Faculty of Natural and Mathematical Sciences
Department of Informatics

King's College London
Strand Campus
London
United Kingdom

7CCSMPRJ
Individual Project Submission 2019/20

Name: Hsin-Ju, Chan
Student Number: 1920253
Degree Programme: MSc Data Science
Project Title: Activity Classification Using Machine Learning Techniques
Supervisor: Simon Parsons
Word count: 10200

RELEASE OF PROJECT
Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

☒ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature: Hsin-Ju, Chan

Date: August 21, 2020

Department of Informatics
King's College London
United Kingdom

7CCSMPRJ Individual Project

Activity Classification Using Machine Learning Techniques

Name: Hsin-Ju, Chan
Student Number: 1920253
Degree Programme: Data Science

Supervisor's Name: Simon Parsons

This dissertation is submitted for the degree of MSc in Data Science

Abstract

This project applies three machine learning algorithms, support vector machine (SVM), K-nearest neighbour (KNN) and naïve Bayes (NB) for classifying different activities, compare the performance among them and improve these classifiers by under-sampling. SisFall dataset is used here for testing the classifiers. Data has been denoised by Butterworth low pass filter and segmented the pre-processed data into 5 seconds a chunk. Each chunk has been summarized into 12 features and feature extraction techniques will be implemented to reduce dimensionality. There are three versions of dataset after segmentation which are based on different methods for the given labels. Radial basis function (RBF) SVM is the best classifier in binary version of dataset, which is 89.41% on accuracy score. In 9 classes and 13 classes version of dataset, KNN obtains the best accuracy score, which are 67.98% and 60.5% respectively. Moreover, comparing the results after addressing data imbalanced problem, though the accuracy score decreased, the sensitivity and specificity score increased when using one-sided selection (OSS) and synthetic minority oversampling technique (SMOTE) + Tomek link (T-Link). In conclusion, KNN may be the most suitable algorithms for classifying gait patterns and the improvement of the classifier implies that label imbalanced is one of the main issues.

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Professor Simon Parsons, for his guidance through each stage of the process, insight and patience, and support throughout this project.

Furthermore, I would like to thank my parents and friends for supporting and encouraging me during the time working on the project.

Table of Content

List of Table

List of Figure

Nomenclature

FDS	Fall Detection System
ADLs	Activities of Daily Living
PCA	Principal Components Analysis
LDA	Linear Discrimination Analysis
SVM	Support Vector Machine
NB	Naïve Bayes
KNN	K-nearest Neighbour
LR	Logistic Regression
DT	Decision Tree
RF	Random Forest
RBF	Radial Basis Function
T-Link	Tomek Link
ENN	Edited Nearest Neighbour
OSS	One-sided Selection
CNN	Condensed Nearest Neighbour
SMOTE	Synthetic Minority Oversampling Techniques
AD	Acceleration data
RD	Rotation data
LOOCV	Leave-one-out Cross-validation
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Chapter 1 Introduction

1.1 Problem Statement

Nowadays, ageing society has become a normal situation for all developed countries and developing countries will soon be entering to the ageing society. The demand for the healthcare system increased accordingly since the population of the elderly grow rapidly (Vallabh et al. 2016, 1-9). Falls become a public health concern simultaneously and lead to devastating consequences for elders, such as morbidity, mortality, and loss of independence (Perell et al. 2001, M761-M766). A report also shows that about 30% of the population over the age of 65 falls each year and it causes around 25% of deaths for 13% of people older than 65 (Axer et al. 2010, 265-274).

Owing to the problems mentioned above, the use of fall detection system is getting popular. At present, there are two research tracks about fall-related system, which are fall detection and fall prevention. Plenty of research works have focused on fall detection in recent years, and the main purpose of the system is to detect fall accurately when it happens and rescue the victim in a short time to reduce elder's pain (Hussain et al. 2019, 4528-4536). For example, to run a fall detection system in a real-time based data from an older person who was living on their own and use this to identify when they need help because of the fall.

1.2 Aims and Objectives

SisFall (Sucerquia, López, and Vargas-Bonilla 2017, 198) is the public dataset we used in this project, also acquire with a wearable device built up with two accelerometers and one gyroscope. It includes 19 activities of daily living (ADLs) and 15 falls performed by 23 young adults and 15 elders. This project aims to classify ADLs and different fall activities by several supervised learning algorithms. In addition, it is to find out which kind of classifier has the highest accuracy on classifying all the activities. To meet the project aims, the objective of this project can be separated more specifically as follows:

- Understand the gait pattern for each activity
- Implement three filtering methods to denoise raw data and choose the best one
- Segment the denoising data and give a different version of labels (i.e. binary and multi-class)
- Extract the best features for the data by principal components analysis (PCA) and linear discrimination analysis (LDA)
- Test three machine learning algorithms for each version of data and compare their efficiency
- Improve the best classifier through under-sampling the data

1.3 Project Plan

To construct this project, it is important to have a certain understanding of gait pattern for each activity, therefore, gait patterns of all subjects will be plotted and analysis the differences. After fully understanding the patterns, data denoising method such as Butterworth low pass filter, moving average smoothing and down-sampling will be implemented. For the next step, the original plan is to detect peak and extract cycles, however, because the patterns in this public dataset are most an action such as stand or sit, so it does not have enough walking cycles. Also, pattern after denoising remains hard to extract cycles since it is difficult to separate local peaks and true peaks. Therefore, we decide to segment the patterns into 5 seconds a chunk and summarize each chunk into 4 features such as mean, standard deviation, maximum value and minimum value. Meanwhile, labels are given to the dataset after summarizing and we generate three versions of dataset which has different labels, including binary, 9 classes and 13 classes version.

Since each dataset includes 12 features, PCA and LDA are used to reduce dimensionality to improve classifiers. We select 10-fold cross-validation and stratified 10-fold cross-validation for training and support vector machine (SVM), k-nearest neighbour (KNN) and naïve Bayes (NB) as models. In this step, we have to find out which training method and model has best performance for each version of dataset. Finally, we choose the best model to implement under-sampling techniques to address the problem of imbalanced label and find out which technique has the best result and compare the performance before and after data balancing.

1.4 Organization

Chapter 2 (Technical Background) provides a short description of some techniques used in this project.

Chapter 3 (Exploratory Data Analysis) presents the dataset with plotting them out and compare the dataset before and after denoising.

Chapter 4 (Project Design) describes the details of the implementation of this project. This chapter will be divided into several parts according to the project structure. It is to explain the reason method for choosing and the process of methods implementation.

Chapter 5 (Experiment Results and Analysis) evaluates the project objectives and some improvement of the performance for the experiments.

Chapter 6 (Conclusion) summarize the whole project achievement and provides directions about the future pathway of this project.

Chapter 2 Literature Survey

This section represents the background knowledge of gait recognition and methodologies from several studies. It lists the important studies about fall detection which guides to construct and demonstrate experiments.

Gait recognition, as an important technology of identifying human behaviour, have several scenarios to apply and the most common one is individual authentication. Unlike fingerprinting and face recognition, human walking postures do not require short distance recognition and high-quality image, therefore, these are the reasons why gait becomes a prominent biometric nowadays. Environments such as banks, military installations and airports gain advantages from gait recognition since gait can be observed at a distance and can identify possible threats efficiently, which gives time to react before the threats arrived. (Dawson 2002) Studies in Medicine (Murary, Drought, and Kory 1967, 290-332) has shown that gait is unique for every person owing to the movements between human body and joints. The height, weight, step and step speed are the factors that cause the differences of gait trait. In addition, non-invasiveness and difficulty in disguise are the advantages which lead researchers to focus on gait behaviour recognition. According to the above advantages, there is a wide range of application in fields today such as fall detection and mobile health and security monitoring. (Liu, Zhao, and Wei 2018, 134-145)

Most research works related to fall detection systems (FDS) have been proposed for human fall detection (Hussain et al. 2019, 4528-4536). Based on existing works, fall detection system can be divided into three categories, ambience FDS, camera-based FDS and wearable FDS. Ambience FDS usually combined optical sensors and environmental sensors, such as pressure sensors or audio sensors, and are construct in the environment to detect fall. Camera-based FDS detects falls by installing cameras on a specific place and continually monitoring elder's activities. Wearable FDS is getting more common nowadays due to its small size, portable and convenient. This device collects data from the user and these data will be filtered to separate noise from the signal. The filtered signal will then be processed for detecting falls (Hussain et al. 2019, 4528-4536). In addition, there are several types of motion systems and sensors in wearable sensors, including accelerometer and gyroscope (Tao et al. 2012, 2255-2283). These two types of motion systems are used in the dataset we selected for this study.

Most studies in this area focus on accelerometer-based fall detection, and it can separate into two main approaches, threshold-based and machine learning-based. Threshold-based method is mostly used in previous studies to detect impact and posture, nevertheless, machine learning-based gradually become widely used recently due to the improvement of the devices. Researchers started to use advanced technologies and algorithms to show improved results (Aziz et al. 2017, 45-55). Therefore, a research (Aziz et al. 2017, 45-55) compared the accuracy for 10 different fall detection algorithms, including 5 threshold-based methods and 5 machine learning-based approaches, such as SVM, NB and KNN, logistic regression (LR) and decision tree (DT). The result indicated that machine learning-based was better than threshold-based. Among all the methods in machine learning-based, SVM got the highest sensitivity and specificity of 96%. Furthermore, the error rate for these five algorithms were lower than 10%, but the performance of SVM, KNN and NB were slightly better than others.

Machine learning in fall detection helps to detect falls accurately based on a subject's activity patterns (Ramachandran and Karuppiah 2020). It can easily raise alarms when subjects change their movements, however, it may have too many alarms which probably occurs mistakes. The main purpose of fall detection system is to raise alarms when falls actually

happen, thus, it becomes important that false positive and false negatives need to be minimized in fall detection system (Ramachandran and Karuppiah 2020). Several performance metrics include sensitivity and specificity are used to prevent from wrong classification. Moreover, steps such as data cleaning, pre-processing and feature extraction also play a vital role in fall detection.

For the construction of fall detection classification system, most studies included several stages such as pre-processing, time segmentation, feature extraction, feature selection, model training and classification (Ramachandran and Karuppiah 2020). The purpose of data pre-processing is to reduce noise and make the data suitable for the method which will probably be used. In (Vallabh et al. 2016, 1-9) and (Hussain et al. 2019, 4528-4536), data were pre-processed to remove noise from the original data by different filters. The median filter and a low pass filter were used in the former study, while the latter one applied for only a low pass Butterworth filter. Vallabh et al. implemented median filter for reducing the noise produced by external vibration and mechanical resonance, while low pass filter was to remove the acceleration generated by body movement (Vallabh et al. 2016, 1-9). For the study from Hussain et al., Butterworth low pass filter was chosen as the method of removing unwanted noise due to its final performance and low computational cost for the system (Hussain et al. 2019, 4528-4536). Both of them got well performance on smoothing the signal. The procedure in these two studies are similar, both of them employ feature extraction, model training and classification. However, the slightly difference is that (Hussain et al. 2019, 4528-4536) segments the pre-processed data into 10-second small chunks.

Furthermore, features were extracted similar between them, for example mean, median, maximum value, minimum value and variance. Feature extraction and selection influence the performance of the classifier directly, since the accuracy score of the system will be determined by the selected features. In other words, it is to determine which features are informative and can separate ADLs and falls effectively. Therefore, these two pieces of research implement feature extraction methods and feature selection algorithms. With these approaches, the size and dimension of feature sets will be reduced, which leads to better performance for each classifier. The result of the study that employed feature extraction got high accuracy, sensitivity and specificity on KNN, SVM and random forest (RF). Almost all are close to 100%, of which KNN is the highest. The other one showed that KNN, ANN and SVM has the best accuracy compared to others, and the best classifier is KNN with k equal to 5 obtained an 87.5% accuracy.

Chapter 3 Technical Background

3.1 Data Denoising

This section divided into three parts, the first part is the use of filters, the definition of Butterworth filter and the different features of filters. The second part focus on moving average filter, describing its definition and benefit. The third part is about down-sampling.

3.1.1 Butterworth Low pass Filter

Filters can be used on plenty of application, especially on signal processing. It is to eliminate or choose the selected frequency from the spectrum, therefore, the signal that passes through the filter is the one we are going to collect.

Butterworth filter is a signal processing filter which is designed to flatten the signal in the passband. The characteristic of this filter is to acquire maximum flat frequency in the passband and gradually reduce the frequency into zero in the stopband. The attenuation rate of the first order Butterworth filter is -6dB for each octave, -12 dB for the second order Butterworth filter and third order at -18 dB and so on.

Filters can be divided by frequency selection, including low pass filter, high pass filter, bandpass filter, band reject filter and all pass filter. Low pass filter means that frequency higher than the cut-off will be reduced, while those are lower than cut-off will pass. On the other hand, high pass filter will pass the frequency higher than the cut-off and reduce those who are lower. Bandpass filter is a combination of low pass and high pass filter. And band reject filter is a parallel combination of low pass and high pass.

