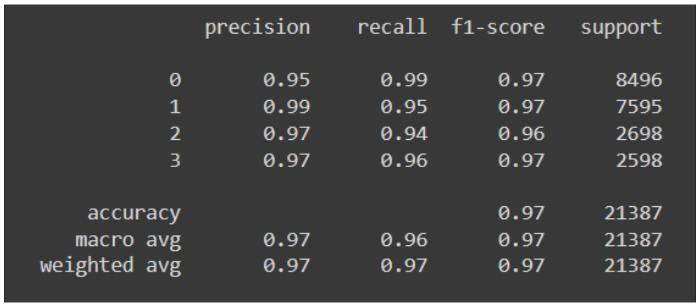
`Accuracy%:96.81%`

Accuracy obtained implementing hyper-parameter optimized model with class equalization technique and Forward Feature Selection Technique is 96.81%.

5.3: Results and Discussions:

5.3.1. Classification report:

A Classification report is used to measure the quality of predictions from a classification algorithm.



```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      8496
           1       0.99      0.95      0.97      7595
           2       0.97      0.94      0.96      2698
           3       0.97      0.96      0.97      2598

    accuracy                           0.97     21387
   macro avg       0.97      0.96      0.97     21387
weighted avg       0.97      0.97      0.97     21387
```

Figure 5.26: Classification Report

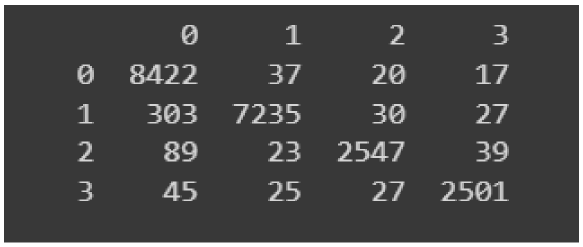
a. Precision: It is the ability of the classifier to provide correct class labels, it is the ratio of true positives and the sum of true positives and false positives.
b. Recall: It is the ability of a classifier to find all the positive instances.
c. F1 – score: 2*(Recall * Precision) / (Recall + Precision)

   It is the weighted mean of precision and recall, making sure the ideal value is 1.0 and worst value is 0.0. Usually, F1 score is used to compare the models over the accuracy score.
d. Support: Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing.
e. Accuracy: TP+TN/TP+FP+FN+TN

   It is the most basic performance measure of a model; it is the ratio of predicted and total observations.
f. Macro_Avg: The macro-averaged F1 score (or macro F1 score) is computed by taking the mean per-class of the F1 scores. It aims at treating all the classes equally irrespective of their support scores.
g. Weighted_Avg: The weighted-averaged F1 score is calculated by taking the mean of all per class F1 scores while considering each class's support**.**

5.3.2. Confusion matrix:

It is used to evaluate the performance of a classifier. The idea is to count the number of times an instance of class A is classified as class B. Each row refers to an actual class and each column refers to a predicted class. We can calculate the precision and recall from the confusion matrix as well, which can be used to compare between models.



```
      0     1     2     3
0  8422    37    20    17
1   303  7235    30    27
2    89    23  2547    39
3    45    25    27  2501
```

Figure 5.27: Confusion Matrix

Precision: (TP) / (TP+FP)

Penalizes the false positives.

Recall: (TP) / (TP+FN)

Penalizes the false negatives.

For example:

Considering two models where apples are used, model A of a supermarket and model B of a cider company. In model A, the main aim is to sell off the apples and make profits, few bad apples wouldn't make any difference but few of the good apples going to waste will lead to loss, hence, here the model with the higher recall would be selected, as it would penalize the false negatives, as they can't have good apples classified as bad mistakenly. In case of model B, even few of the bad apples can spoil a whole of cider. Hence, here the model with higher precision would be chosen, as it penalizes the false positives, here, they can't have bad apples to be mistakenly be classified as good.

True Positive: The actual value was True, and the model predicted as True
True Negative: The actual value was False, and the model predicted False.
False Positive: The actual value was True, and the model predicted False. This is also known as a Type II error.
False Negative: The actual value was False, and the model predicted True. This is also known as a Type I error.

Class 0: Walking with No Weight, Total: 8496

TP:8422

TN:12454

FP:74

FN:134

Percentage Accuracy of Prediction: (TP_0/ $TOTAL_0$) * 100 = 99.1%

Percentage Misclassification: [($TOTAL_0$ – TP_0)/ $TOTAL_0$] * 100 = 0.87%

Class 1: Walking with 5Kgs of Weight, Total: 7595

TP:7235

TN:5114

FP:57

FN:134

Percentage Accuracy of Prediction: $(TP_0 / TOTAL_0) * 100 =$ 95.2%

Percentage Misclassification: $[(TOTAL_0 – TP_0) / TOTAL_0] * 100 =$ 4.7%

Class 2: Walking with 10Kgs of Weight, Total: 2698

TP: 2698

TN: 2501

FP: 39

FN: 27

Percentage Accuracy of Prediction: $(TP_0 / TOTAL_0) * 100 =$ 100%

Percentage Misclassification: $[(TOTAL_0 – TP_0) / TOTAL_0] * 100 =$ 0 %

Class 3: Walking with 15Kgs of Weight, Total: 2598

TP: 2501

Percentage Accuracy of Prediction: $(TP_0 / TOTAL_0) * 100 =$ 96.2%

Percentage Misclassification: $[(TOTAL_0 – TP_0) / TOTAL_0] * 100 =$ 3.7%

From the results above ‘Walking with 10Kgs of Weight Class’, that is Class 2, has the highest prediction accuracy.

5.4: Conclusion:

This chapter is a step towards recognition of daily human activities, proposing an approach based on sensor fusion and feature selection. After rigorously performing multiple simulations and referring to the work of multiple researchers, the authors have come up with the proposed methodology. As concluded in previous chapters, about a 100 ms delay is present between human action and detection of culminative sEMG signal, exploiting this delay as a time window of extracted features, the authors have tried to decode the intension of motion. The vision is to facilitate the development of a robust system to differentiate and identify between the intension of various activities. Empowering the exoskeleton with this ability, will smoothen the process of performing actions, hence, making the exoskeleton the master, assisting not resisting the human body.

5.5: References:

1. https://databricks.com/glossary/machine-learning-models
2. ht tps://monkeylearn.com/blog/classification-algorithms/
3. https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html#:~:text=There%20are%20four%20types%20of,%2Dsupervised%2C%20unsupervised%20and%20reinforcement.
4. https://machinelearningmastery.com/types-of-learning-in-machine-learning/
5. https://docs.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model
6. https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501

7. Badawi AA, Al-Kabbany A, Shaban HA. Sensor type, axis, and position-based fusion and feature selection for multimodal human daily activity recognition in wearable body sensor networks. Journal of Healthcare Engineering. 2020 Jun 7;2020.

8. Chereshnev R, Kertész-Farkas A. Hugadb: Human gait database for activity recognition from wearable inertial sensor networks. InInternational Conference on Analysis of Images, Social Networks and Texts 2017 Jul 27 (pp. 131-141). Springer, Cham.

Chapter 6

Classification of different activities using IMU Data

6.1: Introduction and Literature review:

6.1.1. Types of machine learning:

These models can learn patterns and use them further to take decisions and make predictions for unseen data. These models can be taught to recognize objects, for this, such models need to be trained and optimized. These models involve automating the repetitive decisions making and repetitive evaluations tasks, providing consistent results. The resulting function includes rules and data structures hence, creating a trained machine learning model.

Types of Machine Learning Model: Supervised Model, Unsupervised Model and Reinforcement Model.

Supervised Model:

Here, both, input and target variables are provided as input to the model and model learns to map patterns between the input and target variables. They can be of three types: Classification and Regression. Classification Models, the target of prediction is a class or category label, that input variable would fall under, e.g.: mail being spam or not. Regression Models, here the model is responsible to evaluate and understand the relationship between input variables. Here, it predicts numeric labels.

Unsupervised Model:

Here, the model is responsible for extracting relationships between the input variables. Outputs aren't provided in the training stage; it solely depends on the model in hand to find corelations in the data provided. This algorithm tries to organize that structure the data in the process of creating labels. They can be of two types, Clustering and Density Estimation. In clustering, model segments the data based on similarities into multiple groups or clusters. Density Estimation includes submarining the distribution of the data, in the process reducing the dimensions so as to determine the exact information required or to see the underlying data pattern.

