School of Computer Science and Informatics

Coursework Submission Cover Sheet

Please use Adobe Reader to complete this form. Other applications may cause incompatibility issues.

Student Number	1514995
Module Code	CM3203
Submission Date	10/05/19
Hours spent on this exercise	400

Special Provision

(Please place an x in the box above if you have provided appropriate evidence of need to the Disability & Dyslexia Service and have requested this adjustment).

Group Submission

For group submissions, each member of the group must submit a copy of the coversheet. Please include the student number of the group member tasked with submitting the assignment.

Student number of submitting group member

By submitting this cover sheet you are confirming that the submission has been checked, and that the submitted files are final and complete.

Declaration

By submitting this cover sheet you are accepting the terms of the following declaration.

I hereby declare that the attached submission (or my contribution to it in the case of group submissions) is all my own work, that it has not previously been submitted for assessment and that I have not knowingly allowed it to be copied by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another writer, as one's own is plagiarism. I also understand that plagiarising another's work or knowingly allowing another student to plagiarise from my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings.

ACTIVITY RECOGNITION ON BODY DATA

CM3203: One Semester Individual Project

Author: Timothy Bird
Project Supervisor: Frank C Langbein
Project Moderator: Matthias Treder
School of Computer Science and Informatics, Cardiff University
Date: May 2019

Abstract

This report describes a solution to the problem of when isolation leads to human actions not being recorded or communicated.

In recent years, computers have become much smaller and the field of machine learning is becoming more accessible. Sensors are now being used to upload data to the internet, removing the stage of people having to upload such data.

Making use of these trends, this project has created a human activity prediction system for running, walking, and standing still (and any activity that is none of these). The system is small, makes use of machine learning and has sensors to recognise the activity being performed.

The learning process is fully described, from data collection to testing. The implementation of this system uses a Raspberry Pi and Sense HAT (using its accelerometer and orientation sensors). The Raspberry Pi runs a record script to collect data which is then placed into a more powerful computer to generate prediction systems using two Support Vector Machines. These two Support Vector Machines are then placed back into the Raspberry Pi to run a prediction script.

This report documents the evaluation phase of the project, describing both the methods used to evaluate the system as well as the results created. The report also goes through alternative methods and shows via the evaluation why they are not suitable. Suggested future work that builds on the motivations and aims described within the report is noted. This future work includes both the hardware aspects as well as the machine learning process.

Acknowledgements

First, I would like to acknowledge my supervisor, Frank Langbein, for his continuous support and guidance of the project. Giving his time for weekly meetings has played a huge part in its success. I would also like to thank Daniel Harborne for lending extra hardware parts, which allowed me to speed up the training data process. Finally, I would like to acknowledge all my friends and family who helped to proof read this report.

Contents

Abstract 1
Acknowledgements 1
1. Introduction 4
2. Background 5
2.1 Machine Learning 5
2.2 K-means Clustering 5
2.3 Support Vector Machines 6
2.4 Human Computer Interaction 6
2.5 Internet of Things 6
2.6 Competitors 7
2.7 Raspberry Pi 7
3. Approach 9
3.1 Definitions of activity recognition 9
3.2 Aims 9
3.3 Architecture 10
3.3.1 Software Stack 10
3.3.2 Hardware 11
4. Implementation 12
4.1 Hardware 12
4.2 Software 12
4.3 Activities 13
4.4 Recording 14
4.5 Data Transfer 15
4.6 Machine Learning 16
4.7 Prediction 17
4.8 Improving the models 18
5. Results and Evaluation 19
5.1 Evaluation Methods 19
5.2 Alternative methods 20
5.2.1 Sensors 20
5.2.2 Change of readings 22
5.2.3 Refresh loop 23
5.2.4 Position of Mobile computer 24
5.2.5 Choosing the Training model: K means clustering vs Support Vector Machine 26

5.2.6 Creating the second outlier Support Vector Machine 28
5.2.7 Failed outlier evaluation 32
6. Future Work 33
7. Conclusions 35
8. Reflection on Working 36
Appendix 37
References 38

Figure 1. IoT Lifecycle 7
Figure 2. Prediction and Recording architecture 10
Figure 3. Machine learning architecture 10
Figure 4 Hardware architecture 11
Figure 5. Recording Pseudocode 14
Figure 6 Extract of data.csv file 14
Figure 7 File Structure 15
Figure 8 Conversion of data 16
Figure 9. Prediction Pseudocode 18
Figure 10. 2D visualisation of clusters 27

Table 1. Project Aims 9
Table 2. Python Modules 12
Table 3. Activity Map 17
Table 4. Final Confusion Table 19
Table 5. Final evaluation metrics 19
Table 6 X,Y,Z confusion matrix 20
Table 7 Sensors X,Y,Z evaluation metrics 21
Table 8 Sensors Yaw, Roll, Pitch Confusion Matrix 21
Table 9 Sensors Yaw, Roll, Pitch evaluation metrics 21
Table 10 Unchanged Readings Confusion Matrix 22
Table 11 Unchanged Readings evaluation metrics 22
Table 12 Refresh loop of 1.0 Confusion Matrix 23
Table 13 Refresh loop of 1.0 evaluation metrics 23
Table 14 Refresh loop of 1.5 confusion matrix 24
Table 15 Refresh loop of 1.5 evaluation metrics 24
Table 16 Ankle Confusion Matrix 25
Table 17 Ankle evaluation Metrics 25
Table 18 Wrist Confusion Matrix 25
Table 19 Wrist evaluation metrics 26
Table 20. Clustering Confusion table 28
Table 21. Clustering Evaluation metrics 28
Table 22. Initial Outlier Results 29
Table 23. Outlier Results 30
Table 24 Personal Bias Confusion Matrix 34
Table 25 Personal Bias Evaluation metrics 34

1. Introduction

This project is called “Activity Recognition on Body data”. The motivation behind my project is that the data humans create often disappears and is not recorded or recognised.

What this project can lead to are solutions to problems of isolation. One problem with isolation is that humans must call out when in need of others. A solution to this is to have a computer automatically send out a distress signal. A computer can send a message faster and reach more people than any human. What the computer needs is the signal to send the message. This signal can come from sensors that are always giving the computer data. This motivation is a subset of humanity’s aim to benefit from becoming cyborg organisms. The project is not a novel idea: already there are smart watch products like the iWatch (Apple, 2019) and the Fitbit charge (FitBit, 2019), both of which categorise and record activity. One visible difference between my project and these current devices is that the sensor is worn on the thigh, not on the wrist. The activity is also recognised by a local system, rather than one where data are sent to a remote data centre, where the user has no control of what then happens to their data.

I created two machine learning models that run on a Raspberry Pi with Sense HAT attached. All the data I used was personal data. The Raspberry Pi can recognize walking, running, standing still, and an activity that is not any of these, to an acceptable rate of accuracy when next to the user’s thigh. This can be done by putting the Raspberry Pi in a front pocket. Because the Raspberry Pi requires a power source, the user needs to carry an external battery as well as a micro USB-to-USB cable that connects the two.