3.1.2 Moving Average Filter

Moving average filter is a widely used filter due to its simple concepts and easy to make use of. It is commonly used in time series data to smooth the fluctuations and highlight the long-term trends (Muntakim 2017). As the name indicates, moving average filter generates each point in the output signal by averaging multiple points in the input signal. In the equation form, it is written:

$$y[i] = \frac{1}{M}\sum_{j=0}^{M-1} x[i+j] \tag{2.1}$$

where x[ ] is the input signal, y[ ] is the output signal, M is the number of points.

In (Babu and Reddy 2014, 27-38), moving average filter is used to explore the nature of volatility before applying other models. The benefits of implementing the filter are to understand time series data in advance. Moreover, the performance accuracy in this study when adding moving average filter first was better than the other without this filter.

3.1.3 Down-sampling

Down-sampling is also a well-known method in signal processing. This approach is to reduce the sampling rate of a signal, in other words, it is to reduce the size of the data. In (Kale et al. 2003, 706-714), the reason for implementing down-sampling is that the original width vector of their data has redundancies, therefore, reduce the features into lower dimensions is necessary. Furthermore, the results after down-sampling did not deteriorate rapidly. It got good accuracy in different dimensions of features.

3.2 Machine Learning Methods

This section is about the machine learning algorithms which will be used in this project. Three algorithms are described in this part, including SVM, KNN and naïve Bayes. The definition of these three algorithms will be introduced here.

3.2.1 SVM

Support vector machine is a popular supervised learning model and algorithm, which is used for classification and regression analysis. It is a binary classifier and the purpose is to classify examples by a line or a hyperplane in multidimensional space. To find the hyperplane, we have to find the optimal decision boundary that maximizes the margin between the two classes. The equation form of hyperplane will be written as:

$$f(x) = W^T x + W_0 = 0 \tag{2.2}$$

where $w = [W_1, W_2, W_3 \dots W_d]$ and x is the support vectors. Therefore, the equation of finding the best parameter W and W_0 will be written as:

$$\begin{cases} W^T x + W_0 \geq 1 & when\ y_i = +1 \\ W^T x + W_0 \leq 1 & when\ y_i = -1 \end{cases} \tag{2.3}$$

where y_i is classes. This equation will be calculated several times to achieve the optimal parameters that can completely separate two classes.

There are three types of SVM such as linear SVM, non-linear SVM and multi-class SVM. For linear SVM, it includes linear separable cases and non-separable cases. To successfully acquire decision boundary in non-separable cases, the slack variables (ξi) will be added, which is to relax the constraints of decision boundary. For non-linear SVM, kernel function will be used, for example, radial basis function (RBF) kernel function, Sigmoid kernel function and Gaussian RBF kernel function. Multi-class SVM is to combine several classes of SVM classifiers and there are plenty of approaches such as one against one, one against all, binary decision tree and binary code. In this study, Linear SVM, RBF SVM, polynomial SVM and Sigmoid kernel SBM are chose to classify the activities.

3.2.2 KNN

K-nearest neighbour is also a supervised learning algorithm. It is to find the maximum classes in K closest neighbours for the given test instance and predict text instance as the maximum one. To get the closest neighbours, distance formula is often to be used, for instance, Euclidean distance, Manhattan distance and Chebyshev distance. Moreover, K is usually an odd number since even number has the opportunity to lead to an equal number in each class. However, when k equals to 1, it will be easier to overfit training data and when K equals to a large number, it may cause underfitting. The advantage of this algorithm is that it has high precision and insensitive to outliers, while the disadvantage is that it is time complexity and the model is highly relying on training data.

3.2.3 Naïve Bayes

Naïve Bayes classifier is a probabilistic model used when classification is not very clear at boundaries. It is a simple but effective method, moreover, it helps against overfitting. The assumption of NB is that all features are independent, therefore, the joint probability distribution can be calculated by multiplying conditional probability. The equation form can be written as:

$$\prod_j p(a_j|V_i) = p(a_1|V_i)p(a_2|V_i) \times \cdots \times p(a_n|V_i) \quad (2.4)$$

where a = [a_1, a_2, a_3, …, a_n] which is feature vector and V is the specific class. The output of NB classifier is the probability of each class after inputting feature vector and the class with maximize probability will be the predicted value. In the equation form, it will be written as:

$$V_{NB} = argmax \prod_{V_i \in V}^{j} p(a_j|V_i)p(V_i) \quad (2.5)$$

This machine learning algorithm is well-known for text classification, for example, to determine whether the mail is spam or not and to allocate the types of different files or books.

3.3 Data Balancing and Approaches

Imbalance datasets are those with severe skew in class distribution, which means the majority class has much more examples than the minority class. The reason why addressing imbalance condition is important is that it can influence many machine learning algorithms, in other words, it will ignore the minority class and make the result inaccurate. Therefore, several approaches can be used to solve this problem, the two main methods are to delete examples from the majority class, which is under-sampling, and the other is to generate new examples from the minority class, which is oversampling. There is also a method which is to combine oversampling and under-sampling.

This section will focus on the approaches which will be used in this project. The first part is under-sampling, focusing on some related techniques such as Tomek Link (T-Link), Edited Nearest Neighbours (ENN), One-Sided Selection (OSS) and Condensed Nearest Neighbour (CNN). The next part is the combination of oversampling and under-sampling, and it will explain the technique of SMOTE + Tomek Link.

3.3.1 Under-sampling

Random resampling is the most popular and naïve method that attempts to balance the dataset, including eliminate examples from the majority class randomly and duplicate examples from the minority class randomly. However, these methods have some limitations, for example, it may exclude some useful data when under-sampling, while oversampling may increase the number of examples that are not useful (Tahir, Kittler, and Yan 2012, 3738-3750). Due to these limitations, some techniques were introduced to improve random under-sampling, which is to resample the selected examples such as T-Link, CNN, ENN and OSS.

Hart (P. Hart 1968, 515-516) designed CNN as an algorithm to find the important data points. Data in the training set will be divided into three types: outliers, prototypes and absorbed points. Outliers include the points that will not be classified correctly, prototype gathers the minimum set of points for correctly recognized the non-outlier points, and absorbed points are the points which can be classified correctly. Therefore, rather than the whole dataset, new samples only need to compare with prototype points.

However, the criticism of CNN is that examples are chosen randomly, especially initially, which allows storing redundant examples. Therefore, T-Link is a modification to CNN. It is a method to find pairs of examples that have the smallest Euclidean distance to each other one from each class (Anonymous1976, 769-772). These cross-class pairs are T-Link which defines the class boundary. Though it can find useful examples, it is still not the best method for under-sampling. This approach is often combined with other methods such as CNN and synthetic minority oversampling technique (SMOTE).

Another rule for finding noisy examples is ENN. This rule is to locate those wrongly classified examples in the dataset through k nearest neighbours and remove them before applying k = 1 nearest neighbour (D. L. Wilson 1972, 408-421). During the process of under-sampling, the rule is to remain the correctly classified examples and remove those examples that are misclassified to the minority class (He and Ma 2013). Similar as T-Link, it can only remove noise along the class boundary, the dataset will not be balanced after implementing these techniques.

OSS is an under-sampling rule that combines T-Link and CNN. As the above mention, T-Link is to remove ambiguous examples from the majority class and CNN is to remove redundant data points from the majority class (Fernández and others 2018). Through this method, the dataset becomes balanced.

3.3.2 The Combination of Oversampling and Under-sampling

This approach is to combine random oversampling and random under-sampling techniques. It can use different sampling strategies for both resampling methods. After applying this method, it will generate a balanced dataset.

Batista et al. (Batista, Prati, and Monard 2004, 20-29) applied this kind of approach on their study, and the method is the combination of SMOTE and T-Link. The first step of this approach is to balance the dataset with SMOTE and then delete the T-Links from all classes.

Chapter 4 Exploratory Data Analysis

This section has separate into three parts. The first part is the description of the public dataset, it will focus on the subjects, experiments and other features. The second part is to visualize the dataset, including the raw data and the data after implementing denoising methods. It is to explore what the data looks like, how to work on it and which kind of filter is the best to interpret the data. The final part will compare the difference between the use of filters and decide which filter to use for this project.

4.1 Data Acquisition and Description

As the above section introduced, SisFall (Sucerquia, López, and Vargas-Bonilla 2017, 198) is the dataset used in this study. There are 38 subjects including 23 youngsters and 15 elders. Data are collected by 2 accelerometers and 1 gyroscope at a frequency sample of 200 Hz, which means there are 200 examples per second. There are two types of experiments, ADLs and falls. ADLs include 19 different activities and the other includes 15 different falls. All activities have a different number of trials and duration, which will be shown as follow:

Code	Activity	Trials	Duration [S]
D01	Walking Slowly	1	100
D02	Walking Quickly	1	100
D03	Jogging Slowly	1	100
D04	Jogging Quickly	1	100
D05	Walking upstairs and downstairs slowly	5	25
D06	Walking upstairs and downstairs quickly	5	25
D07	Slowly sit in a half height chair, wait a moment, and up slowly	5	12
D08	Quickly sit in a half height chair, wait a moment, and up quickly	5	12
D09	Slowly sit in a low height chair, wait a moment, and up slowly	5	12
D10	Quickly sit in a low height chair, wait a moment, and up quickly	5	12
D11	Sitting a moment, trying to get up, and collapse into a chair	5	12
D12	Sitting a moment, lying slowly, wait a moment, and sit again	5	12
D13	Sitting a moment, lying quickly, wait a moment, and sit again	5	12
D14	Being on one' s back change to lateral position, wait a moment, and change to one' s back	5	12
D15	Standing, slowly bending at knees, and getting up	5	12
D16	Standing, slowly bending without bending knees, and getting up	5	12
D17	Standing, get into a car, remain seated and get out of the car	5	25
D18	Stumble while walking	5	12
D19	Gently jump without falling (trying to reach a high object)	5	12
F01	Fall forward while walking caused by a slip	5	15
F02	Fall backward while walking caused by a slip	5	15
F03	Lateral fall while walking caused by a slip	5	15
F04	Fall forward while walking caused by a trip	5	15
F05	Fall forward while jogging caused by a trip	5	15
F06	Vertical fall while walking caused by fainting	5	15
F07	Fall while walking, with use of hands in a table to dampen fall, caused by fainting	5	15
F08	Fall forward when trying to get up	5	15
F09	Lateral fall when trying to get up	5	15
F10	Fall forward when trying to sit down	5	15
F11	Fall backward when trying to sit down	5	15
F12	Lateral fall when trying to sit down	5	15
F13	Fall forward while sitting, caused by fainting or falling asleep	5	15
F14	Fall backward while sitting, caused by fainting or falling asleep	5	15
F15	Lateral fall while sitting, caused by fainting or falling asleep	5	15

Figure 1 All activities in the dataset

The units for each example in this dataset are in bits and different sensor has distinct characteristics. This project selects the examples from accelerometer 1 and gyroscope and the characteristics for these two sensors are as follow:

	Resolution	Range
Accelerometer 1	13 bits	+-16g
Gyroscope	14 bits	+-8g

Table 1 Characteristics of selected sensors

To convert the units from bits to gravity, the equations are provided. Acceleration [g]: [(2*Range)/(2^Resolution)]*AD is to convert the acceleration data (AD) from bits to gravity, while Angular velocity [°/s]: [(2*Range)/(2^Resolution)]*RD is to convert the rotation data (RD) from bits to angular velocity. Therefore, the data used in this project are all converted into gravity.

4.2 Data Visualization

This section will focus on the visualization of the dataset. It will first show the pattern for each activity for some specific subjects without any filters and describe the gait pattern for the activity. The next part indicates the pattern that has been pre-processed, which is to denoise by three methods: Butterworth low pass filter, moving average filter and down-sampling.

4.2.1 Raw Data

This part presents various line plots of some subjects which shows the gait pattern of each activity. 4 subjects are selected in this part, SA06, SA15, SE06 and SE11. SA06 and SE06 are both males, however, SA06 is the youth who aged 21, 169 centimetres and 58 kilograms and SE06 is the elder who aged 60, 163 centimetres and 79 kilograms. As for SA15 and SE11, they are both females. SA15 is the youth who aged 25, 160 centimetres and 52 kilograms, and SE11 is the elder who aged 66, 169 centimetres and 63 kilograms. For these four subjects, the gait pattern of the same activity only has slightly different. One of the reasons is that subjects will start their action at different time and leads to a time delay. However, for some similar activities, some part of the gait pattern shows differently.

For the first four activities, “walking slowly”, “walking quickly”, “jogging slowly” and “jogging quickly”, due to their long duration in a trial, the plot is really hard to read. Therefore, figure 2 is the 3-second segments of those activities, which makes the comparison clearer. This figure shows the difference between similar activities and their speed. It is obvious that the gait cycle duration of walking is longer than jogging. Take the subject SA05 as an example. The gait cycle duration of SA05 at top-left is around 120 samples, while the duration of the plot at bottom-left is about 100 samples. In other words, the duration of walking is about 0.6 seconds and jogging is approximately 0.5 seconds. In addition, the amplitude of jogging is severer than walking which presents through the acceleration on y-axis. The amplitude of jogging is around 2g, while walking is about 0.5g to 1g.