Reinforcement Model: This uses regiment training process on the machine learning model, providing a set of actions, parameters and end values. It learns from the patterns and previous results and begins to adapt its course with respect to a situation in order to achieve best possible results. [1,5]

6.1.2. Classification and its types:

Classification models are fed with input and the target variables These models use input training data to predict the probability of it to fall under certain class. It does so by understanding patterns connecting the input variables to the classes in the training phase and when they obtain previously unseen data in testing phase, these patterns are used to predict the classes to which these unseen variables would belong.

Types: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbour and Naïve Bayes.

Logistic Regression: It works as binomial classification. It consists of a sigmoid function that generates probability of the output label and compares it to the defined threshold, e.g., spam or not.

Decision Tree: Here, branches are built in hierarchical approach act with if-else logic, the dataset is partitioned into multiple subsets and final decision is executed at the leaves.

Random Forest: It is the collection of decision trees. Here, the results are aggregated from multiple predictors, it's the ensemble technique. Bagging technique is utilized here as well, allowing every tree to be trained on the random sampled original dataset.

Support Vector Machine (SVM): It classifies the data points based on the positioning and distances from the border between different classes, called as hyper-plane which maximizes the distances between the classes.

K-Nearest Neighbour (KNN): This model represents each data point in n dimensional space which is defined as n features. It calculates the distances between surrounding points and then assigns class labels based on least of the distances.

Naïve Bayes: This uses the prior knowledge to calculate conditional probabilities assuming each of the feature are independent of each other. [6]

Referring to the research papers, 'Sensor Type, Axis, and Position-Based Fusion and Feature Selection for Multimodal Human Daily Activity Recognition in Wearable Body Sensor Networks', authored by, Abeer A. Badawi, Ahmad Al-Kabbany, and Heba A. Shaban, and HuGaDB: Human Gait Database for Activity Recognition from Wearable Inertial Sensor Networks, authored by, Roman Chereshnev and Attila Kert´esz-Farkas, where Human Gait Database (HuGaDb) was used, consisting of the data from the gyroscopes, the accelerometers and the sEMG sensors was used, to classify activities such as, 'Sitting, Walking, Running, Walking, Stairs, Bicycling and Elevator ride', with extracted features being mean, variance, skewness, kurtosis, RMS, etc, using machine learning algorithms such as Random Forest, Decision Tree, Support Vector Machine and K-Nearest Neighbours. Observed from the results, Random Forest algorithm proved to be the most accurate with respect to the data in hand. [7,8]

6.2: Methodology:

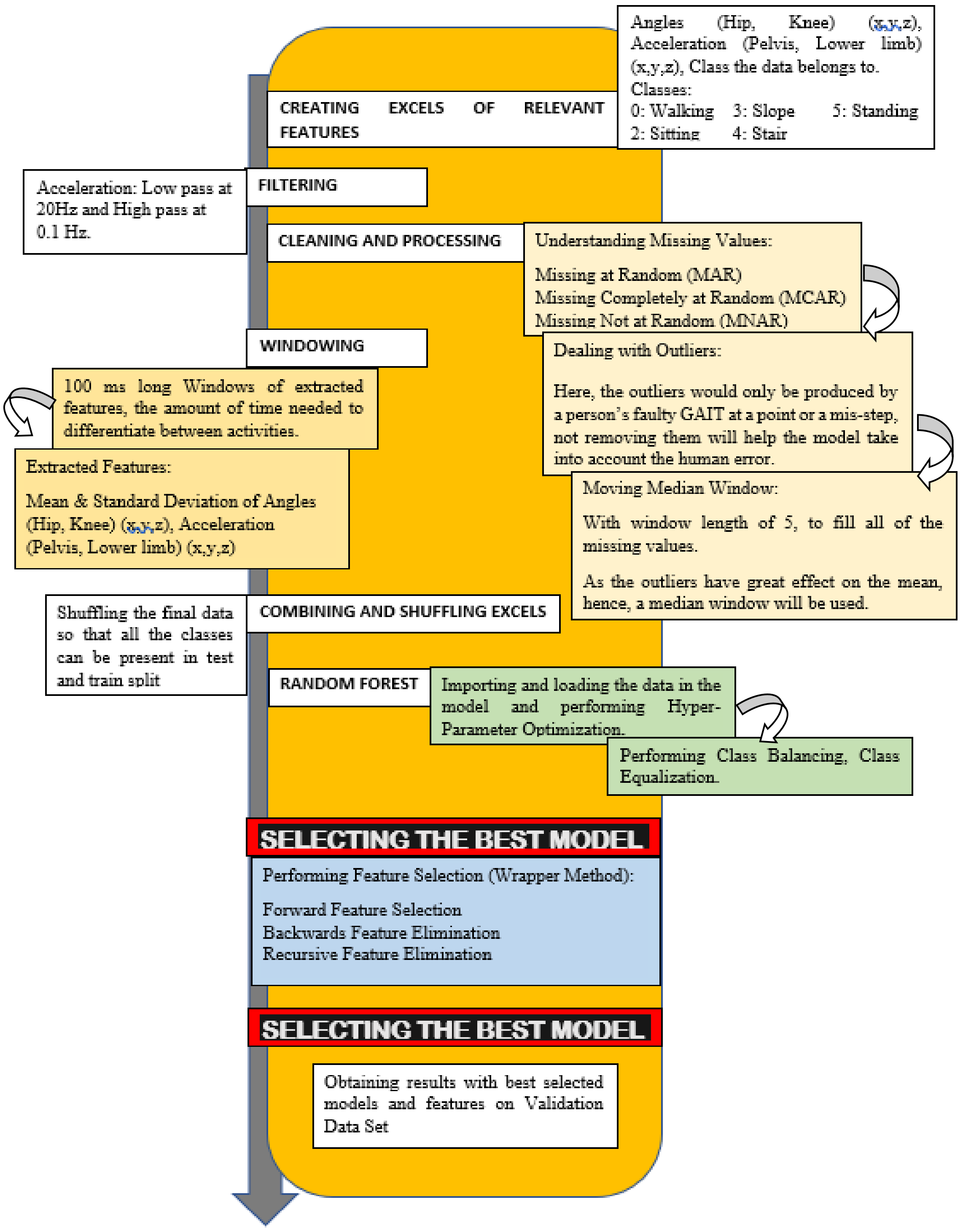
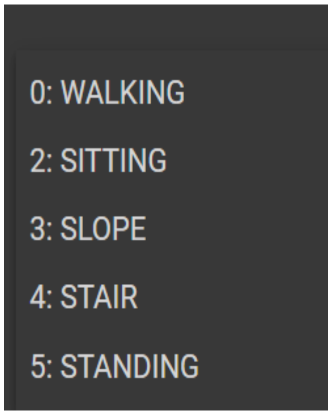

CREATING EXCELS OF RELEVANT FEATURES
Angles (Hip, Knee) (x,y,z), Acceleration (Pelvis, Lower limb) (x,y,z), Class the data belongs to.
Classes:
0: Walking 3: Slope 5: Standing
2: Sitting 4: Stair
FILTERING
Acceleration: Low pass at 20Hz and High pass at 0.1 Hz.
CLEANING AND PROCESSING
Understanding Missing Values:
Missing at Random (MAR)
Missing Completely at Random (MCAR)
Missing Not at Random (MNAR)
Dealing with Outliers:
Here, the outliers would only be produced by a person's faulty GAIT at a point or a mis-step, not removing them will help the model take into account the human error.
Moving Median Window:
With window length of 5, to fill all of the missing values.
As the outliers have great effect on the mean, hence, a median window will be used.
WINDOWING
100 ms long Windows of extracted features, the amount of time needed to differentiate between activities.
Extracted Features:
Mean & Standard Deviation of Angles (Hip, Knee) (x,y,z), Acceleration (Pelvis, Lower limb) (x,y,z)
COMBINING AND SHUFFLING EXCELS
Shuffling the final data so that all the classes can be present in test and train split
RANDOM FOREST
Importing and loading the data in the model and performing Hyper-Parameter Optimization.
Performing Class Balancing, Class Equalization.
SELECTING THE BEST MODEL
Performing Feature Selection (Wrapper Method):
Forward Feature Selection
Backwards Feature Elimination
Recursive Feature Elimination
SELECTING THE BEST MODEL
Obtaining results with best selected models and features on Validation Data Set

Figure 6.1: Procedure Methodology

6.2.1. Creating excels of relevant features:

Angles (Hip, Knee) (x,y,z), Acceleration (Pelvis, Lower limb) (x,y,z), Class the data belongs to from 30 Subjects.