Over the course of the project, the aims and methods were evaluated, and the weighting of the aims evolved. For example, the initial plan involved a mobile front end that would be required to display the activity the computer was predicting. When the Sense HAT was chosen, this was no longer required. The Sense HAT already had an 8x8 RGB LED matrix that allows the program to display letters to represent numbers. This was enough to display what the project required, so a mobile app was no longer required.

A large challenge of this project was picking the right machine-learning model for the activity prediction. The Support Vector Machine was chosen over K-means clustering as it provided better results in an initial evaluation. The evaluation for the models used mathematical measures that ran formulas on test data as well as a live evaluation of testing the machine’s prediction against the activity. This type of evaluation was not as detailed as the coding evaluation, as what was being tested was if it could predict the right activity to a rate that a human would keep using it, which does not need to be as detailed.

Project management was achieved through weekly meetings with my supervisor. The workload changed considerably throughout the project.

This report will start by going through important aspects of computer science that are needed to understand the project, before going through the main approach to the solution. After giving an overview of the solution, it will go into more detail. This is then followed by an evaluation where different methods and the results they bring are described. The report concludes with a description of possible follow on work.

2. Background

2.1 Machine Learning

This prediction system gains data from sensors and then needs to categorise these data into an activity. To create the dividing lines between activities, the machine learns the dividing line rather than the programmer hardcoding these lines. The benefit of machine learning is that human error can be taken out of the process.

Humanity and machines have had a long relationship. Over much of that time, it has been one-sided, with humanity having the intelligence and machines following the laws of physics. That changed in the 1950s when research into Artificial Intelligence (AI) began (Turing, 1950). As computational power has increased this has allowed research theory to be put into practice. In the mid-2000s a subset of AI started to gain popularity: Machine Learning. Google defines this as "A program or system that builds (trains) a predictive model from input data. The system uses the trained model to make useful predictions from new (never-before-seen) data drawn from the same distribution as the one used to train the model" (Google, 2019). This changed the relationship between humans and machines, as machines were no longer unintelligent. They could change their behaviour depending on what data they were given.

The emergence of the internet and knowledge sharing was a big factor in the rise of machine intelligence. As more people used the internet, they uploaded information and data onto remote servers, and as time went on, the data that was on these servers became vast. So vast that it was given the label Big Data. Big Data is defined as" very large sets of data that are produced by people using the internet, and that can only be stored, understood, and used with the help of special tools and methods" (Anon., 2019). These "special tools and methods" include machine learning algorithms. Like humans whose understanding of a subject increases the more examples of a subject they come across, the machine learning algorithm benefits from the more examples it is given. There have been examples of this in the commercial environment with supermarkets can understand the behaviour of their customers to a more personal degree. As using these algorithms has led companies to make greater profits, a lot more research has been put into this field. This further research has allowed machine learning algorithms to be placed into simple packages for anyone to download and use.

This is where this project comes in: as a result of 70 years of research and millions of pounds of investment on what information can be downloaded from the internet.

2.2 K-means Clustering

K-means clustering was one of the methods considered to classify the data into activity groups. K-means clustering is an algorithm that solves clustering problems. The main idea behind the algorithm was published in the late 1950s (Steinhaus, 1956). A clustering problem is when there are multiple data points within a domain of any dimension. The aim is to group the data points that are similar into the same group. It is an unsupervised problem as each point is not given a class. What is inputted into the algorithm is the number of classes = k. The algorithm is as follows:

1. K initial "means" are randomly generated within the data domain.
2. K clusters are created by associating every observation with the nearest mean.

3. The centroid of each K cluster becomes the new mean.
4. Steps 2 and 3 are repeated until the means of every K cluster do not change.

K-means clustering is one of the least computing intensive unsupervised problems (Hinton & Sejnowski, 1999). It was chosen to be included in this report as it was one method explored, through guidance from the project's supervisor.

2.3 Support Vector Machines

Support Vector Machines was another method used to classify the data into activity groups. Support Vector Machines are supervised learning models with associated learning algorithms that take in data points and define them into different classes, making it useful for classification problems (Cortes & Vapnik, 1995). The classification problem works on data points of different classes and tries to work out the separating line between them. The Support Vector Machine solves this problem by constructing a hyperplane, mapping the data points onto this new hyperplane and then performing its classification there. The reason for creating the hyperplane is that it can create the distance between two points of different classes to be as large as possible. It was chosen to be included in this report as its needs in terms of computing were lower than an alternate method, neural networks (DeepAI, 2019), which might have been possible for this project. However, no work was done with neural networks so it cannot be confirmed.

2.4 Human Computer Interaction

A motivation for developing this solution came from looking at the Human - Computer interaction and its limitations.

The internet has allowed humans to share information with the most common type of human-computer interaction being manual typing on a keyboard (Dix, et al., 2004). Through language we can convey thought, but there are some instances when humans cannot convey their thought through typing. One example is falling over. The person falling over does not convey they are falling over to the observer by typing a message. Most commonly, the person gives out a loud yelp and physically falls to the floor. To a human observer, it is not the loud yelp that tells them that the other person is falling over, as a loud yelp is used in multiple examples and is not exclusive to falling over. What is exclusive to falling over is the body movement. The buckling of the knees, the upright position of the body turning horizontal, with the shoulders falling to the ground, and the time taken to complete these actions. The human observer uses these body movements to recognise the action of falling that the person has taken. The observer can turn these observations into text and upload the text onto the internet. This means that a third person can read the text from the internet and relate to the action and understand the action of falling over. If an instance of falling over happens in front of a third person, they would not refer to the text, but they would still use the body movements as indicators. The point of this example is that the internet of text-to-data cannot fully map all reality.

2.5 Internet of Things

To record human activity, sensors can be used to upload data, removing the step of the human uploading the data. The internet has evolved from a state of inputting text to data centres. Alongside big data that can upload commercial transactions, there are also mechanical sensors that upload their readings onto the internet. This network of sensors is called the 'Internet of Things' (Ashton, 2009). The Internet of Things is defined in the

Cambridge dictionary as “objects with computing devices in them that are able to connect to each other and exchange data using the internet”. This removes the human component in the internet. The sensor can collect the data, send it anywhere on the internet to have the data processed, with instructions sent to another device that takes an action depending on the instructions (Figure 1). Within these three stages, there can be multiple devices. Multiple machines can act on the readings of a single sensor, or multiple sensors can be taken into consideration before a single action is made.

Figure 1. IoT Lifecycle

What is most relevant to this project is the collection of data. The collection of data is what shapes the reality that the computer at the analysis phase of the IoT cycle then uses. If you use the metaphor that the computer is like a human brain, then the reality it perceives has to be at least comparable to that of humans to have any utility. Thus, when creating my project, careful consideration was put into what sensors were used and how the data were collected, so the computer had enough grasp of our world to perform the tasks I wanted it to perform. Computer systems can use a vast range of sensors, and they can have advantages over human senses with sensors recording light waves whose frequency causes it to escape the human eye.