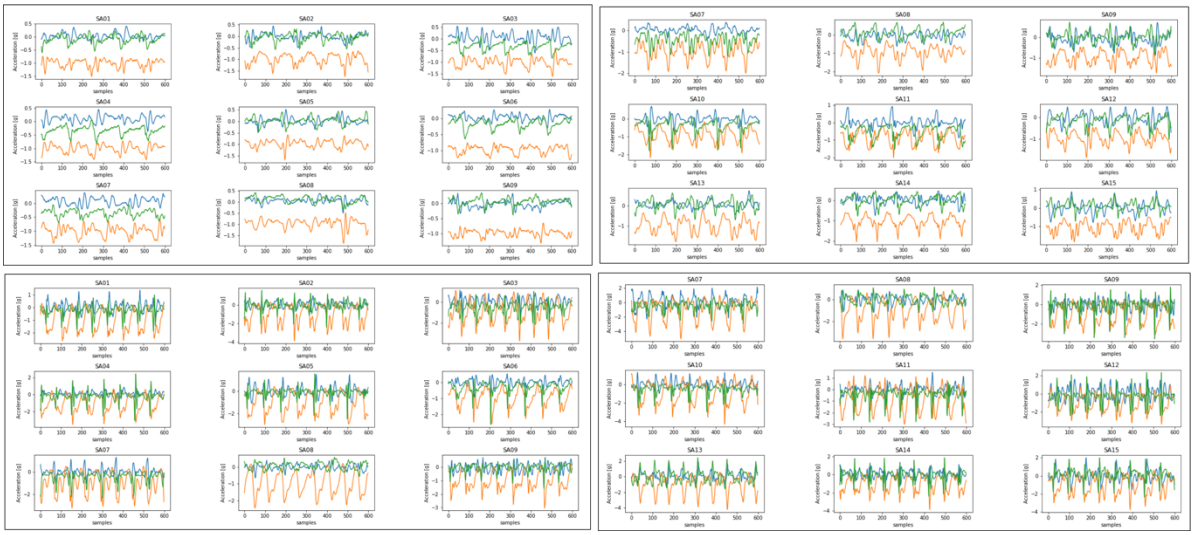


Figure 2 3-second gait patterns of activity walk slowly(top-left), walk quickly (top-right), jog slowly (bottom-left), jog quickly(bottom-right)

For the next two activities walking upstairs and downstairs slowly (D05) and quickly (D06). The pattern shows in figure 3 to 6 is clear that there are two kinds of actions and most subjects stop a while when changing their action. Furthermore, the gait cycle pattern in D06 is intensive since the speed is faster, while in D05 the pattern looks much loose due to its slower speed. However, some subjects' pattern did not show a straight line when they stop a few seconds before changing the action. It may because there are lots of noise in the data or the subject did not stop before walking downstairs.

Activities such as D07, D08, D09 and D10 are quite similar. The differences between them are the speed sitting on a half or low height chair and stand up slowly or quickly. The pattern shows in below figures also look quite the same since the major difference is the speed. In D07 and D08, the gait cycle duration in the former one is longer than the latter one, and it is because the movements of D07 are slow. Patterns of D09 and D10 look alike as well. They are also separated by their distinct speed.

D11 is an activity similar to fall but it actually collapses in a chair, which does not belong to fall. The pattern for this movement has an obvious fluctuate when subjects collapse into a chair. D12 and D13 are also similar. Both of them are sit and lying down, wait a moment and sit again. Speed when lying down leads to different patterns. D12 lies down slowly, therefore the duration of this movement is longer, while D13 is the opposite. D14 is to change their back when lying down. It can be separated by the distinct trend for each axis. D18 may easily misclassify into walking since there is almost three out of four of time is walking cycle. Only when the subject stumble the pattern fluctuates significantly. D19 is to jump two times during the trial and the pattern has obvious fluctuation when the subject jump.

For each falling patterns, all of them have obvious raised when they fall, and the amplitude is much higher than all other no-fall activities. F01 to F03 presents the pattern of falling caused by slip but in different falling posture such as fall forward, fall backward and lateral fall. Their patterns look alike, but the amplitude of all 3 axes are different. Falling patterns of F04 and F05 are caused by a trip but have distinct activity before tripping. F04 was walking and F05 was jogging, therefore, F05 has larger amplitude than F04 apparently. F06 and F07 are both caused by fainting, but the pattern only has slightly different even if one of them have

dampened fall by hands. F08 and F09 is the pattern about falling when trying to get up but fall with different posture. F10 to F12 are about falling in different posture when trying to sit down and F13 to F15 are also falling in different posture but is caused by falling asleep. These 8 falling activities can be separated by observing the trends of three axes.

The falling patterns that mentioned above can be separated through different aspects such as the reason for falling and the posture of falling. In this project, falling activities will be classified by the posture of falling.