Classes:



0: WALKING

2: SITTING

3: SLOPE

4: STAIR

5: STANDING

Figure 6.2: Classes

6.2.2. Filtering:

Acceleration: Low pass at 20Hz and High pass at 0.1 Hz.

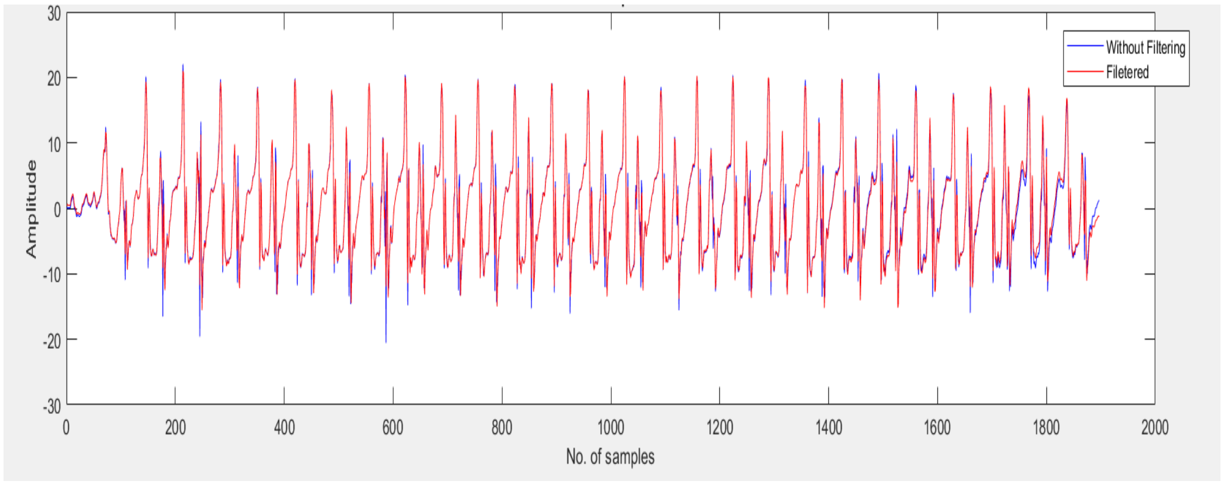


Figure 6.3: Acceleration Signal

6.2.3. Cleaning and Processing:

6.2.3.1. Understanding Missing Values:

Missing values can be of three types, Missing at Random, Missing Completely at Random and Missing not at Random.

a) Missing at Random (MAR): Values are missing because of some other variables, e.g.: in BP data, values can be missing because of the age, not a lot of young people get their BP checked as compared to the older ones.

b) Missing Completely at Random (MCAR): Missing data is unrelated.

c) Missing Not at Random (MNAR): Missing data is related to the variable itself, e.g.: Some people don’t want to reveal their age or salary in the surveys.

With respect to the data in hand, the only reason the values can be missing is because of incomplete downloading, while conducting the trials. So, it can be classified as MAR (Missing at Random).

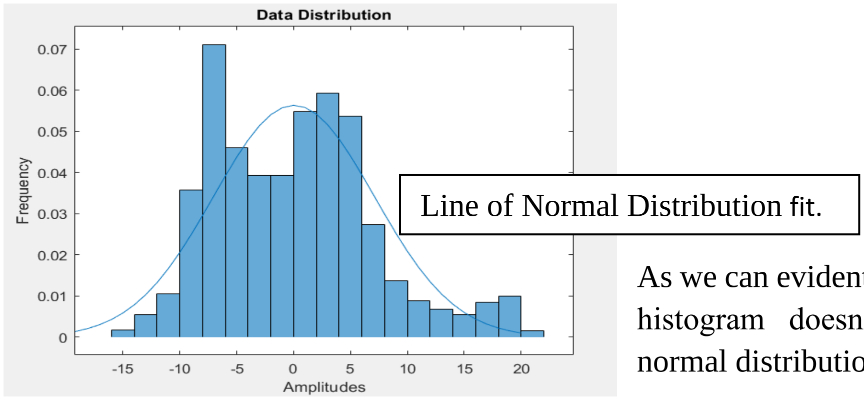
6.2.3.2. Dealing with Outliers:

To check for the outliers, distribution of data has to be analysed.

Types of Data Distributions: Discrete and Continuous. Discrete distribution can be Binomial, Poisson or Geometric. Continuous distribution can be Normal or Student T- distribution.

Histogram:

It represents the distribution of data, identifies presence of outliers, skewness in the data, modality of the data (unimodal, bimodal, multimodal), etc. Mode is the measure of the central tendency, representing the most frequently occurring values in the dataset



As we can evidently see, the histogram doesn’t fit the normal distribution

Figure 6.4: Normal Distribution Fit

In the plot above, it can be clearly spotted that it is a bimodal plot.

QQ (Quantile-Quantile) Plot:

It is the plot of two data quantiles set against each other. It is used to understand if two datasets come from similar data distribution background. It is a graphical method of analysing the probability distributions of two populations by plotting their respective quantiles against each other. Quantile is defined as the percentage or the part of points lying below a certain value, e.g.: 40% quantile is the point at which 40% data will fall below and 60% will fall above that mentioned value.

Here, the quantiles of standard normal distribution data are set against the quantiles of data in hand to understand the deviations between the two.

Using QQ plot many aspects of data distribution can be analysed simultaneously, such as, outlier detection, shift in scale, change in symmetry, etc.

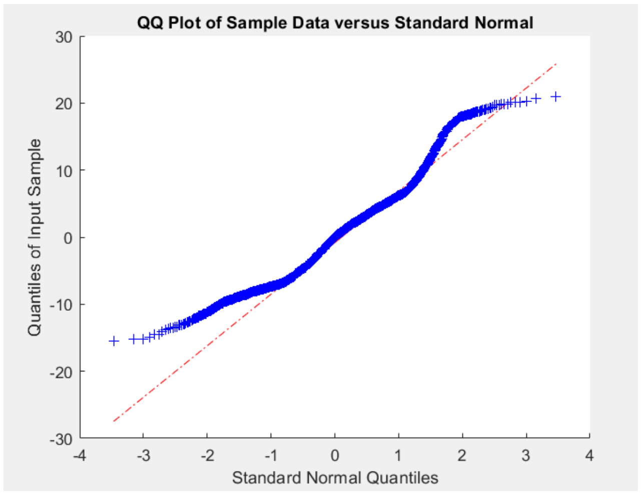


Figure 6.5: Distribution is positively skewed.

Skewness tells us about the direction of outliers. Here, the outliers are on the positive side.

There are methods to transform skewed data to normal distribution: Power Transformation, Log Transformation and Exponential Transformation.

<u>With the data in consideration, the outliers would only be produced by a person's faulty GAIT at a point or a mis-step, not removing them will help the model take into account the human error.</u>

6.2.3.3. Moving Median Window:

Outliers have a great effect on the Mean; hence, the Moving Median Window is be used for filling of missing values., with the window length of 5.

6.2.4. Windowing and Feature Engineering:

Window Length: 100ms: It is the time required to differentiate between activities.

Extracted Features: Mean & Standard Deviation of Angles (Hip, Knee) (x,y,z), Acceleration (Pelvis, Lower limb) (x,y,z).

6.2.5. Combining and Shuffling Excels:

Combining datasets of individual classes (Walking, Standing, Slope, Sitting and Stairs) and shuffling to make sure of equal division in Train and Test split.

<u>Training Dataset</u>: 30 subject data

```
(136640, 28)
```

136640: Rows
28: Columns (Features)

<u>Validation Dataset</u>: 10 Subject Data

```
(14997, 28)
```

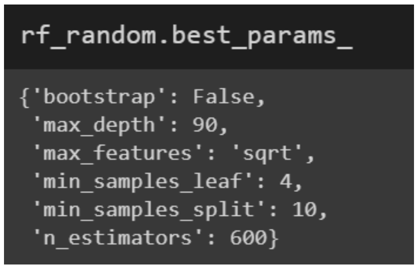
14997: Rows
28: Columns (Features)

6.2.6. Random Forest Algorithm:

6.2.6.1. Hyper Parameter Optimization:

Hyperparameters are configurations that cannot be learnt from the regular data that we provide to the algorithm; these are inbuilt to the algorithm and each algorithm has its own predefined set of hyperparameters. Hyperparameters are often tuned for increasing model accuracy.