2.6 Competitors

The problem detailed in this report has tried to be answered by several solutions that can be found. With this architecture of IoT sensing the world, it is inevitable that the sensors would point towards humans as currently the most intelligence and arguably the most complex object in the universe. A common aim of people is to be healthier, and not to be in pain or to die. Products have appeared to satisfy this aim. The main example relevant to this study is the Fitbit Charge line (FitBit, 2019). Fitbit is a company that owns a wide range of products, but I am going to focus on the Fitbit Charge as it is advertised as a tracker, so its main use is to track human activity for the benefit of that person. It uses the company’s own technology called Smarttrack which can recognize workouts or activities where there is continuous movement. It uses a 3-axis accelerometer to understand its users’ motions. It then uses secret algorithms to convert the data into usable information for the customer.

2.7 Raspberry Pi

The Raspberry Pi (RaspberryPi.org, 2019) is a computing device that can be used to carry out recording and prediction scripts whilst attached to the human body. It is a commercial computer with an educational aim. Its specifications are a 4× ARM Cortex-A53, 1.2GHz CPU, 1GB LPDDR2 of RAM, 2.4GHz 802.11n wireless, and uses a microSD card for storage. The length of the computer is similar to a credit card. As of May 2019, it cost around

£30. The operating system that the computer runs on is called Raspbian, a slightly modified version of Linux.

The Sense HAT is a product also made by the Raspberry Pi foundation. It connects to the Raspberry Pi through the GPIO pins of the Raspberry Pi (HAT, 2019).

The Raspberry Pi and Sense HAT were chosen because they allow all aims of this project to be completed. There were other hardware options that also could be used to complete the same aims. The main disadvantage of the Raspberry Pi and Sense HAT is that they have functionality not required for this project, and are therefore redundant. The features that are not used are: BlueTooth, 2.4GHz and 5GHz wireless, Ethernet port, CSI camera port, DSI display port, Magnetometer, Temperature, Barometric pressure, Humidity. Another option was to have obtained the specific features and to have built a mobile computer. Building a mobile computer means only the features used in the project would be present, for example the only sensors would be the accelerometer and gyroscope. The disadvantage of building a mobile computer is that is takes longer than buying commercially available hardware, such as the Raspberry Pi and Sense HAT. The time of this project was focused on the machine learning creation, so the hardware solution with the smallest amount of time was chosen. There may be other already created computers that fit the aims better than the Raspberry Pi, but the Raspberry Pi had a lot of documentation and software package support which allowed for straightforward programming.

3. Approach

3.1 Definitions of activity recognition

Activity Recognition is when the machine can predict the activity a person is performing. However, the term activity is too broad for this report, where the Cambridge Dictionary defines it as “the situation in which a lot of things are happening or people are moving around” (Dictionary, 2019). Within this report an Activity can only be performed by a single person. The activity must be physical. If someone was playing chess, then the activity being performed would be sitting down with minimal arm movement rather than playing chess.

3.2 Aims

In order to complete my solution to “Activity Recognition on Body data”, I needed to break this down into aims with different levels of priorities.

Table 1. Project Aims

Description	Priority	Achieved by	Background support
Be wearable/ not restrict the user to complete the activities that it is trying to identify	Must	Hardware	See section Human-computer Interaction (p6)
Display text that can be linked to an activity	Must	Hardware	See section Human-computer Interaction (p6)
Be portable	Must	Hardware	See section Raspberry Pi (p7)
Identify that the wearer is standing still at a rate of at least 90%	Should	Software	See section Machine Learning (p5)
Identify that the wearer is running at a rate of at least 90%	Should	Software	See section Machine Learning (p5)
Identify that the wearer is walking at a rate of at least 90%	Should	Software	See section Machine Learning (p5)
Record the user standing still	Should	Software	See section Internet of Things (p6)
Record the user running	Should	Software	See section Internet of Things (p6)
Record the user walking	Should	Software	See section Internet of Things (p6)
Identify when the computer is not sure what the user is doing	Could	Software	See section Machine Learning (p5)

Giving each aim a priority shows that different approaches can be created to solve this problem. Every solution must be wearable, portable and display text. As the computer must be linked to an individual, it needs to be wearable and portable as humans are mobile thus the computer must be able to stay with the person. It must be able to display text as the

computer will need to give information to the benefit of the human user. All the aims marked with the “should” priority let the system record and predict an activity. As the project is called “Activity recognition”, it then needs to record and predict activity. It is not a must priority, however, as there could be other solutions to activity recognition, one solution being setting hard set classification within the sensor readings. However, in the Background section, these aims seem to be the more suitable option of using machine learning, with sensors uploading to a machine with no human input. The final aim of my solution “Identify when the computer is not sure what the user is doing” is marked as could. This is because it is not needed for activity recognition. What it is used for is to make the prediction more accurate and opens the possibility to recognise the $(N+1)^{th}$ activity. N being the number of activities recorded.

3.3 Architecture

3.3.1 Software Stack

Figure 2. Prediction and Recording architecture

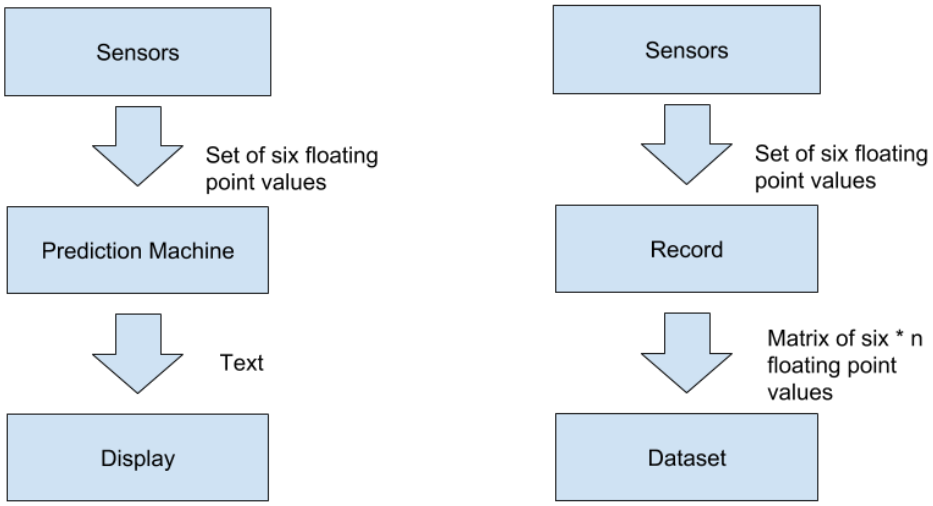
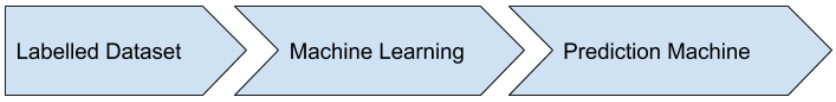