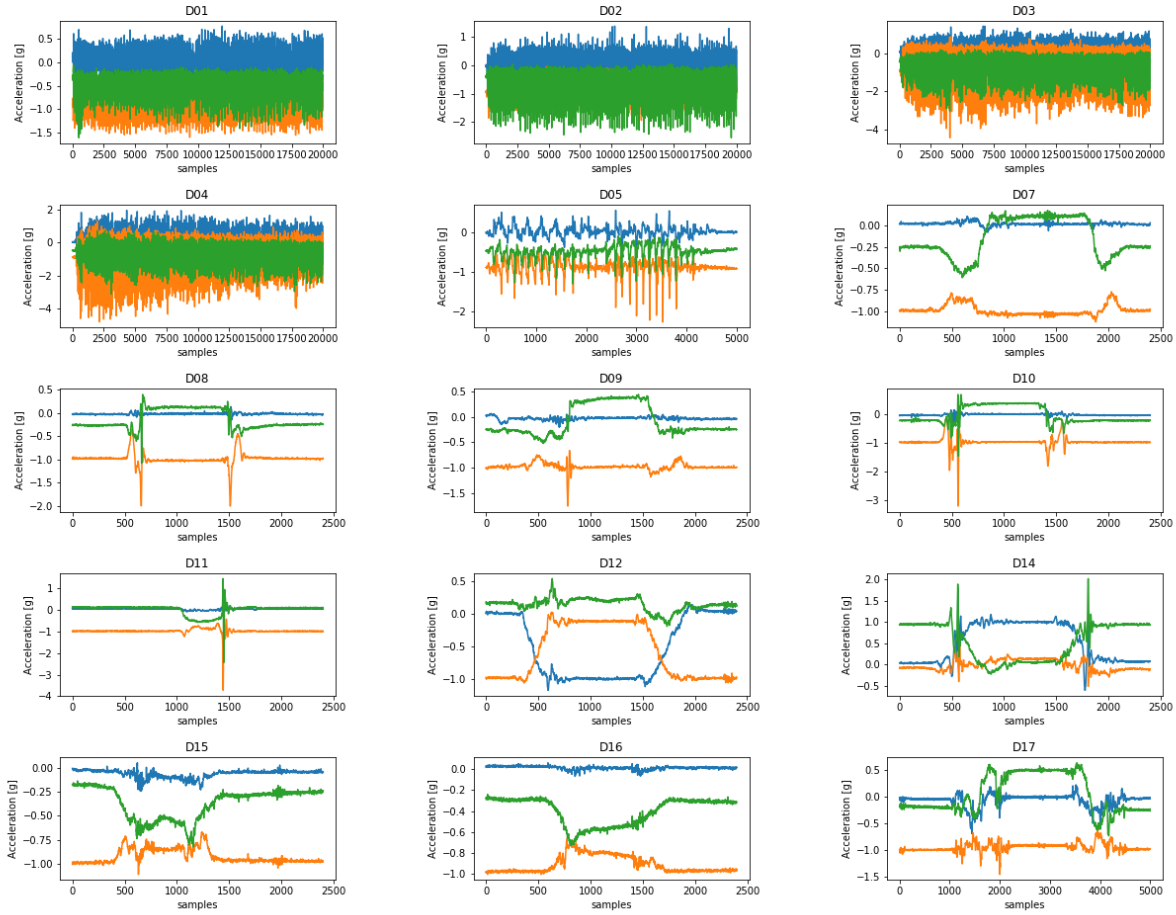


Figure 3 Gait Pattern of SE11

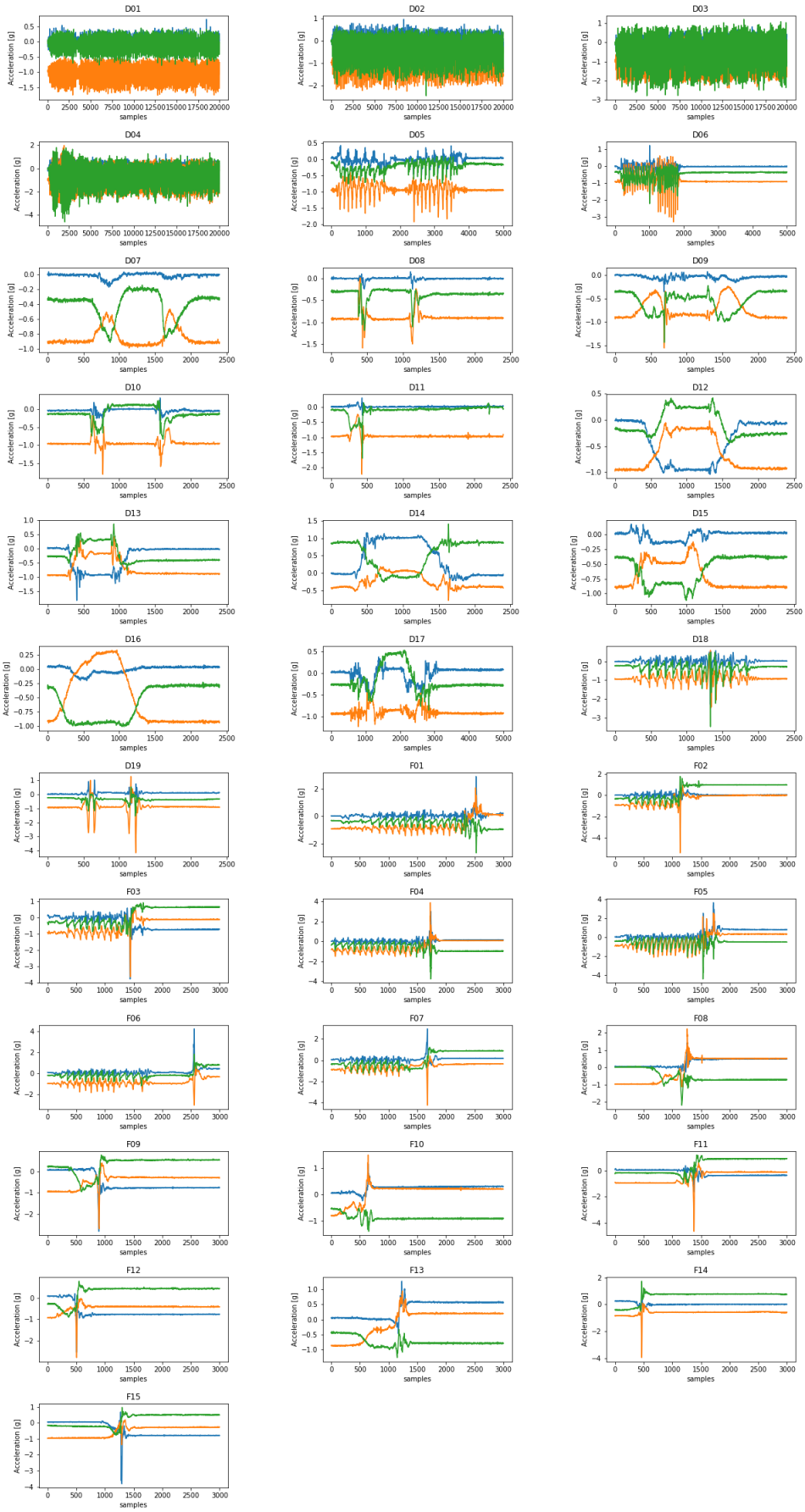


Figure 4 Gait Pattern of SE06

Figure 5 Gait Pattern of SA06

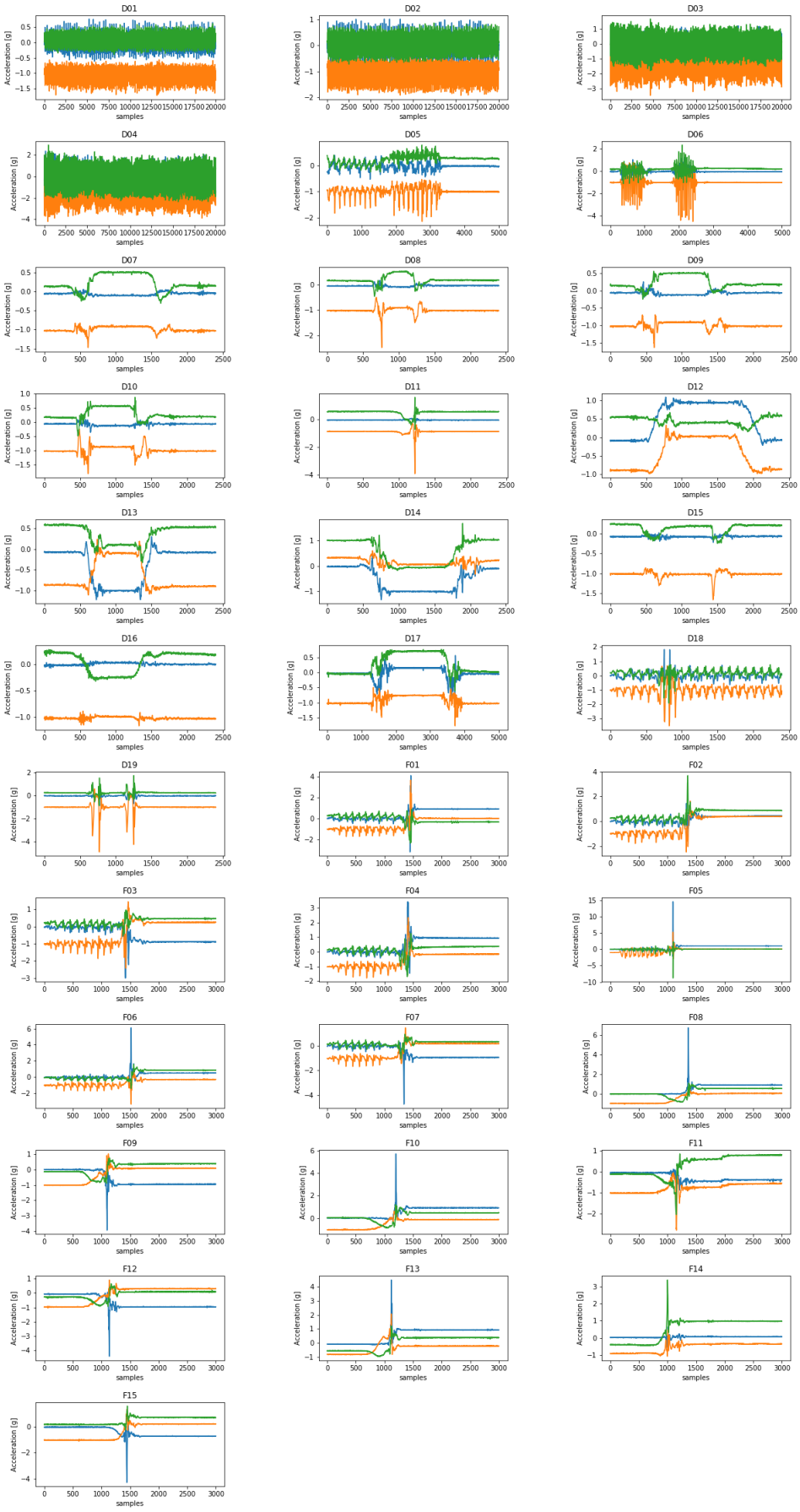


Figure 6 Gait Pattern of SA15

4.2.2 Data Denoising

This section presents the gait pattern after denoising the data. The denoising methods that will be used here has mentioned in chapter 2, which is Butterworth low pass filter, moving average filter and down-sampling. The comparison of before and after plots will be shown here to demonstrate the difference of gait pattern after implementing each denoising method. Due to the above figures show, X-axis in most activities do not present gait cycle clearly, therefore, the following comparison figures will focus on Y and Z axes. The figures below will indicate the pattern of SA06 and the activity is "walking slowly". In addition, this figure captures from 20 seconds to 40 seconds since the duration of a trial is quite long.

A. Butterworth low pass filter

Figure 7 shows the gait pattern before and after implementing Butterworth low pass filter. When implementing this filter, some parameters needed to be chosen such as cut-off frequency and order. Therefore, the cut-off frequency here is 5 and the order is 4. Compared with the raw signals and the filtered patterns, it is obvious that most noises have been filtered and can clearly count the number of cycles. Furthermore, the filter weakens the signals with high frequency, which makes the filtered signals keep stable at the range of [-2.5, -1.5] for y-axis and [-1,0] for z-axis.

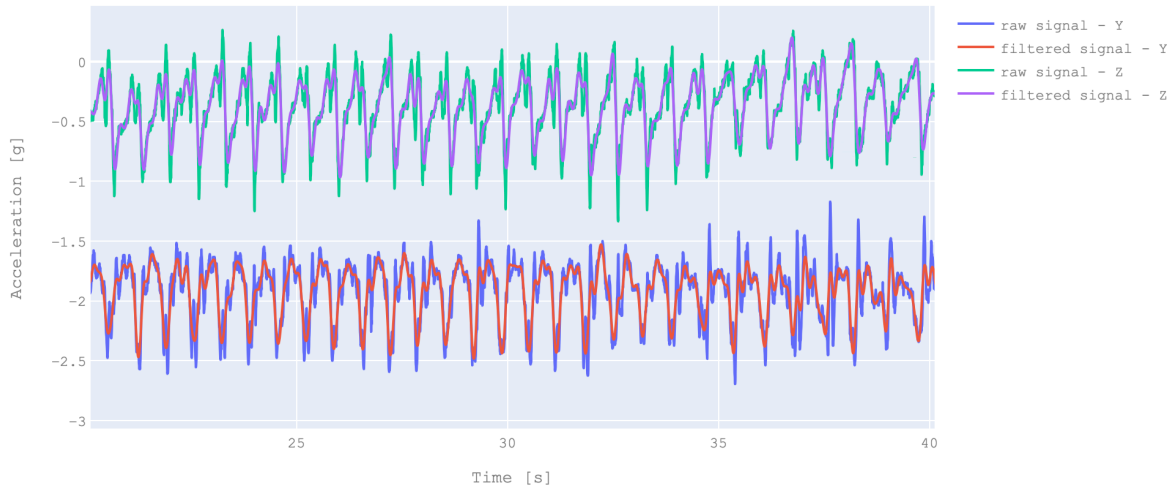


Figure 7 Patterns before and after denoising - Butterworth low pass filter

B. Moving average filter

Figure 8 presents the patterns before and after implementing moving average filter. Parameters also needed to be chosen when using this filter. The parameter is to choose how many examples or how long are we going to take average over. In this figure, we select 15 examples for counting the mean. After implementing moving average filter, the filtered signals look similar to the original signals, but it does delete some noises and make the filtered signal smoother than before. However, those examples with extreme high and low acceleration still remain, which may influence the following classification process.

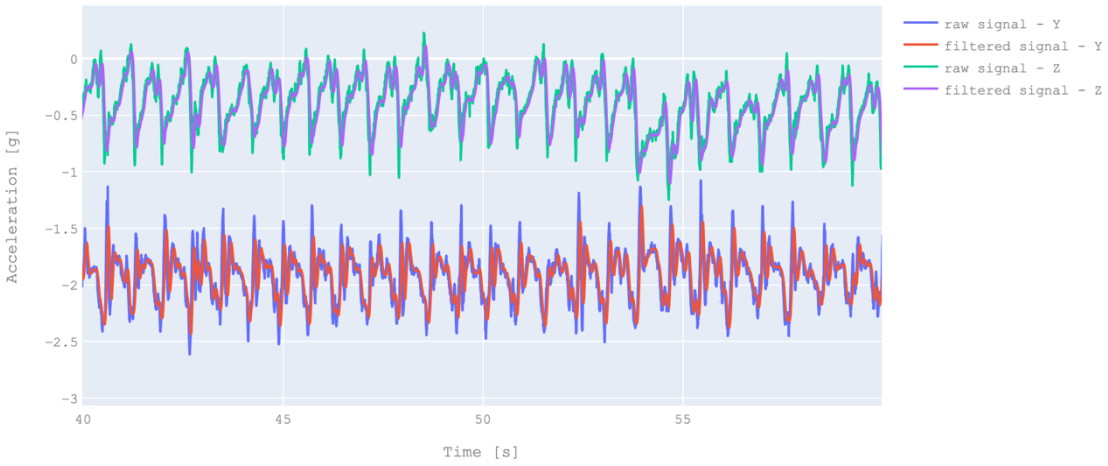


Figure 8 Patterns before and after denoising - Moving average filter

C. Down-sampling

Figure 9 implies the signals before and after using down-sampling. The reason for implementing down-sampling is due to the density of examples for each trial. Since there are 200 examples for each second, there will be 20000 examples for a 100-second trial. Therefore, down-sampling is to reduce the examples for each trial, making the gait cycle easier to determine. The down-sampling technique here is defined to resample 20 examples for every 200 seconds. However, the result of down-sampling is lower than expected. The filtered signal is unstable and much harder to find gait cycles in y-axis. Though gait cycles in z-axis are easier than y-axis to define, there still have some noises in the pattern.

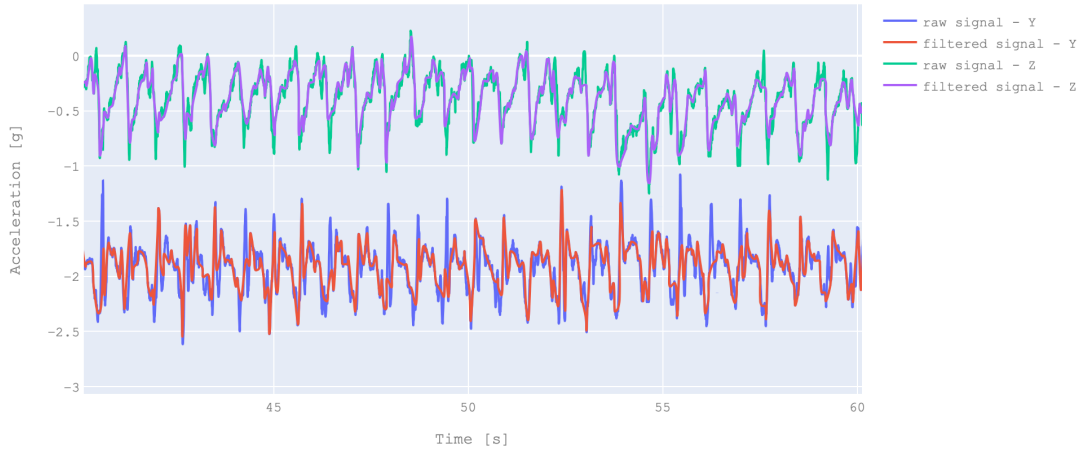


Figure 9 Patterns before and after denoising – Down-sampling

4.3 Comparison

This section is to compare the gait pattern when using different denoising method mentioned above. It is important to find out which kind of method is the best on denoising data since the better it is, the better the accuracy score of the classifier gets. Therefore, the following indicates the best denoising method for this dataset and explains the reason for it.

From the above section, data using Butterworth low pass filter seems to get the best result. It is because the filtered data can clearly define each gait cycle and it has stable amplitude,

which means that the filtered signals have separates most noises from the original signals. The other two methods will not be selected as the denoising technique for this dataset due to its difficulties to define and extract the gait cycle by peak detection. It is hard to find the true peak from the filtered signals which include irregular cycles or unstable amplitude. Details about peak detection will be explained in detail in the next section. As the result, Butterworth low pass filter will be suitable for this dataset.

Chapter 5 Project Design and Implement

This chapter illustrates the steps that construct this project, including data pre-processing, feature extraction, classification and classifier improvement. All the following steps are addressed by Python 3 and its libraries such as Matplotlib, Plotly, Scikit-learn, Scikit-plot and imbalanced-learn.

5.1 Data Pre-processing

This section shows three pre-processing methods before training classifiers, including data denoising, peak detection as well as segmentation and labels given. For each part, it explains the reason for using this method and the results after this process.

5.1.1 Denoising

According to chapter three, we tried three denoising methods and alternatively select one of them. Moving average filter smooth the patterns indeed, but the patterns still cause cycle extraction hard to process. Down-sampling reduces the examples for each trial efficiently, but the gait patterns become irregular which leads to difficulties to extract cycles. Butterworth low pass filter turns out to be the best method for denoising, since it has regular gait cycle and probably be more suitable for working on peak detection. Therefore, the following steps will use the data which was denoised through Butterworth low pass filter.

5.1.2 Peak Detection

Peak detection is a method to extract cycles from gait patterns. The initial plan of implementing peak detection is to select several cycles as a chunk, then using these chunks on feature extraction. In order to find peaks for the gait patterns, the function find_peaks from the library Scipy will be used. This function is to find local maxima by comparing with the neighbour values. The maximum value or the peak is defined as any sample with two direct neighbours that have lower amplitude. The plot on the top of figure 10 and 11 shows the peaks for the activity “walking slowly”. It is clear that not all cycles have a peak and even have two peaks or more in one cycle. Apart from this, the plot at the bottom of figure 10 and 11 add a line to define true peaks of the pattern. The reason for finding true peaks is to extract the cycles correctly (Thang et al. 2012, 344-348), however, there is not any line which can accurately separate true peaks from peaks. Additionally, most activities in SisFall dataset are sit, stand or fall, so there is not enough gait cycle to extract. Therefore, peak detection is not suitable for this dataset, and segmenting time chunks becomes a new decision before implementing feature extraction.