Selected Hyper-Parameters:



```
rf_random.best_params_

{'bootstrap': False,
 'max_depth': 90,
 'max_features': 'sqrt',
 'min_samples_leaf': 4,
 'min_samples_split': 10,
 'n_estimators': 600}
```

Figure 6.6: Hyper-Parameter Optimization

Bootstrap: A method of replacing and statistically resampling the original dataset in turn creating multiple training datasets.

Max_depth: It controls the growth height of the trees in the random forest. Increasing this parameter plays a significant role in increasing the accuracy of the prediction algorithm, until overfitting is induced.

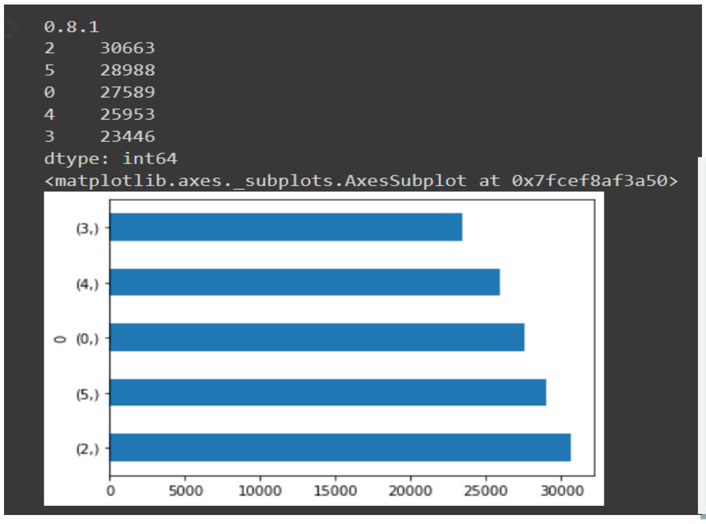
Max_features: Random Forest Algorithm chooses a subset of data at random and runs the prediction for best the suitable features.

Min_samples_leaf: It identifies the least number of samples post-split that a node shall hold, to avoid over-fitting.

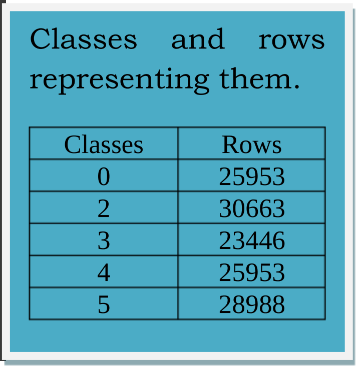
Min_samples_split: It identifies the least number of samples on an internal node for it to split further. If this value happens to be extremely low, then it will eventually lead to over-fitting of the model. Increasing this value, we can decrease the value of total number of splits, limiting the model parameter, hence reducing the induced over-fitting. On an average it is set between 2 to 6.

n_estimators: This identifies the suitable number of decision tress in the random forest model classifier.

5.2.6.2. Understanding Divisions between the Classes: Class Equalization and Class Balancing: Training Data:



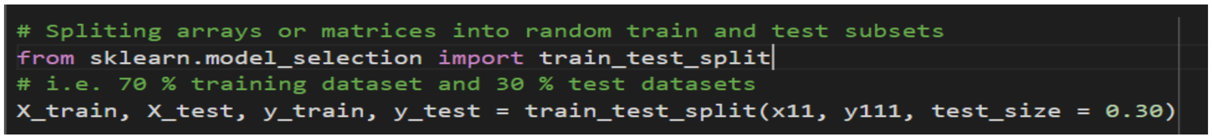
```
0.8.1
2     30663
5     28988
0     27589
4     25953
3     23446
dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7fcef8af3a50>
```



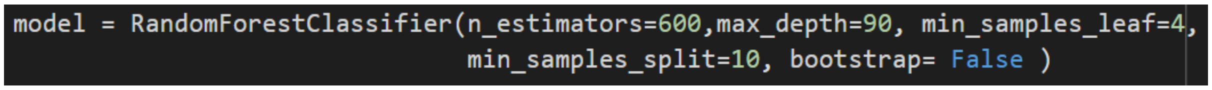
Classes and rows representing them.

Classes	Rows
0	25953
2	30663
3	23446
4	25953
5	28988

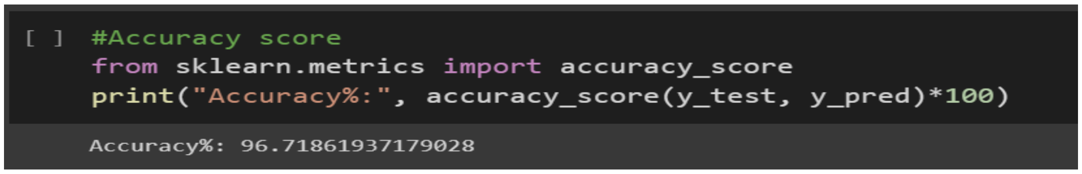
Figure 6.7: Class Distribution



```
# Spliting arrays or matrices into random train and test subsets
from sklearn.model_selection import train_test_split
# i.e. 70 % training dataset and 30 % test datasets
X_train, X_test, y_train, y_test = train_test_split(x11, y111, test_size = 0.30)
```



```
model = RandomForestClassifier(n_estimators=600,max_depth=90, min_samples_leaf=4,
                               min_samples_split=10, bootstrap= False )
```

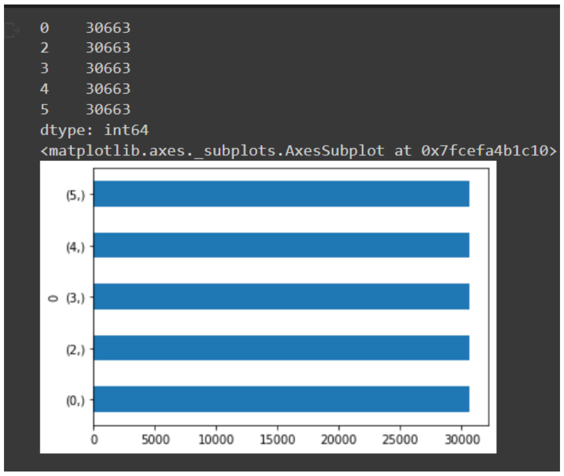


```
[ ]  #Accuracy score
     from sklearn.metrics import accuracy_score
     print("Accuracy%:", accuracy_score(y_test, y_pred)*100)

     Accuracy%: 96.71861937179028
```

Figure 6.8: Building Model with Hyper-Optimized Parameters

Accuracy obtained: 96.71%, on training data after implementing hyper-optimized parameters.

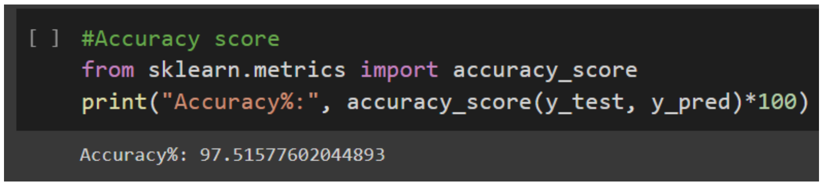
a. <u>Class Equalization</u>: Implementing Class Equalization technique.



```
0    30663
2    30663
3    30663
4    30663
5    30663
dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7fcefa4b1c10>
```

Figure 6.9: Class Equalization

Equalizing the number of rows bellowing to each class to 30,663, which is equivalent to the highest number of rows belonging to a particular class (Class 2), it is a form of over sampling using SMOTE algorithm. It helps removes the bias; the algorithm isn't more trained with respect to few classes.