Figure 3. Machine learning architecture



The architecture approach to the solution can be split into three processes; Recording, Machine Learning and Prediction (Figure 3). The Recording process has the mobile machine’s sensors take in values at a rate of 0.5 second (for justification of 0.5 seconds see

Sub Section Refresh loop of section Evaluation). These values are sent into a dataset under respective headers of X, Y, Z, Pitch, Roll, Yaw. The name of these headers come from each dimension the sensor is recording. The dataset is then put into a machine learning capable computer. It is labelled with the correct activity that was recorded. This allows supervised learning to take place. Through the supervised learning two prediction models are created: the activity recognition model and the outlier model. These two prediction models are placed back into the mobile computer. For the prediction process, the sensors take in the values at a rate of 0.5 second. This is the same as the recording process. The inputs are then placed into two prediction models, one set of recording values at a time. The prediction models' output is displayed via text.

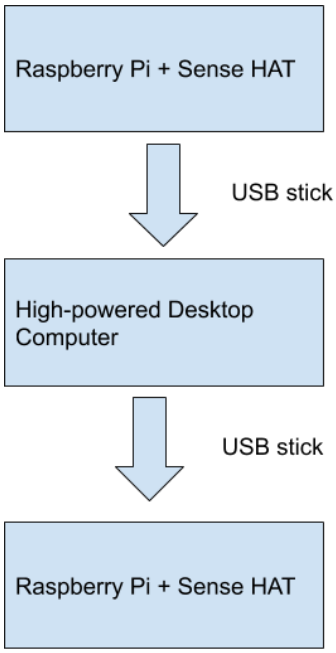
To transfer data between the mobile computer and the computer capable of creating the prediction model I chose the approach that the data needs to be manually transferred via portable memory rather than a request over the internet. The reason behind this choice was of privacy and of giving back data ownership to the user. If data were sent via the internet, it could be retrieved at any time the mobile machine was turned on. This leads to privacy concerns, as the central system could take personal data generated by the person performing the activities without their consent. Manual transfer of data is slower and involves mores steps but removes privacy concerns as the one using the mobile machine must hand over their data.

The sensors come from X, Y, Z, Pitch, Roll, Yaw. The X, Y, Z values are the recording of G-force acting on each axis. The Pitch, Roll, Yaw are recording on how much the object is rotated around the X, Y and Z axis. Through the Pitch, Roll, Yaw readings the orientation of object can be calculated.

3.3.2 Hardware

In Figure 4 it shows the hardware cycle of the machine learning process. The Raspberry Pi is needed for the recording and prediction. The High-powered Desktop computer is needed for the Support Vector Machine creation. The USB stick is needed for data transfer.

Figure 4 Hardware architecture



4. Implementation

4.1 Hardware

The aim of a wearable and portable computer was implemented by the Raspberry Pi. Being the size of a credit card, it could fit in a trouser pocket. The Sense HAT add-on was used in order to complete the aim “Display text that can be linked to an activity”. The Sense HAT has an 8x8 RGB LED matrix, which can be programmed for each LED. With this capability I was able to display letters on the matrix. The Sense HAT has a multitude of sensors, but the ones used for this study were the 3D accelerometer and 3D gyroscope.

Another piece of hardware that was used was a 16gb Micro USB and a memory stick. The portable storage units were needed as the Raspberry Pi is used to record and predict activity, but the CPU is not powerful enough to create the models. So, these two devices were used to transfer data between the two computers. The specification of the computer to create the machine learning model are 8GB of RAM, i5-4690k CPU @3.5Hz with a Nvidia 970 gtx Graphics Card. These specifications are much greater than the Raspberry Pi. It was used not only to compute the prediction models but also to create the code of the program. The Raspberry Pi only has two scripts for the two modes, one for recording and the other for predicting.

4.2 Software

Python was used as the programming language. This was because the Raspberry Pi supports Python and Python has modules that are useful for this project, a main example being Sklearn. The data was recorded in CSV file format. With the CSV format, I could read and write in Python, as well as it not taking that much data. The style of my data was two dimensional 6 * n, therefore I did not need the files to hold any other formatting styles, thus CSV was enough.

Along with Python, 3rd party libraries were used. This was to give additional functionality. These libraries were installed via the command “Python –m pip install ‘PackageName’” using the module Pip. Pip is a module that is used to install other modules and version control. To make sure the programs worked on both the mobile computers and the powerful computer used to create the Support Vector Machine, the versions of all the packages needed to be the same. In order to do this, on the powerful computer, the command “Python –m pip freeze” was used to display all the versions. On the mobile computers packages were installed with the additional parameter to install the package to the same version as the powerful computer. The command to do this was “Python –m pip install ‘PackageName’== ‘version number’”.

Table 2. Python Modules

Name	Version	Purpose within project
Python	2.7.10	The main programming language
Pip	19.0.3	Used to install these packages.

Pickle	2.0	Gives the ability to save/load prediction models to/from ".sav" file
Pandas	0.24.1	Has data frame object that places the training data into a form that can be used with the support vector machines
NumPy	1.16.1	Uses the array, argmax, bincount function to create the short-term memory in the predict function.
CSV	2.7.10	The reader and the writer functions are used to take and place data to and from the .csv files
Sense_hat.SenseHat	2.2	Has the ability has to take the sensor readings
Sklearn	0.18	The Machine learning package that gives the support vector machines.

Sense_hat.SenseHat is the only module that can be used to access the sensor readings of the Raspberry Pi. However, the modules Pickle, Pandas, Numpy, Sklearn give functionality that can also be found in other modules. The reason why these four modules were chosen was due to their popularity and documentation. These modules were found due to searching the required functionality in a Python environment. These modules were the first to appear. Their functionality was appropriate, so they are used.

4.3 Activities

This project records four activities: Standing Still, Running, Walking and an activity that is none of the three. To identify the difference between the activities, the activities must be clearly defined:

1. Standing still has the user standing upright with both feet on the floor.
2. Walking has the user moving forward with at least one foot on the floor.
3. Running has the user move at a speed faster than a walk, never having both feet on the ground at the same time (Anon., 2019).
4. An activity that is not one of the first three.

Standing still was the first activity to be chosen as it is an activity where a good classifier is that all readings should be 0. Knowing the reading (0,0,0,0,0,0) equals standing still, then it can used to test the prediction models when the reading (0,0,0,0,0,0) is given to them. Walking and Running was then chosen because they are both different from standing still but also very similar to each other. Falling over was also chosen as an activity to record, being different to walking and running due to not being a continuous action. However, the results of this activity prediction were too low to include in this final implementation. The work used to try to predict the falling over activity was used in the creation of predicting the outlier activity. This was chosen to improve the prediction machine as will be described in the sub-section 'improving the models'.