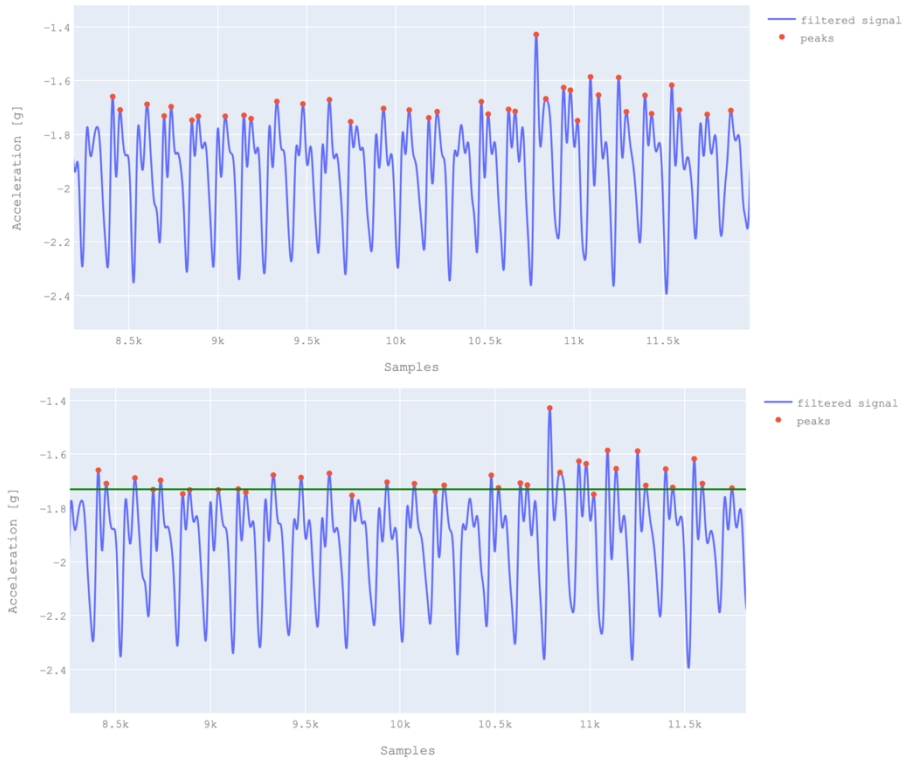


Figure 10 Peak detection plot – y-axis

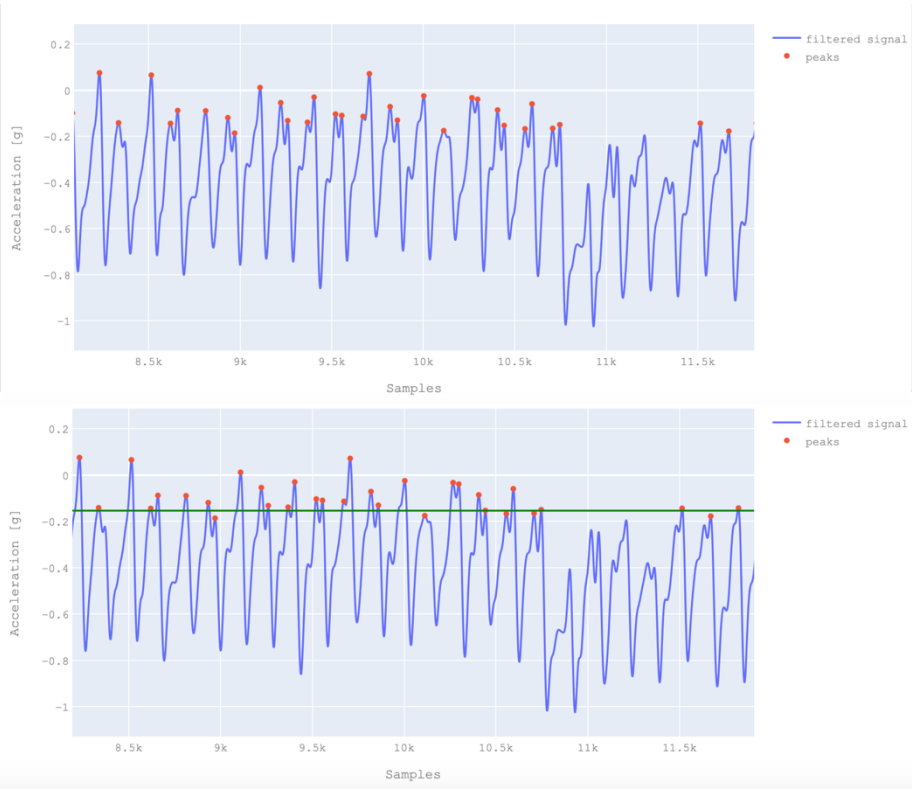


Figure 11 Peak detection plot – z-axis

5.1.3 Segmentation and Labels Given

As the previous section mentioned, the weak point of peak detection leads to a new decision, which is to segment gait patterns every 5 seconds (around 1000 rows) and the label will be given to each chunk at the same time.

Since this dataset does not give any labels for each activity, we decide to give labels on our own. Therefore, in this project, there are several versions of labels given such as binary version, 9 classes version and multi-class version. For the dataset in binary version, the labels are fall and no fall. The number of cases in the class "fall" is 5396 and "no fall" is 11721.

For 9 classes version, one of the labels is "fall" and others are separate from ADLs. The activity code D01, D02, D18 will be given a label "walking", D03 and D04 are labelled as "jogging", D05 and D06 are "walk up and down", from D07 to D10 will be "sit and stand", D11 is "sit and collapsed when standing", D12, D13 and D14 is "lying down", from D15 to D17 will be "stand and bend" and the final one D19 is "jump". After labelling with this concept, the number of cases for each class presents below in figure 12.

```
Number of Cases in Each Class - 9 Classes
-----------------------------------------
fall                                5396
sit and stand                       2250
stand and bend                      2068
walking                             1914
walk up and down                    1545
jogging                             1516
lying down                          1500
sit and collapsed when standing      568
jump                                 360
Name: Class, dtype: int64
```

Figure 12 Number of Cases for 9 classes version

For multiclass version, the labels separate not only ADLs but also falling activities. Falling activities are divided into three classes such as fall forward, fall backward and lateral fall. ADLs are also divided into several classes as 9 classes version did but with slightly different. In multiclass version, apart from previous 8 classes in ADLs, two classes are added, which is "stand and get in car" and "stumble". The following figure shows the number of cases in each class.

```
Number of Cases in Each Class - 13 Classes
------------------------------------------
fall forward                        2514
sit and stand                       2250
lateral fall                        1802
walking                             1554
walk up and down                    1545
jogging                             1516
lying down                          1500
stand and bend                      1130
fall backward                       1080
stand and get in car                 938
sit and collapsed when standing      568
jump                                 360
stumble                              360
Name: Class, dtype: int64
```

Figure 13 Number of Cases for Multiclass Version

5.2 Feature Extraction

Feature extraction is a method that helps reduce dimensionality when the dataset has too many features. In other words, it is to find features that are functions of raw data. There are several techniques for feature extraction, among which PCA and LDA are the most commonly used approaches for reducing dimensionality. Therefore, these two techniques will be used in this project.

After segmenting and labelling the data, we select some commonly used features for extraction such as mean, standard deviation, maximize value and minimize value. Thus, the labelled data will be summarized to the four features mentioned above. The size of each feature for a single sensor will be [1 * 3], since each sensor has 3 axes. There are four features in the labelled data, thus, the size of feature vector will now become [1 * 3 (number of axes) * 4 (number of features)] = [1 * 12] for a single chunk that was extracted from a sensor. Finally, the size of all feature vectors for feature extraction is [17117 rows * 13 columns], and the following figure is an example of all feature vectors.

	meanX	meanY	meanZ	stdX	stdY	stdZ	minX	minY	minZ	maxX	maxY	maxZ	Class
0	0.085602	-2.046910	-0.165432	0.224971	0.240815	0.262214	-0.539383	-2.612986	-0.787136	0.672398	-1.430116	0.255364	1
1	0.064067	-2.042821	-0.198283	0.182148	0.228697	0.268010	-0.357230	-2.586323	-0.675377	0.483438	-1.702898	0.247042	1
2	0.034508	-2.040240	-0.195478	0.180356	0.224229	0.269846	-0.281573	-2.606188	-0.712617	0.509439	-1.702678	0.229507	1
3	0.065178	-2.045608	-0.202303	0.189933	0.245237	0.299422	-0.407757	-2.580269	-0.796416	0.501892	-1.693133	0.357045	1
4	0.092112	-2.049985	-0.196110	0.196736	0.276952	0.281025	-0.387427	-2.732318	-0.729251	0.666451	-1.662035	0.256040	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
17112	-0.918362	-1.170019	-0.178421	0.942848	0.600601	0.837504	-5.480759	-1.798114	-0.972943	0.048145	0.084476	1.843797	9
17113	-1.696379	-0.529891	0.831250	0.009389	0.006787	0.014280	-1.717233	-0.541640	0.796336	-1.679399	-0.513499	0.857291	9
17114	0.081148	-1.809066	-0.694367	0.006606	0.015856	0.039572	0.065494	-1.830724	-0.769370	0.094392	-1.768264	-0.641940	9
17115	-0.402707	-1.513808	-0.473482	0.841468	0.530855	0.724804	-5.469134	-1.809878	-0.875560	0.184359	0.118549	2.080549	9
17116	-1.584445	-0.556138	0.994940	0.032885	0.032622	0.046630	-1.758578	-0.678292	0.786969	-1.522205	-0.457036	1.046977	9

17117 rows × 13 columns

Figure 14 Example of All Feature Vectors

When implementing feature extraction techniques, it is important to find out how many principal components to choose. To find the suitable number of principal components, explained variation ratios will be checked, for it implies how much information will be held in each number of principal components. For this study, the first four principal components hold 84.9% of information when implementing PCA, while for LDA, three principal components hold 91.1% of information. Consequently, the dimension of the features reduced from 12 to 4 in PCA and from 12 to 3 in LDA.

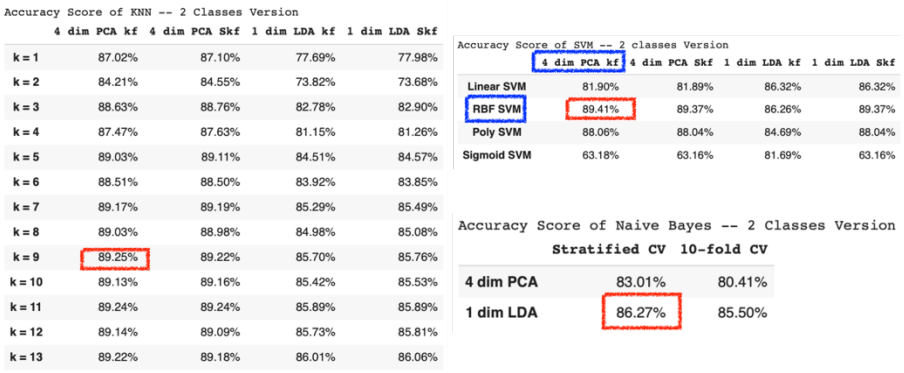
5.3 Model Selection

Model selection is the process of selecting one of many candidate machine learning models for predicting modelling problems. In this project, machine learning algorithms such as SVM, KNN and NB will be determined that which one is the final model. In order to find out which model is the best, cross-validation will be used. It is a big family which include several different methods, for example, k-fold cross-validation, leave one out cross-validation (LOOCV) and stratified k-fold cross-validation. The most widely used method is k-fold cross-validation, which splits the training dataset into k folds, and each example only appears once in the test set. Another is LOOCV, where the test set consists of one sample and each sample has an opportunity to become a test set. In addition, N, the number of samples in the

training set, models need to be constructed and evaluated. Stratified k-fold cross-validation is a variation of k-fold, where each set includes approximately the same percentage of samples of each target class. Since our dataset has 17117 rows, LOOCV will be a time-consuming method, hence, 10-fold cross-validation and stratified 10-fold cross-validation will be chosen as the technique of model selection.

The following part shows the results of model selection for each version of dataset and the comparison of each model.