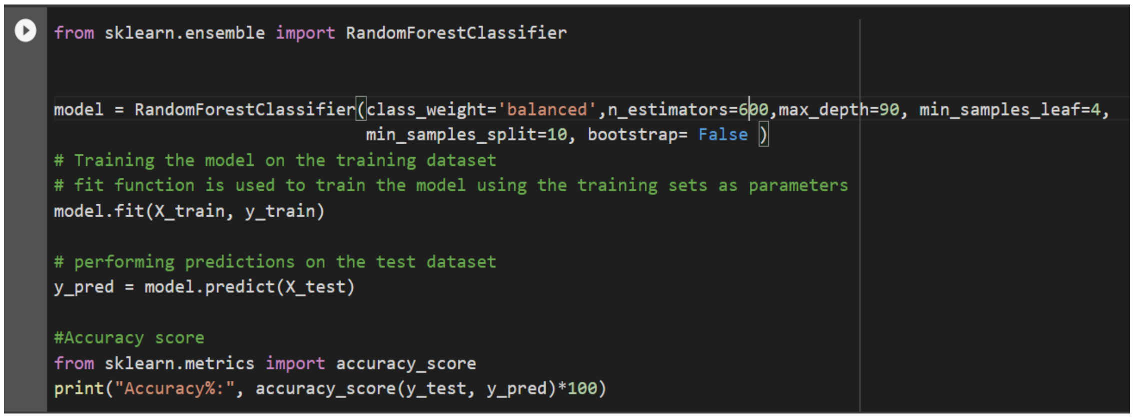
```
[ ]  #Accuracy score
     from sklearn.metrics import accuracy_score
     print("Accuracy%:", accuracy_score(y_test, y_pred)*100)

     Accuracy%: 97.51577602044893
```

Figure 6.10: Building Model with Hyper-Optimized Parameters and Class Equalization

Accuracy obtained implementing hyper-parameter optimized model with class equalization technique is 97.15%.

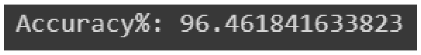
b. Class Balancing: Implementing Class Balancing technique.

Using weight balancing technique, higher weightage is given to the features with lower value counts and lower weightage to the features with higher value counts, hence, creating equalization. Weights can be changed manually or by using the option to simply balance it automatically.



```
from sklearn.ensemble import RandomForestClassifier


model = RandomForestClassifier(class_weight='balanced',n_estimators=600,max_depth=90, min_samples_leaf=4,
                               min_samples_split=10, bootstrap= False )
# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
model.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = model.predict(X_test)

#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy%:", accuracy_score(y_test, y_pred)*100)
```

Figure 5.11: Building Model with Hyper-Optimized Parameters and Class Balancing

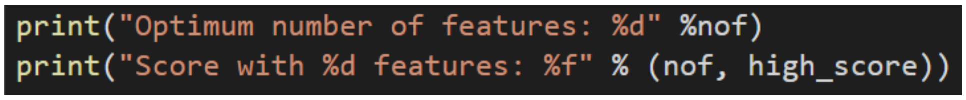


```
Accuracy%: 96.461841633823
```

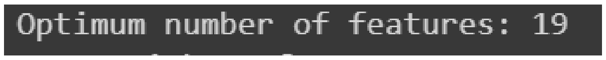
Accuracy obtained implementing hyper-parameter optimized model with class balancing technique is 96.46%.

BEST MODEL: HYPER PARAMETERIZED MODEL + CLASS EQUALIZATION MODEL: Accuracy: 97.15%

6.2.6.3. Feature Selection (Wrapper Method): Forward Feature Selection, Recursive Feature Elimination.



```
print("Optimum number of features: %d" %nof)
print("Score with %d features: %f" % (nof, high_score))
```

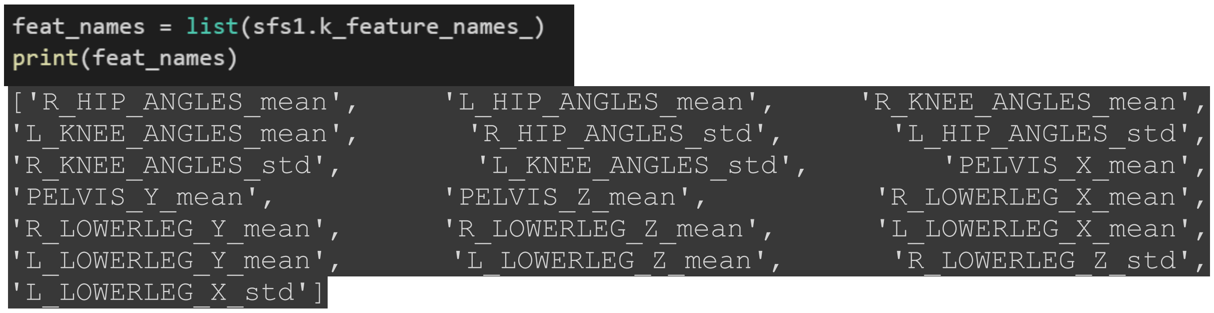


```
Optimum number of features: 19
```

Figure 6.12: Optimum Number of Features

a. Forward Feature Selection:
   Here, features are selected one by one based on p value and significance level. Features are selected into a null model based on the minimum p value and highest significance level.

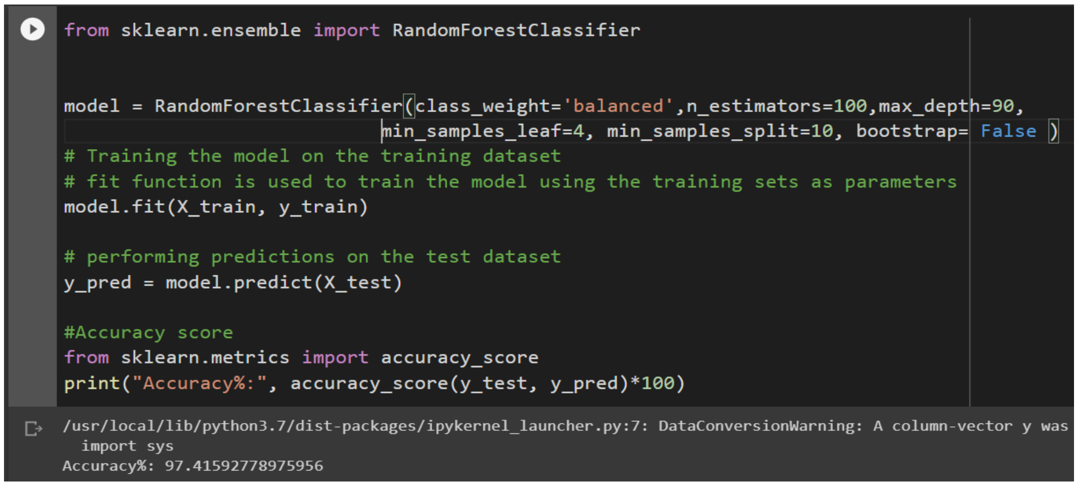
Significant Features:



```
feat_names = list(sfs1.k_feature_names_)
print(feat_names)
```

```
['R_HIP_ANGLES_mean',      'L_HIP_ANGLES_mean',      'R_KNEE_ANGLES_mean',
'L_KNEE_ANGLES_mean',       'R_HIP_ANGLES_std',       'L_HIP_ANGLES_std',
'R_KNEE_ANGLES_std',        'L_KNEE_ANGLES_std',        'PELVIS_X_mean',
'PELVIS_Y_mean',           'PELVIS_Z_mean',          'R_LOWERLEG_X_mean',
'R_LOWERLEG_Y_mean',       'R_LOWERLEG_Z_mean',      'L_LOWERLEG_X_mean',
'L_LOWERLEG_Y_mean',        'L_LOWERLEG_Z_mean',       'R_LOWERLEG_Z_std',
'L_LOWERLEG_X_std']
```

Figure 6.13: Significant Features from Forward Feature Selection



```
from sklearn.ensemble import RandomForestClassifier


model = RandomForestClassifier(class_weight='balanced',n_estimators=100,max_depth=90,
                               min_samples_leaf=4, min_samples_split=10, bootstrap= False )
# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
model.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = model.predict(X_test)

#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy%:", accuracy_score(y_test, y_pred)*100)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: DataConversionWarning: A column-vector y was
  import sys
Accuracy%: 97.41592778975956
```