4.4 Recording

The project, from a machine learning point of view, is a classification problem with supervised learning. The Recording script makes use of the joystick on the Sense HAT. The Record.py has a while loop that runs an iteration every 0.5 seconds (Figure 5). There is also a Boolean called “isRecording”. Only when “isRecording” is set to true, then the sensors are written to CSV. “IsRecording” is set to true as default but can be changed by moving the

Figure 5. Recording Pseudocode

```
IsRecording = true
While True:
        If joystick.down:
                isRecording = false
        if joystick.up
                isRecording = true
        if(isRecording)
                record sensor values
        Wait for 0.5 seconds
```

joystick on the Sense HAT. Moving the joystick down will turn the “IsRecording” to false, which stops the recording of sensors. Turning the joystick up again will turn the Boolean to true, which will re-start the recording. This while loop does not start as soon as the program starts. The while loop begins upon the first movement of the joystick up. Both the Record.py and Prediction.py script run as soon as the Raspberry Pi boots up. This is done by adding a line in the Raspberry Pi’s cron file to call the script to run on reboot.

To create the training data, I performed the activities myself. The recording program writes to a file called “data.csv”. Before each recording session, the file “data.csv” has to be cleared. The activity would then be performed for a set amount of time. “data.csv” would be moved to the USB stick whilst changing the name to the activity that was performed. Figure 6 are six lines from “10minRun1.csv”. There are no column headers but the sensors will always send their data into the same column. Pitch is in column one, Roll is in column two, Yaw is in column three, X is in column four, Y is in column five, and Z is in column six.

Figure 6 Extract of data.csv file

6.432524	8.418149	143.9565	0.011601	0.077208	0.685763
328.2209	55.08839	115.0223	0.241652	0.424316	0.217134
308.3137	50.70705	79.6206	0.609994	0.800973	0.087487
2.020355	107.0929	347.323	0.312028	0.479798	0.197394
2.073489	190.2453	350.0914	0.497499	0.184522	0.116974
26.64409	174.7224	8.241159	0.640538	1.151612	2.532741

- To gather the data for the activity standing up, I would place the Raspberry Pi in my pocket and stand for 10 minutes to create ‘newStanding_edit.csv’.

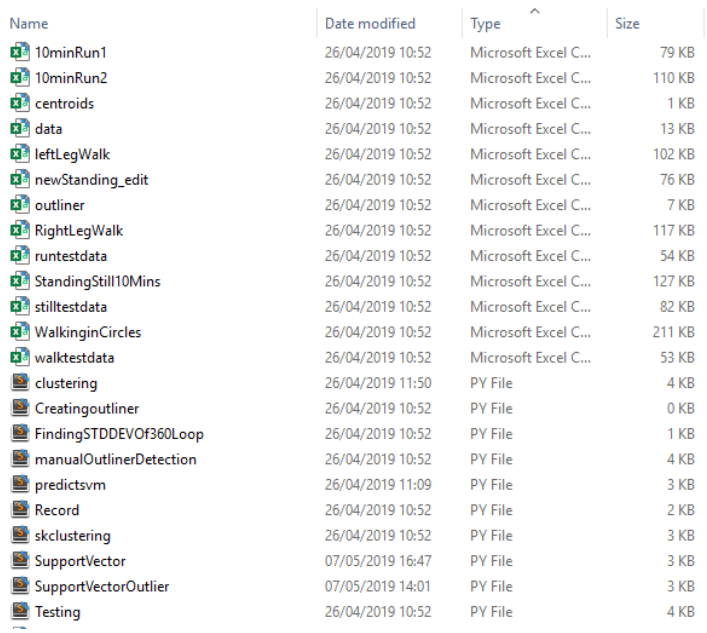
- To gather the data for the activity walking I put a mobile computer in my left and right pocket and walked mostly in one direction for ten minutes. This created two csv files, “leftLegWalk.csv” and “RightLegWalk.csv”. To broaden the range of readings of sensors, I placed the mobile computer in a pocket, and I would walk around in circles for ten minutes. For Five minutes the direction would be clockwise and for the other five minutes the direction would be anti-clockwise. This created “WalkinginCircles.csv”.
- To gather the data of the activity Running data, the recording was similar to walking with two mobile computers running the record program in each front pocket. When running for 10 minutes, the route has sufficient turns that there was no need to record a “RunninginCircles.csv”. The two CSV files that were created were “10minRun1.csv” and “10minRun2.csv”.

The training data were gathered before the Support Vector Machine training was fully implemented. This meant extra data for each activity was recorded for testing purposes in their own csv file. These CSV files are called: “stilltestdata.csv”; “walktestdata.csv”; and “runtestdata.csv”. But because of the training step used, the test data did not need to be in their own CSV file. This meant that the recording in these CSV was put in with the rest of the data. This meant for the three activities the total data points were: 1,826 for standing still; 3,095 for walking; and 2,726 for running.

4.5 Data Transfer

Once the CSV file is created and renamed it is moved onto the USB stick. The USB stick is then safely ejected from the Raspberry Pi and put into the hard powered computer. Once the computer can access the CSV file it is transferred from the USB folder to the folder that includes the SupportVector.py file. Having the CSV files in the same folder as the SupportVector.py and SupportVectorOutlier.py files is needed as those files look for the CSV files in the same folder. Figure 7 shows file structure of the CSV and programs in the same folder. The program files on the Raspberry Pi are placed on the desktop folder. Using the crontab file line “@reboot python /home/pi/Record.py”. it runs the file on reboot, knowing where to look for the file.

Figure 7 File Structure



Name	Date modified	Type	Size
10minRun1	26/04/2019 10:52	Microsoft Excel C...	79 KB
10minRun2	26/04/2019 10:52	Microsoft Excel C...	110 KB
centroids	26/04/2019 10:52	Microsoft Excel C...	1 KB
data	26/04/2019 10:52	Microsoft Excel C...	13 KB
leftLegWalk	26/04/2019 10:52	Microsoft Excel C...	102 KB
newStanding_edit	26/04/2019 10:52	Microsoft Excel C...	76 KB
outliner	26/04/2019 10:52	Microsoft Excel C...	7 KB
RightLegWalk	26/04/2019 10:52	Microsoft Excel C...	117 KB
runtestdata	26/04/2019 10:52	Microsoft Excel C...	54 KB
StandingStill10Mins	26/04/2019 10:52	Microsoft Excel C...	127 KB
stilltestdata	26/04/2019 10:52	Microsoft Excel C...	82 KB
WalkinginCircles	26/04/2019 10:52	Microsoft Excel C...	211 KB
walktestdata	26/04/2019 10:52	Microsoft Excel C...	53 KB
clustering	26/04/2019 11:50	PY File	4 KB
Creatingoutliner	26/04/2019 10:52	PY File	0 KB
FindingSTDDEVOf360Loop	26/04/2019 10:52	PY File	1 KB
manualOutlinerDetection	26/04/2019 10:52	PY File	4 KB
predictsvm	26/04/2019 11:09	PY File	3 KB
Record	26/04/2019 10:52	PY File	2 KB
skclustering	26/04/2019 10:52	PY File	3 KB
SupportVector	07/05/2019 16:47	PY File	3 KB
SupportVectorOutlier	07/05/2019 14:01	PY File	3 KB
Testing	26/04/2019 10:52	PY File	4 KB

4.6 Machine Learning

After the recording, there is a pre-processing step in which I labelled the data with the relevant data as only one class was recorded per session. The seven * n matrix was then processed again so each row now had the difference between the current row and the previous one to create a seven * (n-1) matrix. Figure 8 shows an example of the conversion on an extract of data.