For binary version of the dataset, when using KNN as the model, the accuracy is 89.25% when k equals to 9, using the dataset created by PCA with 4 dimensions and implementing 10-fold cross-validation. As for SVM, the dataset and the cross-validation method which got the best accuracy are the same as KNN, an RBF SVM got the best accuracy in binary version, which is 89.41%. While for NB, dataset from LDA with 1 dimension got the best accuracy. Compare with all three models, SVM had the highest accuracy, therefore, for 2 classes dataset, the best model will be RBF SVM. The accuracy score will be shown as follow:



Accuracy Score of KNN -- 2 Classes Version

	4 dim PCA kf	4 dim PCA Skf	1 dim LDA kf	1 dim LDA Skf
k = 1	87.02%	87.10%	77.69%	77.98%
k = 2	84.21%	84.55%	73.82%	73.68%
k = 3	88.63%	88.76%	82.78%	82.90%
k = 4	87.47%	87.63%	81.15%	81.26%
k = 5	89.03%	89.11%	84.51%	84.57%
k = 6	88.51%	88.50%	83.92%	83.85%
k = 7	89.17%	89.19%	85.29%	85.49%
k = 8	89.03%	88.98%	84.98%	85.08%
k = 9	89.25%	89.22%	85.70%	85.76%
k = 10	89.13%	89.16%	85.42%	85.53%
k = 11	89.24%	89.24%	85.89%	85.89%
k = 12	89.14%	89.09%	85.73%	85.81%
k = 13	89.22%	89.18%	86.01%	86.06%

Accuracy Score of SVM -- 2 classes Version

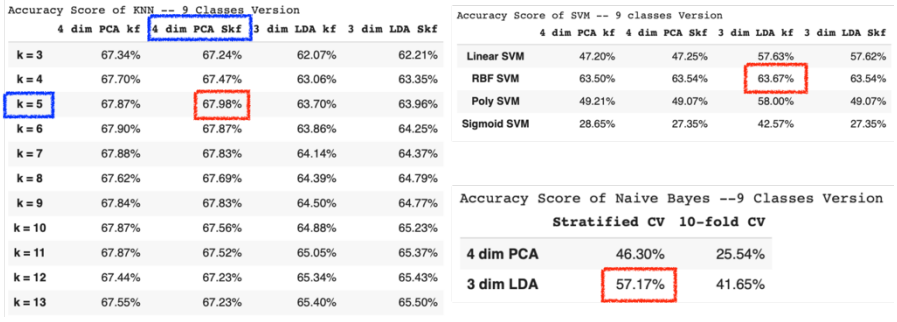
	4 dim PCA kf	4 dim PCA Skf	1 dim LDA kf	1 dim LDA Skf
Linear SVM	81.90%	81.89%	86.32%	86.32%
RBF SVM	89.41%	89.37%	86.26%	89.37%
Poly SVM	88.06%	88.04%	84.69%	88.04%
Sigmoid SVM	63.18%	63.16%	81.69%	63.16%

Accuracy Score of Naive Bayes -- 2 Classes Version

	Stratified CV	10-fold CV
4 dim PCA	83.01%	80.41%
1 dim LDA	86.27%	85.50%

Figure 15 Accuracy Score for 3 models - 2 Classes Version

For 9 classes version, the best accuracy score in SVM is 63.67% when implementing RBF SVM, 10-fold cross-validation and using 3 dimensions dataset generated by LDA. While for KNN, when k equals to 5, the accuracy score is 67.98%, using 4 dimensions dataset from PCA and stratified 10-fold cross-validation. As for NB, the best score is 57.17%. Among these three models, KNN got the best result.



Accuracy Score of KNN -- 9 Classes Version

	4 dim PCA kf	4 dim PCA Skf	3 dim LDA kf	3 dim LDA Skf
k = 3	67.34%	67.24%	62.07%	62.21%
k = 4	67.70%	67.47%	63.06%	63.35%
k = 5	67.87%	67.98%	63.70%	63.96%
k = 6	67.90%	67.87%	63.86%	64.25%
k = 7	67.88%	67.83%	64.14%	64.37%
k = 8	67.62%	67.69%	64.39%	64.79%
k = 9	67.84%	67.83%	64.50%	64.77%
k = 10	67.87%	67.56%	64.88%	65.23%
k = 11	67.87%	67.52%	65.05%	65.37%
k = 12	67.44%	67.23%	65.34%	65.43%
k = 13	67.55%	67.23%	65.40%	65.50%

Accuracy Score of SVM -- 9 classes Version

	4 dim PCA kf	4 dim PCA Skf	3 dim LDA kf	3 dim LDA Skf
Linear SVM	47.20%	47.25%	57.63%	57.62%
RBF SVM	63.50%	63.54%	63.67%	63.54%
Poly SVM	49.21%	49.07%	58.00%	49.07%
Sigmoid SVM	28.65%	27.35%	42.57%	27.35%

Accuracy Score of Naive Bayes --9 Classes Version

	Stratified CV	10-fold CV
4 dim PCA	46.30%	25.54%
3 dim LDA	57.17%	41.65%

Figure 16 Accuracy Score for 3 models - 9 Classes

For 13 classes version, the accuracy score of NB is the lowest, while KNN again gets the highest score, which is 60.5% when k = 5, using 4 dimensions dataset created from PCA and stratified 10-fold cross-validation.

Accuracy Score of KNN -- 13 Classes Version

	4 dim PCA kf	4 dim PCA Skf	3 dim LDA kf	3 dim LDA Skf
k = 3	58.85%	58.81%	50.32%	50.31%
k = 4	59.61%	59.78%	52.47%	52.50%
k = 5	60.20%	60.50%	53.08%	53.14%
k = 6	60.24%	60.13%	54.12%	54.22%
k = 7	60.49%	60.41%	54.57%	54.49%
k = 8	60.47%	60.19%	55.07%	54.89%
k = 9	60.27%	60.22%	55.19%	55.33%
k = 10	60.40%	60.02%	55.42%	55.43%
k = 11	60.07%	60.09%	55.61%	55.73%
k = 12	59.91%	59.98%	55.94%	55.92%
k = 13	59.93%	60.02%	55.93%	56.22%

Accuracy Score of SVM -- 13 classes Version

	4 dim PCA kf	4 dim PCA Skf	3 dim LDA kf	3 dim LDA Skf
Linear SVM	42.24%	42.23%	49.68%	49.73%
RBF SVM	56.52%	56.71%	55.36%	56.71%
Poly SVM	46.04%	46.07%	49.03%	46.07%
Sigmoid SVM	20.77%	20.39%	28.30%	20.39%

Accuracy Score of Naive Bayes -- 13 Classes Version

	Stratified CV	10-fold CV
4 dim PCA	43.03%	19.02%
3 dim LDA	47.22%	32.44%

Figure 17 Accuracy score for 3 models - 13 Classes

From the above figures for model selection, RBF SVM is the best model for binary version of dataset and KNN is the best model for 9 and 13 classes version of dataset when K equals to 5.

5.4 Result Measurement

Model evaluation is a process to explain whether the specific model has good performance or not. There are several evaluation tools that can be used for classification models, including confusion matrix, Cohen's Kappa, precision, recall, specificity, accuracy, F1 measure and ROC curve.

This project focus on the classification result of different versions of dataset, though the accuracy is a straightforward measurement and it looks good in all experiments, it cannot be the only measurement since it is not comprehensive enough. Therefore, confusion matrix, sensitivity and specificity will be used to make the result more reliable and the definition will be explained as follow.

5.4.1 Confusion Matrix

Confusion matrix is commonly used in supervised learning. Each row of matrix represents the predicted class of the instances, and each column represents the actual class of the instances. This matrix easily shows whether the model confusing two classes with each other, which means misclassified one as another. In addition, there are four basic terms to describe which are:

True positives (TP): this is the case that instances in actual class are yes and also predicted as yes, which is 45 in figure 15.

True negatives (TN): this is the case that instances in actual class are no and also predicted as no, which is 22 in figure 15.

False positive (FP): this is the case that instances in actual class are no but wrongly predicted as yes, which is 35 in figure 15.

False negative (FN): this is the case that instances in actual class are yes but wrongly predicted as no, which is 18 in figure 15.

N = 120			
		predicted class	
		yes	no
Actual Class	yes	45	18
	no	35	22

Figure 18 Example of Confusion Matrix

5.4.2. Accuracy

Accuracy, which means the percentage of correctly predicted instances over all instances.

$$(TP + TN) / Total = (45 + 22) / 120 = 0.56 \quad (4.1)$$

Error rate is the percentage of wrongly predicted instances over all instances.

$$(FP + FN) / Total = (35 + 18) / 120 = 0.44 \quad (4.2)$$

5.4.3 Specificity

Specificity is also known as true negative rate, which is to calculate the proportion of actual negatives over all false instances.

$$TN / (FP + TN) = 22 / (35 + 22) = 0.39 \quad (4.3)$$

5.4.4 Sensitivity

Sensitivity is known as recall and true positive rate, which is to measure the proportion of actual positives over all true instances.

$$TP / (TP + FN) = 45 / (45 + 18) = 0.71 \quad (4.4)$$

5.5 Classifier Improvement

Since the dataset in each version has imbalanced classes and it may affect the accuracy of the classifier, thus, one method to improve the classifier is data balancing. Several data balancing methods have explained in detail in section 3.3.

In this project, the data balancing method will focus on under-sampling. For binary and 9-class version of dataset, their balancing method is to randomly select the classes and resample them. While for multiclass version, the balancing method will be selecting the specific class and rebalance them, which means to identify which class has a lot of misclassification and check the number of examples. If the cases are much less than others, then rebalance them.

Chapter 6 Experiment Results and Analysis

This section lists the results acquired from all experiments including confusion matrix, accuracy score, sensitivity and specificity.

6.1 Nine Classes Version

In section 5.3, it shows that KNN has the best accuracy in 9 classes version of dataset. When k equals to 5, the accuracy score is 67.98%. Though this score is good in such many classes, confusion matrix has to be checked if there is any reason that influences the accuracy score. Since the training method is stratified 10-fold cross-validation, it will run for 10 trials in different training sets, therefore it has 10 confusion metrics. The following figure is one of the confusion metrics of this classifier, which is the closest accuracy score to the overall accuracy 57.89%. From this confusion matrix, the class “fall” has the most cases to be correctly classified, while the class “jump” and “sit and collapsed when standing” is the less. From the specificity of each class, it represents that this classifier is less misjudging different categories. As for the sensitivity of the classes “jump”, “sit and collapsed when standing”, “sit and stand”, “stand and bend” and “walk up and down”, the sensitivity of these classes are lower than 50%, which implies that these classes have lower chances to classify correctly. In addition, several activities such as “stand and bend”, “walk up and down” and “sit and collapsed when standing” are misclassified as “sit and stand” and the class “jump” is misclassified as “jogging”. One of the reasons is that the gait patterns of these activities are similar, another may because some activities have fewer samples which lead to imbalanced classes.

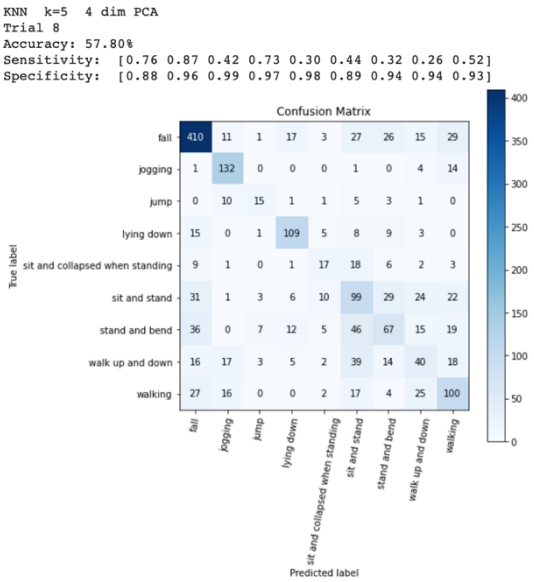


Figure 19 Metrics of the Best Classifier in 9 Classes Version

To solve the problem of imbalanced dataset, under-sampling techniques will be implemented. Figure 20 represents the results after implementing randomly under-sampling and over sampling methods such as reduce the majority class examples, T-Link, ENN and SMOTE + T-Link. It is clear that No.1 in figure 20 is the worst one, since it not only has the lowest accuracy score, but also misclassifies many categories which were not misclassified before

implementing under-sampling. ENN is also not a good modify approach which has the same reason as reducing the examples of majority class to the minority class. T-Link (2) and SMOTE + T-Link (4) get better accuracy score, however, No. 4 is better than No. 2. SMOTE + T-Link gets the best accuracy and most categories classified correctly. To compare with the original classifier, though the accuracy score is a bit lower, the sensitivity for each category is higher than the original one, and the confusion matrix presents that misclassification case decreased.

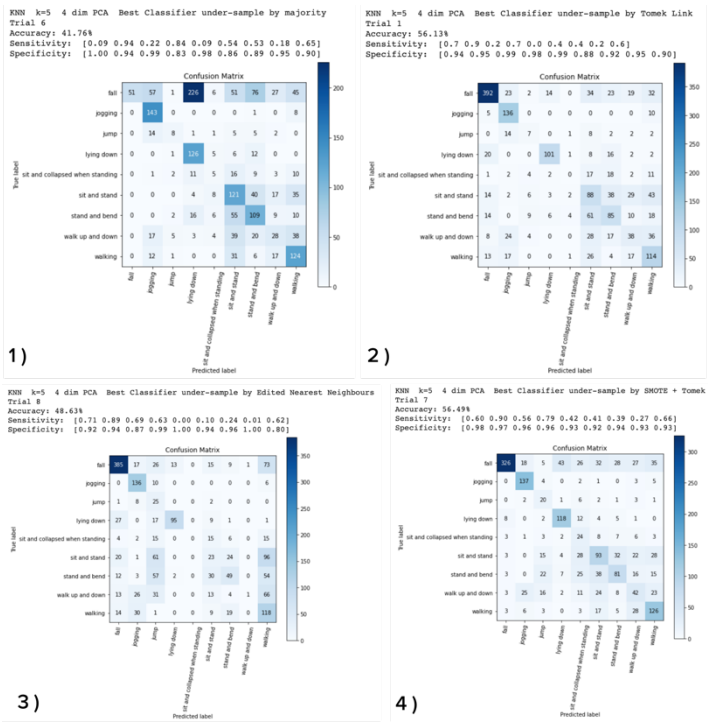


Figure 20 Metrics of Randomly Under-sampling

Another method to improve the classifier is to under-sample the selected categories, thus one-sided selection will be used. The following figure includes 4 different sampling strategies. The difference between these strategies is to select different categories to resample. The standard of selecting resampling categories is based on the original classifier, in other word, categories are selected due to the degree of misclassification in the original classifier. Therefore, the classes “stand and bend”, “walk up and down”, “sit and collapsed when standing” and “sit and stand” are the fourth basic categories to select due to the high misclassification. Other categories such as “fall”, “jump” and “jogging” will be selected as well. The results for these 4 strategies have shown in figure 21 and the best one is located at the top left which chose one additional category “fall” to resample. The overall accuracy score for this strategy is similar to the original one. However, compared with the original classifier and the randomly under-sampling classifier, the latter is the best. Because the strategy here causes more misclassification and the sensitivity for some categories are lower.