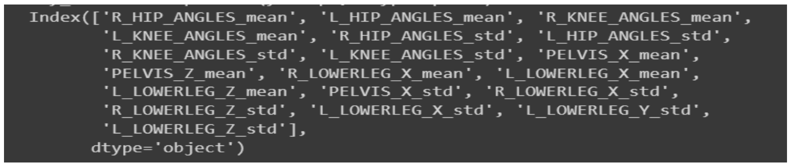
Figure 6.14: Building Model with Hyper-Optimized Parameters, Class Equalization and Forward Feature Selection Technique

Accuracy obtained implementing hyper-parameter optimized model with class equalization technique and Forward Feature Selection Technique is 97.4%.

b. Recursive Feature Elimination:

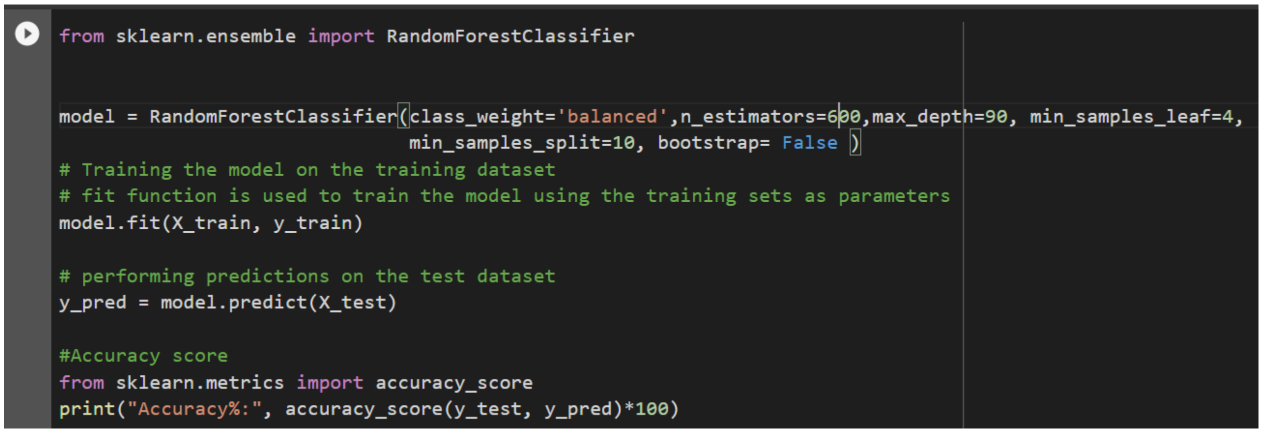
It's basically a backward selection of the predictors. This technique begins by building a model on the entire set of predictors and computing an importance score for each predictor. The least important predictor(s) are then removed, the model is re-built, and importance scores are computed again. In practice, the analyst specifies the number of predictor subsets to evaluate as well as each subset's size.

Significant Features:



```
Index(['R_HIP_ANGLES_mean', 'L_HIP_ANGLES_mean', 'R_KNEE_ANGLES_mean',
       'L_KNEE_ANGLES_mean', 'R_HIP_ANGLES_std', 'L_HIP_ANGLES_std',
       'R_KNEE_ANGLES_std', 'L_KNEE_ANGLES_std', 'PELVIS_X_mean',
       'PELVIS_Z_mean', 'R_LOWERLEG_X_mean', 'L_LOWERLEG_X_mean',
       'L_LOWERLEG_Z_mean', 'PELVIS_X_std', 'R_LOWERLEG_X_std',
       'R_LOWERLEG_Z_std', 'L_LOWERLEG_X_std', 'L_LOWERLEG_Y_std',
       'L_LOWERLEG_Z_std'],
      dtype='object')
```

Figure 6.15: Significant Features from Recursive Feature Elimination



```
from sklearn.ensemble import RandomForestClassifier


model = RandomForestClassifier(class_weight='balanced',n_estimators=600,max_depth=90, min_samples_leaf=4,
                               min_samples_split=10, bootstrap= False )
# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
model.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = model.predict(X_test)

#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy%:", accuracy_score(y_test, y_pred)*100)
```

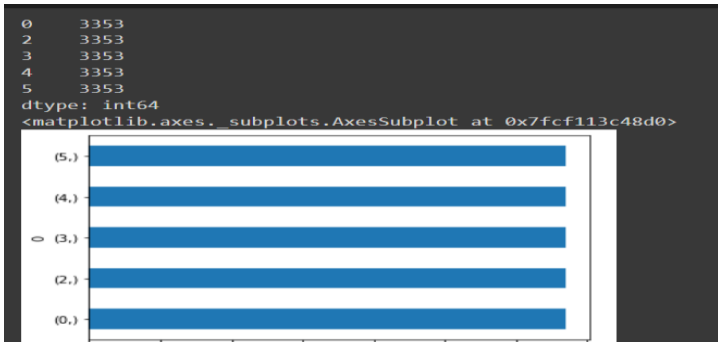
Figure 6.16 : Building Model with Hyper-Optimized Parameters, Class Equalization and Recursive Feature Elimination Technique

Accuracy obtained implementing hyper-parameter optimized model with class equalization technique and Recursive Feature Elimination Technique is 97.25%.

BEST MODEL: HYPER PARAMETERIZED MODEL+ CLASS EQUALIZATION MODEL + FORWARD FEATURE SELECTION TECHNIQUE: Accuracy: 97.4%

6.2.7. Best model parameters are implemented on Validation Dataset:

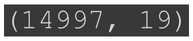
a. Performing Class Equalization on Validation Dataset: `(14997, 28)` 14997: Rows
28: Columns (Features)



```
0     3353
2     3353
3     3353
4     3353
5     3353
dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7fcf113c48d0>
```

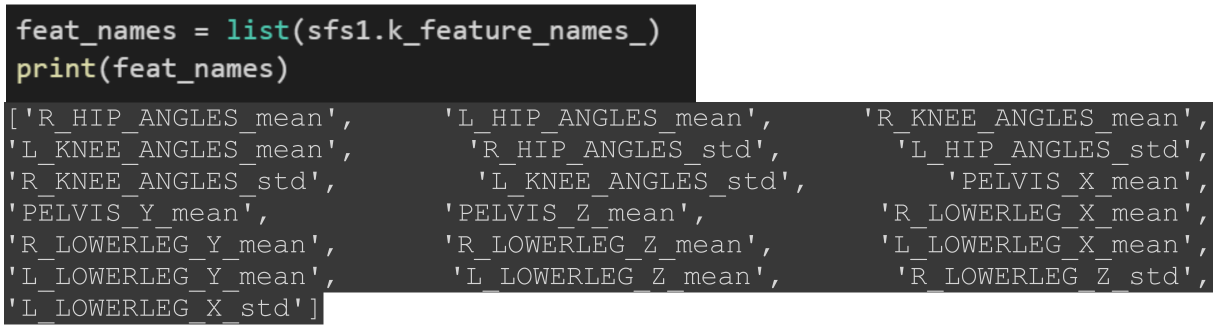
Figure 6.17 : Class equalization for Validation Dataset 1