Figure 8 Conversion of data

6.432524	8.418149	143.9565	0.011601	0.077208	0.685763
328.2209	55.08839	115.0223	0.241652	0.424316	0.217134
308.3137	50.70705	79.6206	0.609994	0.800973	0.087487
2.020355	107.0929	347.323	0.312028	0.479798	0.197394
2.073489	190.2453	350.0914	0.497499	0.184522	0.116974
26.64409	174.7224	8.241159	0.640538	1.151612	2.532741

Conversion in the importChangeofData(csvname, classInput) function

38.1791	46.67024	28.9342	0.230051	0.347108	0.468629
19.9072	4.38134	35.4017	0.368342	0.376657	0.129647
53.70666	56.38585	92.2976	0.297966	0.321175	0.109907
0.053134	83.1524	2.7684	0.185471	0.295276	0.08042
24.5706	15.5229	18.14976	0.143039	0.96709	2.415767

Using the change of data is better than just the data as it shows the rate of movement. This array of rate of change was then split into 4:1 ratio of training data to test data. Both of these two data sets are further split into X_train, X_test, Y_test,Y_train. Data Sets X_train and X_test are a six * n matrix with the six columns being the six sensors. Y_test and Y_train are a one * n matrix, holding the class value for each row. Sciklit-learn SVM package is used with the lines below to create the class model.

```
clf = svm.SVC(kernel='linear')
```

```
clf.fit(X_train,y_train)
```

What clf.fit() is doing is working out where to place the line in 6-dimensional space to differentiate the three activities. The model is then put under some evaluation with sklearn.metrics, printing the confusion matrix and classification_report.

```
y_pred = clf.predict(X_test)
```

```
print(confusion_matrix(y_test,y_pred))
```

```
print(classification_report(y_test,y_pred))
```

Please note the two train data were used in the clf.fit() function to create the Support Vector Model and the two test data were used in the evaluation lines. This is needed as testing should be done on data different to what was used to train the model.

For the other prediction model, the same data is used but not using the Class attribute, as the model is trying to put a line between what can be recognised and what cannot. I did not need to distinguish between the different classes. Therefore I did not need to break up my data into four sets, only two sets called inlinerdf_train and inlinderdf_test. The Support Vector Machine was created with the lines below.

```
clf = svm.OneClassSVM(nu=0.1, kernel="rbf",gamma = 0.0001)
clf.fit(inlinerdf_train)
```

Once I created both models, I used the python module Pickle. This turns the model into a .sav file that can be transferred via the USB stick to the Raspberry Pi for its second script called predict.

4.7 Prediction

The predict.py script uses the pickle module to load the two models. The same sensors used to record are used again to record at the same rate of 0.5 seconds placing their recordings into arrays in a while true loop. After every other iteration, it works out the difference between the two readings. The change in readings is placed in the outlier prediction module. If it predicts that it is an outlier, the Raspberry Pi outputs “O” on its display and the recording is placed into a csv file called “outlier.csv”. If it predicts that it is not an outlier, then the other prediction model is used to predict what activity is happening. The prediction is placed into an array of length 5. Once the array is full, the Raspberry Pi displays a letter which is connected to the activity that appears the most in the array. After the prediction models are finished, the recording arrays are cleared so they can be used for the next set of readings.

Table 3. Activity Map

Activity	Letter output
Standing Still	S
Walking	W
Running	R
Outlier	O

Figure 9. Prediction Pseudocode

```
ShortTermMemory  = array of length 5
Create ActivityMap
While true;
        Get Recording
        If Secound recording:
                Data point =  difference between recording
                If Data Point == outlier:
                        return “O”
                Else:
                        Predict activity from data point
                        shorttermMemory.Add(activity)
                        if ShortTermMemory .length > 5:
                                delete oldest ShortTermMemory  item
                        if ShortTermMemory  == 5:
                                return ActivityMap (Activity that is the mode of ShortTermMemory)
```

4.8 Improving the models

When improving the Support Vector Machine Models the training data needed to improve. In order to do this the outliers recorded that were actually mis categorized and was one of the activities that must be placed back into the activity training data. This is done by clearing the outlier.csv file before a prediction session. The mobile computer then runs the prediction program whilst the user performs one of the three activities. Once the program ends, all points in the Outlier.csv file are false positives. These points can be transferred into the training data of that activity. Both Support Vector Machines must be re-trained.

5. Results and Evaluation

5.1 Evaluation Methods

Once fully completed my project was able to recognise the three activities and recognise when it was not sure.

To evaluate the measure of my success, I wrote into the create Support Vector Machine script some evaluation measures which are: support, precision, recall, F1 score and the confusion matrix. In the model creation script, the data were broken up into training and testing data, the split being 4:1. The confusion matrix gives a visual representation as you can see how many items are recognised in the wrong category. Other measurements as well as they put different values on different areas.

Precision, which equals to $\frac{true\ positive}{total\ predicited\ positve}$ is good when the false positives cost is high. In our situation of activity detection, it would be bad if the correct transaction gets detected as something else. Recall, which equals $\frac{true\ positive}{total\ actual\ positive}$, highlights the importance of false negatives. This can be even more important than precision as it measures how many true activity points are not being detected, which is the whole point of the classifier.

My final measure is the F1 score, which equals to $2 \times \frac{precision \times recall}{precision + recall}$, because it is a better combination method rather than the confusion matrix. This is because it gives more weight to false negatives and false positives whereas the confusion matrix mostly relies on true negatives. The support is the number of occurrences of each class that appear as a positive. As it prints the number, it can be compared to the number of rows within the test data to see how much they match. Once the Support Vector Machine is created the measures are printed out. All the measures are recorded for each activity. For each measure I made an average for all the activities to give a single number. This allowed me to see if one activity is easier to recognise in a classification method.