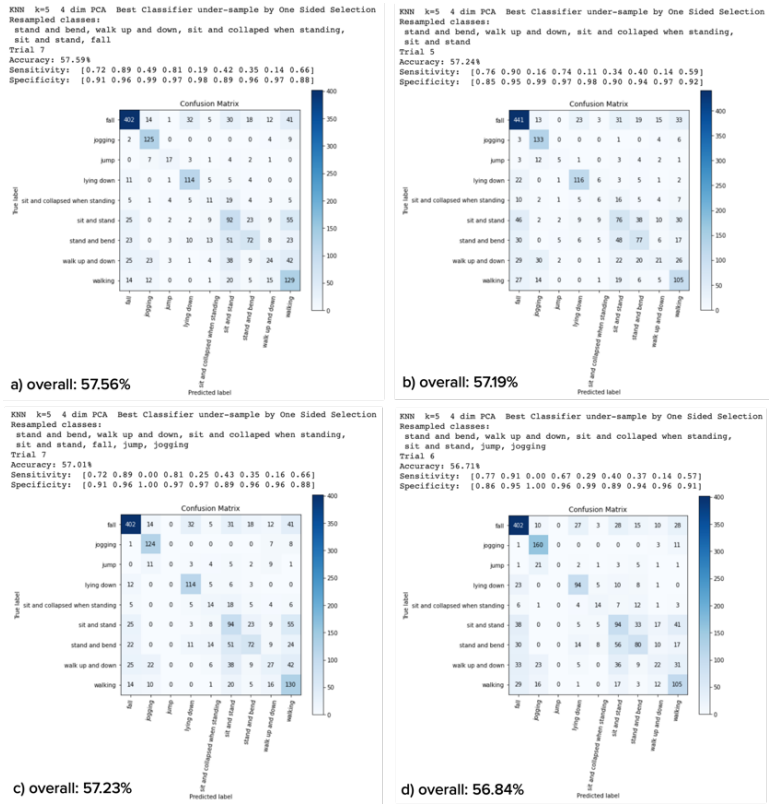


Figure 21 Metrics of One-sided Selection

6.2 Binary Version

RBF SVM is the best classifier when classifying binary version of dataset. In section 5.3, the accuracy score of RBF SVM is 89.41%. Similar as nine classes version, the accuracy score is quite high, which means that this classifier can correctly classify “fall” and “no-fall”. Since the training method is 10-fold cross-validation, it includes 10 trials of training. The following figure is some metrics for this classifier when training by 10-fold cross-validation. The confusion matrix implies that the class “no-fall” can be classified correctly in most cases, but the class “fall” sometimes misclassified as “no-fall”. To compare with the numbers of the falls that are correctly classified, the number in the bulk of predicting fall as no-fall is large. This suggests that the classifier have a bias towards predicting no-fall in all cases, because there is more “no-fall” than “fall” examples and it causes the classifier does well when saying “no-fall” most of the time.

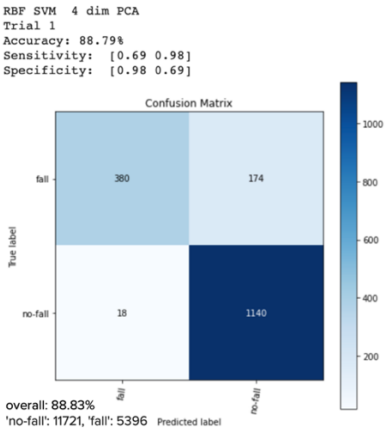


Figure 22 Metrics of the Best Classifier in Binary Version

According to the above problem, the method to improve this classifier is to balance the dataset. The under-sampling techniques used in section 5.1 will be implemented here and check whether these methods have a good influence on this classifier. The accuracy score for all these methods are also high, therefore other metrics have to be considered. The results show that T-Link and OSS have no obvious improvement because the results are almost the same as the classifier without under-sampling. For the other fourth experiments, reducing the samples from the majority class (2) and SMOTE + T-Link (5) seem to have better improvement. Though both of their accuracy scores are not higher than the original classifier, their sensitivity and specificity have some improvement. It is also clear that the number of misclassifications of fall as no-fall decreased, and the number of correctly classifying the class “fall” increased. This indicates that the bias towards predicting no-fall has reduced. In addition, the misclassification number of no-fall as fall increased, but compared to the number of correct prediction, it is a small number.

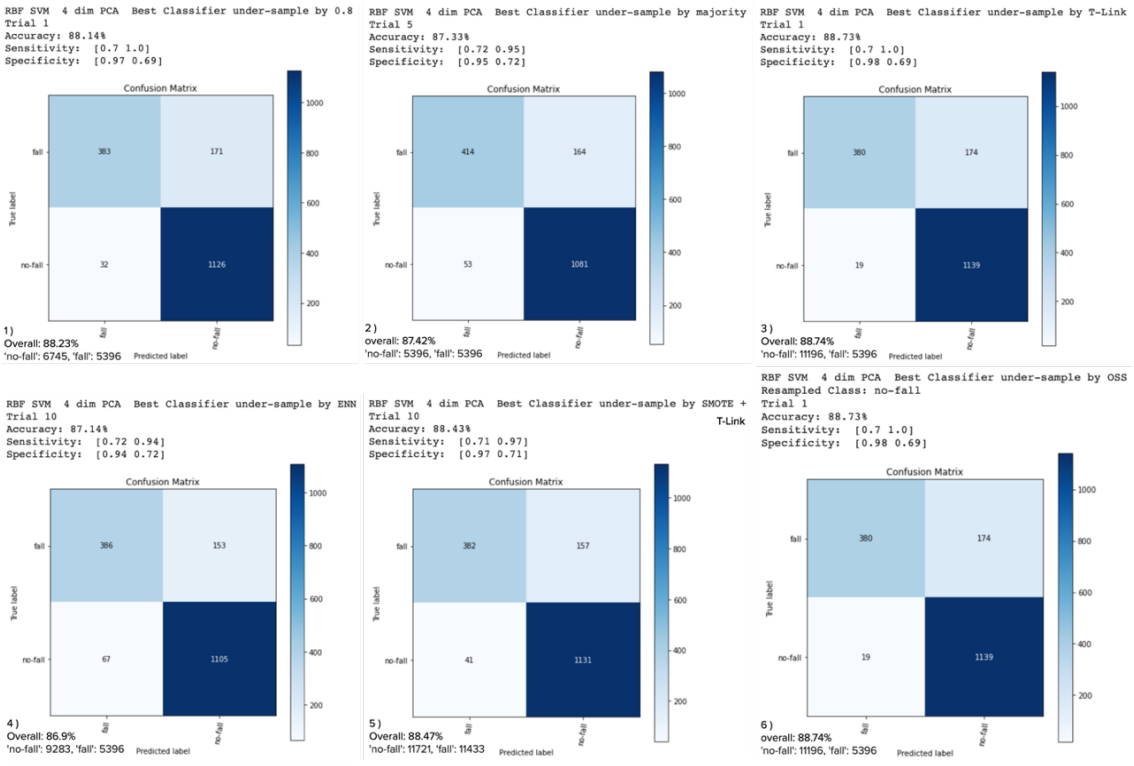


Figure 23 Metrics of Under-sampling

The improvement for the binary version of dataset seems to be not significant, and the reason may due to the process of labels given. The labels of binary version are based on the public dataset used in this project. The activities that belong to ADLs will be given a label “no-fall”, while other activities that belong to Falls will be given a label “fall”. Therefore, this labelling process is not specific enough to have good classification results when trying to improve the original classifier which already has a good result.

6.3 Multiclass Version

Section 5.3 indicates that the best classifier in multiclass version is KNN when K equals to 5, and the accuracy score of this classifier is 60.5%. It is a good accuracy score for a 13 classes classifier; however, other metrics need to be considered as well to make the results more comprehensive. Therefore, classifier in this version also implements stratified 10-fold cross-validation and includes not only accuracy score, but also sensitivity, specificity and confusion matrix. The following figure shows the metrics about the best classifier in 13 classes version and the number of cases in each class. Apparently, the classes “fall forward”, “jogging”, “lateral fall”, “lying down”, “sit and stand” and “walking” have classified correctly in most cases. However, some classes are highly misclassified with other classes, for example, the class “fall forward” and “lateral fall” will wrongly be classified with each other and the classes “walk up and down”, “stand and bend”, “stand and get in car”, and “sit and collapsed when standing” often misclassified as “sit and stand”. In addition, the class “stumble” cannot be classified correctly, all of them are misclassified as another class. These situations may be caused by similar patterns among each class and the imbalanced classes in this dataset. Figure 24 also shows the number of cases in each class, and it is obvious that the classes “stumble”, “jump” and “sit and collapsed when standing” are much less than other classes.

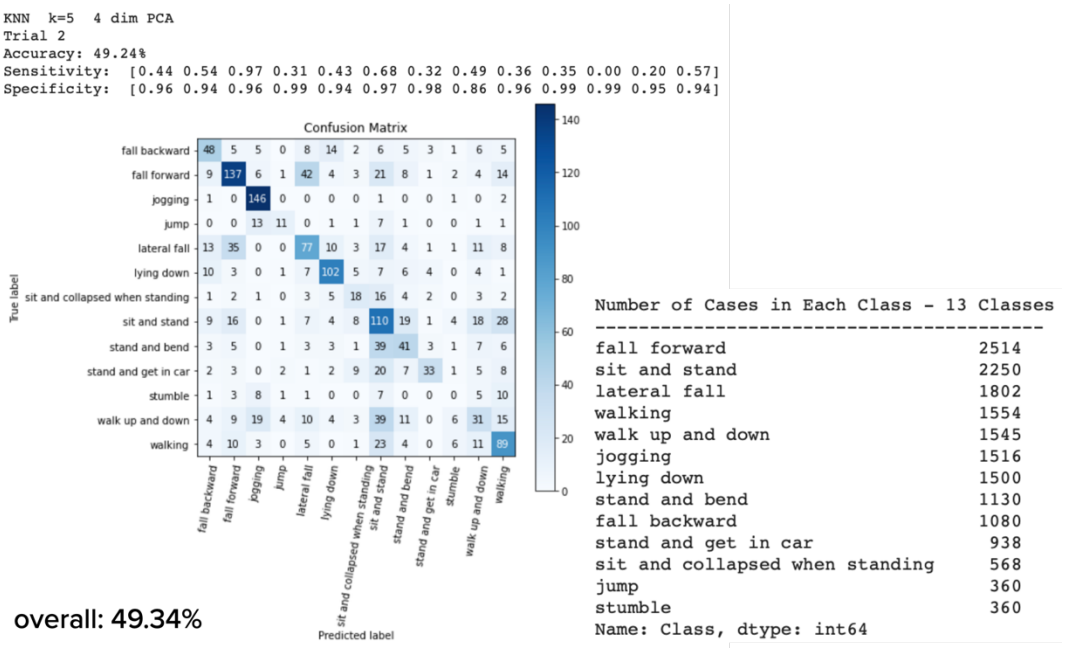


Figure 24 Metrics of the Best Classifier and the Number of Cases

According to the above reasons, under-sampling techniques will be implemented here. Unlike the methods used in section 6.1 and 6.2, the approach of under-sampling the selected classes, including OSS, CNN, ENN and T-Link, will be used in 13 classes version of dataset. Before

implementing under-sampling to the classifier, it is important to identify the places where there are a lot of misclassifications and check whether the number of examples is much less than other cases. As the previous paragraph indicates, several classes such as “jump”, “sit and collapsed when standing”, “walk up and down”, “stand and bend”, “stand and get in car”, and “stumble” have plenty of misclassification and some of their cases are much less than other classes. Consequently, resampling the classes in this dataset is the priority to improve this classifier. Since there are plenty of methods and strategies to rebalance the dataset, the following table will show the overall accuracy of each method before selecting the classes.