b. Performing Forward Feature Selection:



```
(14997, 19)
```

14997: Rows
19: Selected Features

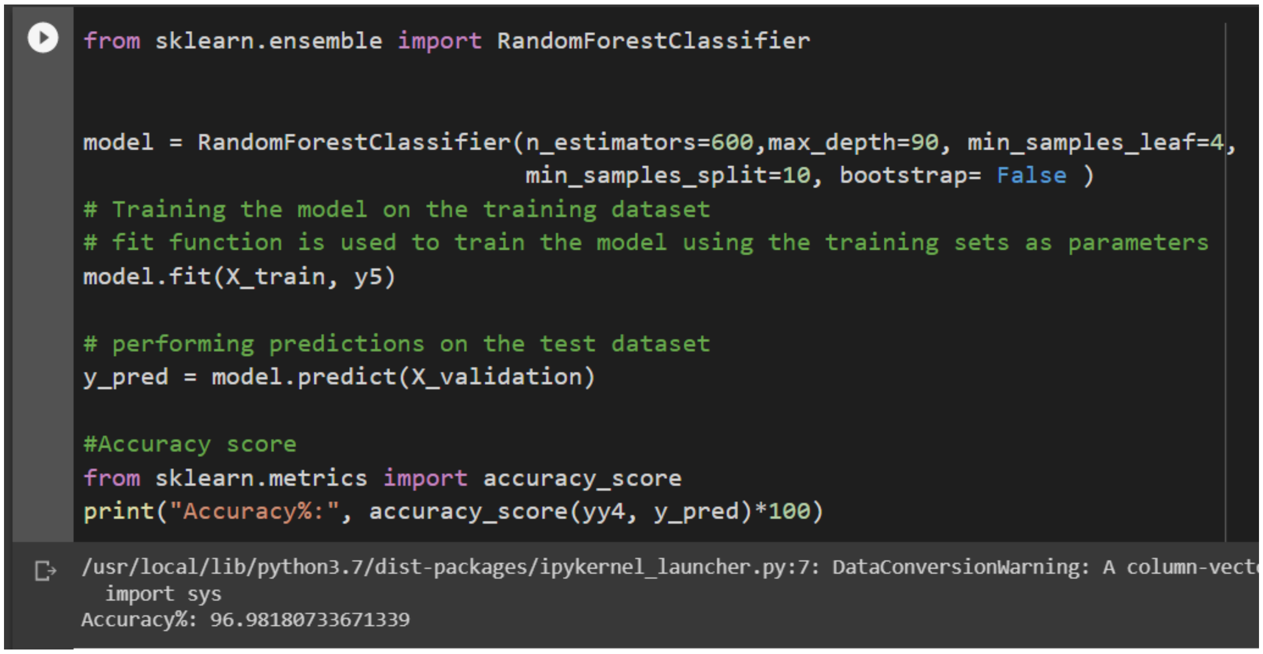


```
feat_names = list(sfs1.k_feature_names_)
print(feat_names)
```

```
['R_HIP_ANGLES_mean',      'L_HIP_ANGLES_mean',      'R_KNEE_ANGLES_mean',
'L_KNEE_ANGLES_mean',       'R_HIP_ANGLES_std',        'L_HIP_ANGLES_std',
'R_KNEE_ANGLES_std',        'L_KNEE_ANGLES_std',           'PELVIS_X_mean',
'PELVIS_Y_mean',           'PELVIS_Z_mean',          'R_LOWERLEG_X_mean',
'R_LOWERLEG_Y_mean',       'R_LOWERLEG_Z_mean',      'L_LOWERLEG_X_mean',
'L_LOWERLEG_Y_mean',        'L_LOWERLEG_Z_mean',       'R_LOWERLEG_Z_std',
'L_LOWERLEG_X_std']
```

Figure 6.18: Significant Features from Forward Feature Selection

Running the hyper-parameter optimised model with class equalized training and validation datasets after selection of most significant features using Forward Feature Selection.



```
from sklearn.ensemble import RandomForestClassifier


model = RandomForestClassifier(n_estimators=600,max_depth=90, min_samples_leaf=4,
                               min_samples_split=10, bootstrap= False )
# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
model.fit(X_train, y5)

# performing predictions on the test dataset
y_pred = model.predict(X_validation)

#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy%:", accuracy_score(yy4, y_pred)*100)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: DataConversionWarning: A column-vect
  import sys
Accuracy%: 96.98180733671339
```

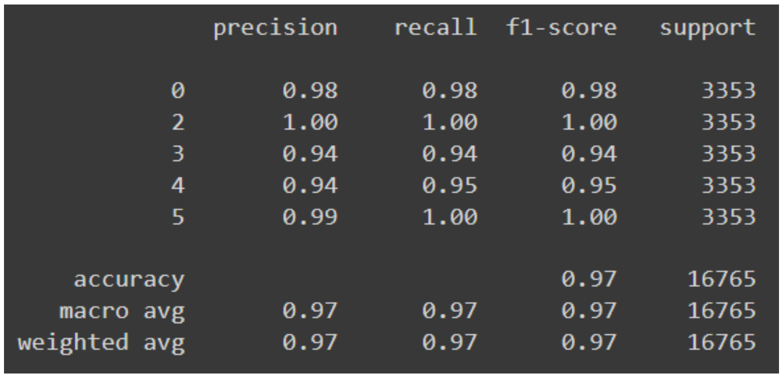
Figure 6.19: Building Model with Hyper-Optimized Parameters, Class Equalization and Forward Feature Selection Technique

Accuracy obtained implementing hyper-parameter optimized model with class equalization technique and Forward Feature Selection Technique is 96.98%.

6.3: Results and Discussions:

6.3.1. Classification report:

A Classification report is used to measure the quality of predictions from a classification algorithm.



```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      3353
           2       1.00      1.00      1.00      3353
           3       0.94      0.94      0.94      3353
           4       0.94      0.95      0.95      3353
           5       0.99      1.00      1.00      3353

    accuracy                           0.97     16765
   macro avg       0.97      0.97      0.97     16765
weighted avg       0.97      0.97      0.97     16765
```

Figure 6.20: Classification Report

a. Precision: It is the ability of the classifier to provide correct class labels, it is the ratio of true positives and the sum of true positives and false positives.
b. Recall: It is the ability of a classifier to find all the positive instances.
c. F1 – score: 2*(Recall * Precision) / (Recall + Precision)
   It is the weighted mean of precision and recall, making sure the ideal value is 1.0 and worst value is 0.0. Usually, F1 score is used to compare the models over the accuracy score.
d. Support: Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing.

e. Accuracy: TP+TN/TP+FP+FN+TN

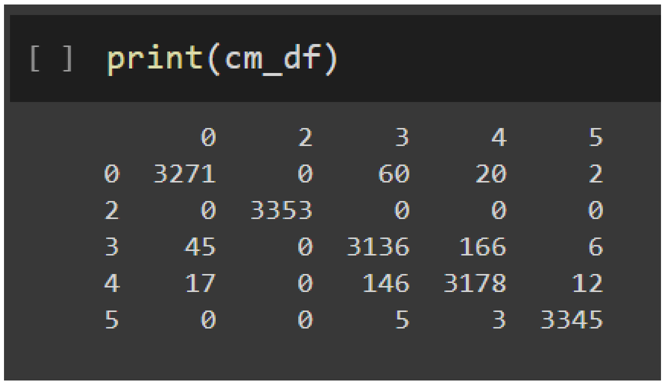
   It is the most basic performance measure of a model; it is the ratio of predicted and total observations.

f. Macro_Avg: The macro-averaged F1 score (or macro F1 score) is computed by taking the mean per-class of the F1 scores. It aims at treating all the classes equally irrespective of their support scores.

g. Weighted_Avg: The weighted-averaged F1 score is calculated by taking the mean of all per class F1 scores while considering each class's support**.**

6.3.2. Confusion matrix:

It is used to evaluate the performance of a classifier. The idea is to count the number of times an instance of class A is classified as class B. Each row refers to an actual class and each column refers to a predicted class. We can calculate the precision and recall from the confusion matrix as well, which can be used to compare between models.



```
[ ] print(cm_df)

        0     2     3     4     5
0    3271     0    60    20     2
2       0  3353     0     0     0
3      45     0  3136   166     6
4      17     0   146  3178    12
5       0     0     5     3  3345
```

Figure 6.21: Confusion Matrix

Precision: (TP) / (TP+FP)

Penalizes the false positives.

Recall: (TP) / (TP+FN)

Penalizes the false negatives.

For example:

Considering two models where apples are used, model A of a supermarket and model B of a cider company. In model A, the main aim is to sell off the apples and make profits, few bad apples wouldn't make any difference but few of the good apples going to waste will lead to loss, hence, here the model with the higher recall would be selected, as it would penalize the false negatives, as they can't have good apples classified as bad mistakenly. In case of model B, even few of the bad apples can spoil a whole of cider. Hence, here the model with higher precision would be chosen, as it penalizes the false positives, here, they can't have bad apples to be mistakenly be classified as good.

6.3.3. Percentage accuracies and misclassification for each class:

True Positive: The actual value was True, and the model predicted as True
True Negative: The actual value was False, and the model predicted False.
False Positive: The actual value was True, and the model predicted False. This is also known as a Type II error.
False Negative: The actual value was False, and the model predicted True. This is also known as a Type I error.