Table 4. Final Confusion Table

Actual		Predicted		
		Standing	Walking	Running
	Standing	363	1	0
	Walking	4	576	46
	Running	0	79	461

Table 5. Final evaluation metrics

	Precision	Recall	F1-score	Support
Standing	0.99	1.00	0.99	326
Walking	0.88	0.92	0.90	626

Running	0.91	0.85	0.88	540
Avg / total	0.92	0.92	0.91	1530

I also evaluated the models by performing one activity whilst recording the prediction and seeing how many times it was correct. This gave less information than the coded in measures, but it is the measure that the user will experience. The measure I was looking out for is how many times will the device predict the wrong number until the user gets fed up with it. However, this is not a standard measure as each person will get fed up at different rates and getting fed up is not a well-defined measure. But, adding the human factor into the testing stage is essential as this project is created to be used by humans.

With these two types of evaluation I could try out different ways of recording and which classifier to use.

5.2 Alternative methods

This section of the report examines alternative methods and ways of implementation that were tested and how the implementation method stated in the implementation section was evaluated and chosen.

5.2.1 Sensors

There are a few decisions regarding certain values within the project. The number of sensors is six, being X, Y, Z, Roll, Yaw, Pitch. These sensors are needed to capture the human movement. The X, Y, Z sensors measure the g force in their relative dimension. Adding all these sensors gives the g force in the 3-dimensional world. The sensors Roll, Yaw, Pitch were added as they show the orientation of the device. Both sets were chosen as they give better results in the evaluation models compared to only one set of three sensors. Initially the orientation gives little information on the body movement as the device can achieve all range of orientation whilst standing still. This is why I used the change of sensors readings. The change of orientation readings gives a better understanding of movement. A higher rate of orientation readings indicates a more exaggerated movement.

For all six sensors – please refer to table 4 and 5

Using only Sensors X,Y,Z: tables 6 and 7

Table 6 X,Y,Z confusion matrix

Actual		Predicted		
		Standing	Walking	Running
	Standing	365	0	0
	Walking	18	595	54
	Running	2	121	375

Table 7 Sensors X,Y,Z evaluation metrics

	Precision	Recall	F1-score	Support
Standing	0.95	1.00	0.97	365
Walking	0.83	0.89	0.96	667
Running	0.87	0.75	0.81	498
Avg / total	0.87	0.87	0.87	1530

Using only Sensors Yaw, Roll, Pitch: tables 8 and 9

Table 8 Sensors Yaw, Roll, Pitch Confusion Matrix

Actual		Predicted		
		Standing	Walking	Running
	Standing	362	11	0
	Walking	9	530	88
	Running	2	185	343

Table 9 Sensors Yaw, Roll, Pitch evaluation metrics

	Precision	Recall	F1-score	Support
Standing	0.97	0.97	0.97	373
Walking	0.73	0.85	0.78	627
Running	0.80	0.65	0.71	530
Avg / total	0.81	0.81	0.80	1530

Looking at the results the acceleration sensors gave better results than the orientation sensors. But even the acceleration results were not as good using both sets of readings. The

difference between the separate acceleration and orientation results could be due to how acceleration and orientation movement was captured. Both work in a 3-dimensional space, whereas acceleration is the change of velocity and orientation is the rotation around a point. Velocity is more useful to movement than rotation. This could be why a competitor, the Fitbit charge, has an accelerometer but no gyroscope.

5.2.2 Change of readings

The benefit of using the change of readings is that the device can be fitted in any orientation on the person. Using the sensor readings would mean wearing the device upside down would change how the prediction machine will handle the readings. One issue with using the change of degree is that the orientation sensors range from 0 to 360 degrees. This means if one reading is 2 degrees and the next reading is 355 degrees the difference between the two are 353 degrees. However, the true difference is 7 degrees. The reason for this problem is the scale is not linear, after 360 degrees it should go to 1 degree. To resolve this, I created a simple correction function at the pre-processing stage. The correction function takes the two values as input and finds the difference. If the difference is more than 180 degrees it is the incorrect value, so the function returns (360 – difference).

Change of readings - Please refer to table 4 and 5

Using the readings

Table 10 Unchanged Readings Confusion Matrix

Actual		Predicted		
		Standing	Walking	Running
	Standing	315	11	6
	Walking	52	939	57
	Running	28	206	330

Table 11 Unchanged Readings evaluation metrics

	Precision	Recall	F1-score	Support
Standing	0.80	0.95	0.87	332
Walking	0.81	0.90	0.85	1048
Running	0.84	0.59	0.69	564
Avg / total	0.82	0.81	0.81	1944

Comparing the Unchanged readings to the difference of data shows a difference of 10 in the avg/total columns. The reason for this decrease is mostly likely due the orientation as stated in the previous paragraph. The flaw of recording different orientation readings for the same type of movement will cause errors.

5.2.3 Refresh loop

The Refresh loop is 0.5 seconds. This period of time allows for the device to move to another position so that the change of readings can pick up some difference in the sensors. If the refresh loop is too short, there would not be enough time for the movement to take place, meaning the difference between the readings would always be small. If the refresh loop is too long, then movements could be missed out. The activities of walking and running are also movements that happen in a loop, walking is just a cycle of one leg forward and then another. The decision for the time for the refresh loop must take in account how long a cycle of the activities will be. The two readings could be of the same place with the walking cycle, meaning the differences would be very small. I tested a range from 0.5 second to 1.5 second with a step of 0.5 seconds. The result was 0.5 gave the best results.

Refresh loop of 0.5 – please refer to table 4 and 5

Refresh loop of 1.0:

Table 12 Refresh loop of 1.0 Confusion Matrix

Actual		Predicted		
		Standing	Walking	Running
	Standing	170	1	0
	Walking	4	275	39
	Running	0	59	217

Table 13 Refresh loop of 1.0 evaluation metrics

	Precision	Recall	F1-score	Support
Standing	0.98	0.99	0.99	171
Walking	0.82	0.86	0.84	318
Running	0.85	0.79	0.82	276
Avg / total	0.87	0.87	0.86	765

Refresh loop of 1.5

Table 14 Refresh loop of 1.5 confusion matrix

Actual		Predicted		
		Standing	Walking	Running
	Standing	100	2	0
	Walking	2	199	31
	Running	0	36	140

Table 15 Refresh loop of 1.5 evaluation metrics

	Precision	Recall	F1-score	Support
Standing	0.98	0.98	0.98	102
Walking	0.84	0.86	0.85	232
Running	0.82	0.80	0.81	176
Avg / total	0.86	0.86	0.86	510

With these results it is interesting to note that even though the refresh loop of 1.0 and 1.5 gives worse results than 0.5, the results of 1.0 and 1.5 are not too dissimilar. One thing to note, is that for each longer refresh loops, there are less data points due to using the same test data.

5.2.4 Position of Mobile computer

With only one machine to pick up movement of the body, comes the decision as to where to place the machine on the body. The decision was to put it in the front trouser pocket. The machine would then be in front of the upper thigh. There was no difference between which leg, the right or the left. The evaluation behind this decision came from both from the activity and my testing scheme. The activities I wanted to look at were walking, running and standing still. Looking at a human performing these activities, one will notice that the main difference of movement is within the limbs. The short-list of areas for the sensor to be was on the wrists, ankles, or thigh. Out of these three areas, the thigh has the benefit of being near a pocket which allows it to be held in the same position. The other two areas needed to be attached via straps. The ability to hold the sensors in place on the human body is needed for

reliant testing data. If the mobile computer moves not with the body, it picks up incorrect data. This can be seen with the following test results.