Method	Accuracy
One-Sided Selection (OSS)	49.23%
Condensed Nearest Neighbour (CNN)	29.96%
Edited Nearest Neighbour (ENN)	40.15%
AllKNN	46.98%
Tomek Link (T-Link)	48.54%

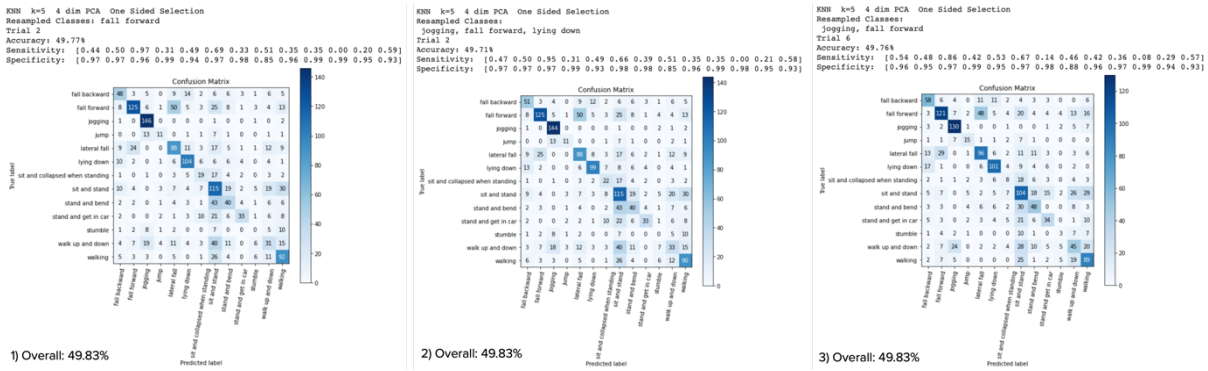
Table 2 Overall Accuracy of Each Method

From the above table, it indicates that OSS and T-Link have better accuracy on improving the classifier, therefore, the following table will present the results of these two methods when selecting classes to resample. Regarding the choice of the resample classes, the standard is to select the classes which have more cases correctly classified. Thus, the classes jogging, fall forward, lying down, sit and stand, walking and lateral fall are choosing as the resampling classes. With these classes, OSS and T-Link are implementing to get accuracy from these categories. Table 3 presents the accuracy when using different resampling classes, and the best one in OSS is 49.83%, which only selects the class “fall forward”. The best accuracy score in T-Link is 49.72% with selecting the classes “jogging” and “fall forward”. After knowing the best result in two different methods, comparing the metrics with the original classifier will be the final step.

Method	Selected Classes	Accuracy
OSS	jogging, fall forward, lying down, sit and stand, walking, lateral fall	48.98%
	jogging, fall forward, lying down, sit and stand	49.29%
	jogging, fall forward, lying down	**49.83%**
	jogging, fall forward	**49.83%**
	jogging, lying down	49.34%
	lying down, fall forward	49.83%
	fall forward	**49.83%**
	lying down	49.34%
	jogging	49.34%
T-Link	jogging, fall forward, lying down, sit and stand, walking, lateral fall	48.41%
	jogging, fall forward, lying down, sit and stand	49.04%
	jogging, fall forward, lying down	49.69%
	jogging, fall forward	**49.72%**
	jogging, lying down	49.26%
	lying down, fall forward	49.67%
	fall forward	49.71%
	lying down	49.26%
	jogging	49.26%

Table 3 Accuracy Score of Using Different Classes

The following figure shows the metrics of OSS which gets 49.83% on overall accuracy score. Those with bold texts in the table under these three confusion metrics imply that the cases have reduced after implementing under-sampling. In terms of the results for these three different strategies, the first two seems to have similar results since the sensitivity, specificity and confusion matrix are quite the same. In addition, most misclassification categories still remain similar cases as the original classifier, which represents that these two classifiers do not have obvious improvement. As for the third one, the sensitivity and specificity score are better than the other two. Furthermore, misclassification cases decreased when compared with the original classifier. Even though the cases that are correctly classified had decreased, the classifier which resampled the classes “fall forward” and “jogging” has improved.



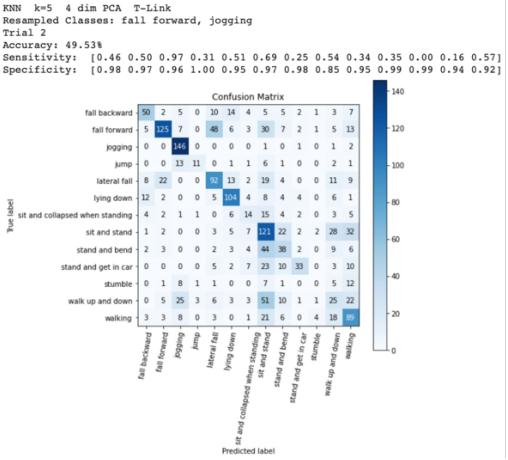
Number of cases in each class	
Sit and stand	2250
Fall forward	**2026**
Lateral fall	1802
Walking	1554
Walk up and down	1545
Jogging	1516
Lying down	1500
Stand and bend	1130
Fall backward	1080
Stand and get in car	938
Sit and collapsed when standing	568
Jump	360
Stumble	360

Number of cases in each class	
Sit and stand	2250
Fall forward	**2026**
Lateral fall	1802
Walking	1554
Walk up and down	1545
Jogging	**1432**
Lying down	**1333**
Stand and bend	1130
Fall backward	1080
Stand and get in car	938
Sit and collapsed when standing	568
Jump	360
Stumble	360

Number of cases in each class	
Sit and stand	2250
Fall forward	**2026**
Lateral fall	1802
Walking	1554
Walk up and down	1545
Jogging	**1516**
Lying down	1500
Stand and bend	1130
Fall backward	1080
Stand and get in car	938
Sit and collapsed when standing	568
Jump	360
Stumble	360

Figure 25 Results of OSS When Resample Different Classes

Figure 26 represents the result of T-Link when resampling the classes “fall forward” and “jogging”. To compare with the original classifier, some categories remain the same sensitivity and specificity score, some of them are lower and few of them become higher. This classifier does not seem to be better than the original one according to the confusion matrix since the misclassification cases increased in several categories.



Number of cases in each class	
Sit and stand	2250
Fall forward	**2026**
Lateral fall	1802
Walking	1554
Walk up and down	1545
Lying down	1500
Jogging	**1432**
Stand and bend	1130
Fall backward	1080
Stand and get in car	938
Sit and collapsed when standing	568
Jump	360
Stumble	360

Figure 26 Results of T-Link When Resample Different Classes

To conclude the best improving classifier for 13 classes version of dataset, OSS with the classes “fall forward” and “jogging” resampled is the one that has some progress, while other strategies are either regressive or only minor improvements.

6.3 Comparison

In these three versions of dataset, they all improve the original classifier to a varying degree and each of them is suitable for distinct methods. For the dataset with 9 classes, the technique, SMOTE + T-Link, with both randomly oversampling and under-sampling is useful. In binary version, most improving techniques do not have a significant effect. Only randomly under-sample the majority class and SMOTE + T-Link have minor progress. Finally, the 13 classes version of dataset has some improvements when under-sampling the selected classes. OSS seems to have better improvement among all other methods when under-sampling the classes “fall forward” and “jogging”.

After implementing these data balancing techniques, all of their original classifiers have improved performance on reducing the misclassification cases. Therefore, this represents that the size of the different classes of examples is a crucial problem in this classification experiment.

Chapter 7 Conclusion

7.1 Overview

Fall detection plays an important role in the ageing society nowadays, since it can be a severe condition when people fall, especially for elders. Therefore, fall-related technologies such as fall detection and fall prevention rise accordingly. Most studies concentrate on fall detection currently, which is to create a real-time based system and detect falls accurately, while the research on fall prevention is still preliminary.

In this thesis, we tried a variety of machine learning algorithms (SVM, KNN, and NB) and compared their performance on classifying different gait patterns. Moreover, we considered three versions of labels given to the dataset such as binary, nine classes, and 13 classes. We also applied under-sampling methods to improve the performance of the classifiers. Our results show that KNN works better in most cases and achieve high accuracy. SMOTE + T-Link and OSS lead to minor progress for each version of dataset, which reduces the misclassification cases for most categories.

In conclusion, KNN can be a method to apply on a fall detection system since it has good performance on classifying activities. Also, when the data includes imbalanced labels, it would be a good choice for implementing SMOTE + T-Link or OSS depending on how many classes are there.

7.2 Future Work

Although the performance of the classification in this project shows well, there are still some limitations in work. There are several aspects to work on in the future.

1. In the experiment work, we have tested which machine learning algorithm is better when classifying different activities and improve its classifier by modifying the cases in each category. However, we have not focus on how to separate those similar gait patterns. In the future work, the method to define the pattern of each activity can be discussed.
2. Most fall detection systems are implementing in a real-time based environment to detect fall immediately and accurately when fall happens, and this helps reduce the pain of getting injuries. However, this work focus on classifying the dataset that was designed as experiments, which sometimes may lead to distortion of the activities. Therefore, in the future work, classification on the real-time based environment can be considered.
3. Most studies nowadays are regarding with fall detection, so we have plenty of methods for detecting fall. Fall prevention is still in a preliminary stage, therefore, finding strategies for fall prevention can be a large scope to discuss.

It is clear that there are several tasks to work on, and each of them can be studied respectively.

References

Two Modifications of CNN 1976. Vol. SMC-6. doi:10.1109/TSMC.1976.4309452.

Axer, Hubertus, Martina Axer, Heinrich Sauer, Otto W. Witte, and Georg Hagemann. 2010. "Falls and Gait Disorders in Geriatric Neurology." *Clinical Neurology and Neurosurgery* 112 (4): 265-274.

Aziz, Omar, Magnus Musngi, Edward J. Park, Greg Mori, and Stephen N. Robinovitch. 2017. "A Comparison of Accuracy of Fall Detection Algorithms (Threshold-Based Vs. Machine Learning) using Waist-Mounted Tri-Axial Accelerometer Signals from a Comprehensive Set of Falls and Non-Fall Trials." *Medical & Biological Engineering & Computing* 55 (1): 45-55.

Babu, C. Narendra and B. Eswara Reddy. 2014. "A Moving-Average Filter Based Hybrid ARIMA–ANN Model for Forecasting Time Series Data." *Applied Soft Computing* 23: 27-38. doi:https://doi.org/10.1016/j.asoc.2014.05.028. http://www.sciencedirect.com/science/article/pii/S1568494614002555.

Batista, Gustavo EAPA, Ronaldo C. Prati, and Maria Carolina Monard. 2004. "A Study of the Behavior of several Methods for Balancing Machine Learning Training Data." *ACM SIGKDD Explorations Newsletter* 6 (1): 20-29.

D. L. Wilson. 1972. *Asymptotic Properties of Nearest Neighbor Rules using Edited Data*. Vol. SMC-2. doi:10.1109/TSMC.1972.4309137.

Fernández, Alberto, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, and Francisco Herrera. 2018. *Learning from Imbalanced Data Sets* Springer.

He, Haibo and Yunqian Ma. 2013. *Imbalanced Learning: Foundations, Algorithms, and Applications* John Wiley & Sons.

Hussain, Faisal, Fawad Hussain, Muhammad Ehatisham-ul-Haq, and Muhammad Awais Azam. 2019. "Activity-Aware Fall Detection and Recognition Based on Wearable Sensors." *IEEE Sensors Journal* 19 (12): 4528-4536.

Kale, Amir, Naresh Cuntoor, B. Yegnanarayana, A. N. Rajagopalan, and Rama Chellappa. 2003. "Gait Analysis for Human Identification."Springer, .

Muntakim, Abu Hena. 2017. "No Title." Acoustic Noise Characterization for Leak Detection in Water Mains.

P. Hart. 1968. *The Condensed Nearest Neighbor Rule (Corresp.)*. Vol. 14. doi:10.1109/TIT.1968.1054155.

Perell, Karen L., Audrey Nelson, Ronald L. Goldman, Stephen L. Luther, Nicole Prieto-Lewis, and Laurence Z. Rubenstein. 2001. "Fall Risk Assessment Measures: An Analytic Review." *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences* 56 (12): M761-M766.

Sucerquia, Angela, José David López, and Jesús Francisco Vargas-Bonilla. 2017. "SisFall: A Fall and Movement Dataset." *Sensors* 17 (1): 198.

Tahir, Muhammad Atif, Josef Kittler, and Fei Yan. 2012. "Inverse Random Under Sampling for Class Imbalance Problem and its Application to Multi-Label Classification." *Pattern Recognition* 45 (10): 3738-3750. doi:https://doi.org/10.1016/j.patcog.2012.03.014. http://www.sciencedirect.com/science/article/pii/S0031320312001471.

Tao, Weijun, Tao Liu, Rencheng Zheng, and Hutian Feng. 2012. "Gait Analysis using Wearable Sensors." *Sensors* 12 (2): 2255-2283.

Thang, Hoang Minh, Vo Quang Viet, Nguyen Dinh Thuc, and Deokjai Choi. 2012. "Gait Identification using Accelerometer on Mobile Phone."IEEE, .

Vallabh, Pranesh, Reza Malekian, Ning Ye, and Dijana Capeska Bogatinoska. 2016. "Fall Detection using Machine Learning Algorithms."IEEE, .