Class 0: Walking, Total: 3353

TP:3271

TN:13350

FP:82

FN:62

Percentage Accuracy of Prediction: (TP_0/ $TOTAL_0$) * 100 = 97.5%

Percentage Misclassification: [($TOTAL_0$ – TP_0)/ $TOTAL_0$] * 100 = 2.44%

Class 2: Sitting, Total: 3353

TP:3353

TN:13133

FP:0

FN:0

Percentage Accuracy of Prediction: (TP_0/ $TOTAL_0$) * 100 = 100%

Percentage Misclassification: [($TOTAL_0$ – TP_0)/ $TOTAL_0$] * 100 = 0%

Class 3: Slope, Total: 3353

TP: 3136

TN: 6538

FP: 172

FN: 151

Percentage Accuracy of Prediction: (TP_0/ $TOTAL_0$) * 100 = 93.5%

Percentage Misclassification: [($TOTAL_0$ – TP_0)/ $TOTAL_0$] * 100 = 6.47%

Class 4: Stairs, Total: 3353

TP: 3178

TN: 3345

FP: 12

FN: 3

Percentage Accuracy of Prediction: (TP_0/ $TOTAL_0$) * 100 = 94.7%

Percentage Misclassification: [($TOTAL_0$ – TP_0)/ $TOTAL_0$] * 100 = 5.2%

Class 5: Standing, Total: 3353

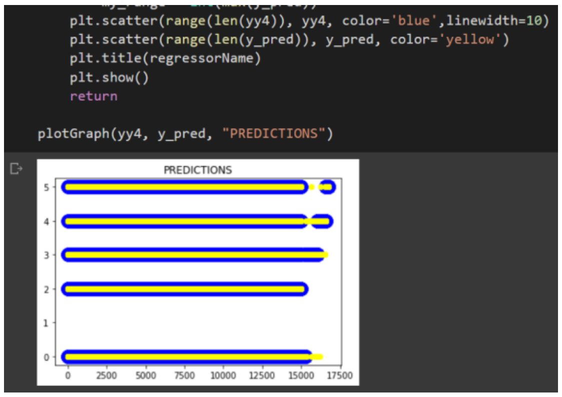
TP: 3345

Percentage Accuracy of Prediction: (TP_0/ $TOTAL_0$) * 100 = 99.7%

Percentage Misclassification: [($TOTAL_0$ – TP_0)/ $TOTAL_0$] * 100 = 0.23%

From the results above Sitting Class, that is Class 2, has the highest prediction accuracy.

Plotting Prediction Classes against Actual Classes to portray the accuracy of model predictions:



```
    plt.scatter(range(len(yy4)), yy4, color='blue',linewidth=10)
    plt.scatter(range(len(y_pred)), y_pred, color='yellow')
    plt.title(regressorName)
    plt.show()
    return

plotGraph(yy4, y_pred, "PREDICTIONS")
```

PLOTTING THE ACTUAL VALIDATION OUTPUTS WITH THE PREDICTED OUTPUTS FOR EACH OF THE CLASSES.

ACTUAL: BLUE

PREDICTED: YELLOW

Figure 6.22: Plotting Prediction Classes against Actual Classes

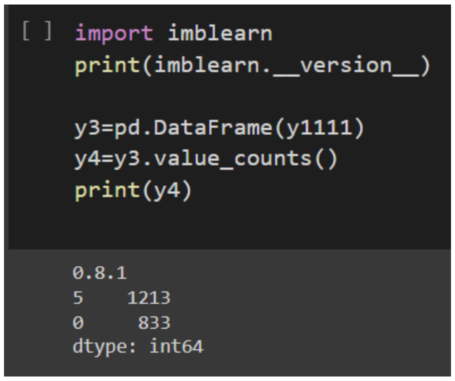
From the results above Sitting class that is Class 2 has the highest prediction accuracy.

To conclude the research one last trail was conducted with the aim of estimating the model efficiency when a totally unknown dataset is provided to the model, who's trails have been conducted in varied circumstances than of the training set, making sure no part of the validation set 2 has been introduced to the training data.

Shape of validation set:

```
(2048, 28)
```

Validation data set consisted of two classes 'Walking' as 0 , and 'Standing' as 5.

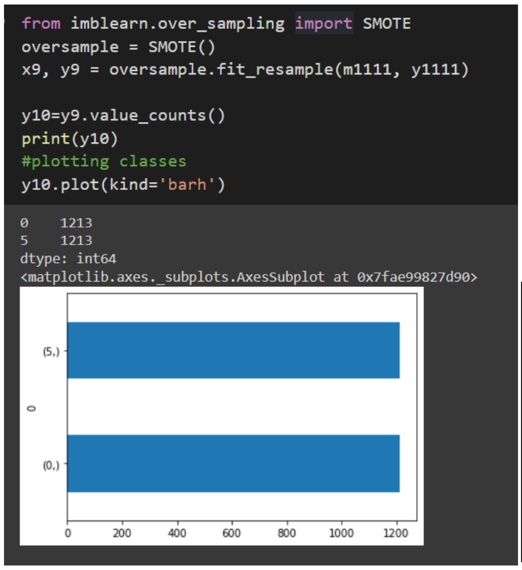


```
[ ] import imblearn
    print(imblearn.__version__)

    y3=pd.DataFrame(y1111)
    y4=y3.value_counts()
    print(y4)
```

```
0.8.1
5    1213
0     833
dtype: int64
```

Figure 6.23: Class Distribution for Validation Dataset 2

Class Balancing:



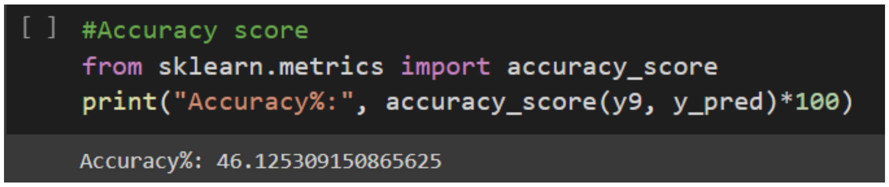
```
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
x9, y9 = oversample.fit_resample(m1111, y1111)

y10=y9.value_counts()
print(y10)
#plotting classes
y10.plot(kind='barh')
```

```
0    1213
5    1213
dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7fae99827d90>
```

As we can observe by using SMOT algorithm both the classes have been equalized to contain equivalent to 1213 rows.

Figure 6.24: Class Equalization for Validation Dataset 2

Final Model after applying Hyper-Parameter Optimization, Class Balancing and Forward Feature Selection Wrapper method:



```
[ ] #Accuracy score
    from sklearn.metrics import accuracy_score
    print("Accuracy%:", accuracy_score(y9, y_pred)*100)

    Accuracy%: 46.125309150865625
```

Figure 6.25: Building Model with Hyper-Optimized Parameters, Class Equalization and Forward Feature Selection Technique

The model’s resultant efficiency of prediction of various activities on totally unseen data was found to be about 46.12%

6.4: Conclusion:

This chapter is a step towards recognition of daily human activities, proposing an approach based on sensor fusion and feature selection. After rigorously performing multiple simulations and referring to the work of multiple researchers, the authors have come up with the proposed methodology. As concluded in previous chapters, about a 100 ms delay is present between human action and detection of culminative sEMG signal, exploiting this delay as a time window of extracted features, the authors have tried to decode the intension of motion. The vision is to facilitate the development of a robust system to differentiate and identify between the intension of various activities. Empowering the exoskeleton with this ability, will smoothen the process of performing actions, hence, making the exoskeleton the master, assisting not resisting the human body. However, the model efficiency as we have observed is quite limited when it comes to totally unknown datasets, this aspect can only be improved by further rigorous training by introducing the training dataset with much varied and vast datasets in order to eventually make it capable of quality predictions on totally unknown datasets as well, laying the foundations of further development of human intension prediction.

6.5: References:

9. https://databricks.com/glossary/machine-learning-models
10. ht tps://monkeylearn.com/blog/classification-algorithms/
11. https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html#:~:text=There%20are%20four%20types%20of,%2Dsupervised%2C%20unsupervised%20and%20reinforcement.

12. https://machinelearningmastery.com/types-of-learning-in-machine-learning/

13. https://docs.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model

14. https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501

15. Badawi AA, Al-Kabbany A, Shaban HA. Sensor type, axis, and position-based fusion and feature selection for multimodal human daily activity recognition in wearable body sensor networks. Journal of Healthcare Engineering. 2020 Jun 7;2020.

16. Chereshnev R, Kertész-Farkas A. Hugadb: Human gait database for activity recognition from wearable inertial sensor networks. InInternational Conference on Analysis of Images, Social Networks and Texts 2017 Jul 27 (pp. 131-141). Springer, Cham.