Thigh results – Please refer to table 4 and 5.

Ankle results

Table 16 Ankle Confusion Matrix

Actual		Predicted		
		Standing	Walking	Running
	Standing	194	3	0
	Walking	28	240	66
	Running	2	119	108

Table 17 Ankle evaluation Metrics

	Precision	Recall	F1-score	Support
Standing	0.87	0.98	0.92	197
Walking	0.66	0.72	0.69	334
Running	0.62	0.47	0.54	229
Avg / total	0.70	0.71	0.70	760

Wrist results

Table 18 Wrist Confusion Matrix

Actual		Predicted		
		Standing	Walking	Running
	Standing	223	2	0
	Walking	2	181	27
	Running	1	47	165

Table 19 Wrist evaluation metrics

	Precision	Recall	F1-score	Support
Standing	0.99	0.99	0.99	225
Walking	0.79	0.86	0.82	210
Running	0.86	0.77	0.81	213
Avg / total	0.88	0.88	0.88	648

From the results of positions, all work with the activity of standing. Standing has very little movement so the position does not matter much. The activity of walking and running have a lot of movement, with each body part moving a bit differently. Putting the sensor near the ankle gives the worse results, partially with the activity Running. Putting the sensor near the wrist gives better results but not near the thigh. What is interesting to note is that the FitBit charge puts its sensors near the wrist. However, when the sensor is near the thigh it gives the best results.

5.2.5 Choosing the Training model: K means clustering vs Support Vector Machine

Once the training data was recorded, I had thousands of six digit coordinates ranging from (0,0,0,0,0,0) to (180,180,180,10,10,10). The problem was to put a classifier onto this graph and put a line to distinguish the parts of the graph which are a certain activity. For this solution, it was achieved by first using a linear Support Vector Machine to draw a plane within the graph to separate the three activities and then use an outlier Support Vector Machine to draw another plane to separate the three activities to outlier points.

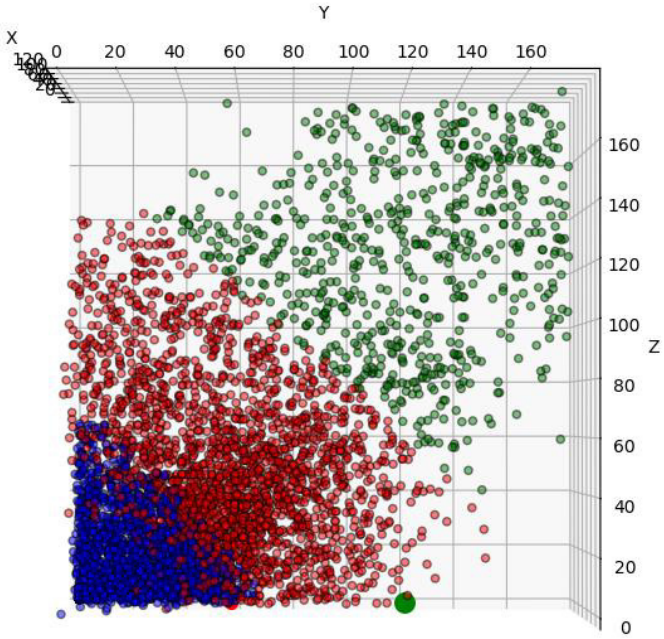
At first, I considered k means clustering. This was because I was trying to work out if this classification problem could actually be a clustering problem. The difference between classification and clustering is that classification draws separating vectors between your data, whilst clustering draws clustroids within your data. Clustroids mean the centre point in each cluster. As the number of activities to be recognised is set to three, it means three clusteroids could be placed into the graph to find three clusters. This solution was first tried manually and using a library. For the manual method all the data are imported into the panda dataobject which represents a six-dimensional graph. Three points within the range of the six axes are chosen at random to be starting clustroids. Then all the data coordinates are given an assigned centroid which is the one to which they are closest. When all data points are assigned, the centeroids are updated to become the average of all the points that were assigned. The loop of assigning the data points and updating the centroids continues until the centroids stay in the same place. This method however drew poor results. In order to make sure it was the clustering at fault and not a coding fault, I used the sklearn clustering function as well. When using the sklearn clustering the program is the same until the data is placed into a Panda Dataframe. But afterwards the Sklean.KMeans function is used with the

input (n_clusters = 3). It uses the method fit () with the dataframe as the parameter. Then the evaluation methods could be used.

Both methods created poor results. My theory into why this is the case is because walking and running overlap too much in terms of movement and thus so would the sensor readings. As the results show, the activity standing still shows the best results. This is because standing still is very different and distinct from the other two activities. This means in the data, there is a clear separation between the two clusters, which standing still having the defined clustering shape. Walking and Running however, are one large cluster, with overlapping boundaries. Therefore, Support Vector Machines were chosen, as the difference of standing still and moving was an easy problem to solve, the harder problem was working out the difference between the different types of movement. Therefore, the problem turned into where to place the separating plane in the overlapping area of running and walking, confirming this problem as a classification.

Below is a visualization of this problem, please note this is just for visualization purposes and is not accurate (Figure 6). The reason for this because a 6-dimensional graph is difficult to visualise so please use this 3-dimensional graph using only the accelerometer sensors. This method is used to understand the problem of there is no clear clustering.

Figure 10. 2D visualisation of clusters



Blue = Standing Still

Red = Walking

Green = Running

Table 20. Clustering Confusion table

Actual		Predicted		
		Standing	Walking	Running
	Standing	0	1826	0
	Walking	0	3087	8
	Running	735	1526	465

Table 21. Clustering Evaluation metrics

	Precision	Recall	F1-score	Support
Walking	0.00	0.00	0.00	1826
Standing	0.48	1.00	0.65	3095
Running	0.98	0.17	0.29	2726
Avg/total	0.54	0.46	0.37	7647

5.2.6 Creating the second outlier Support Vector Machine

Using the Support Vector Machines gives better results than clustering but using only the Class Support Vector Machine gives too many false positives. When the user wearing the mobile computer performs an activity that is none of the three defined activities, the computer would still predict one out of the three activities. This is why the second Support Vector Machine is created using the sklearn.OneClassSVM function with parameters nu = 0.1, kernel="rbf", gamma = 0.0001. This allows the mobile computer to predict an activity that is not one of the three activities. The default kernel is used due as the others are not appropriate.

The evaluation methods of the outlier were different from the main Support Vector machine. The oneclassSVM draws a boundary around all the data points, and marks everything outside the boundary as -1, and everything inside the boundary as 1. This means the data it takes to train on are the same csv files as the Class Support Vector Machine, without the class attribute. This diminishes the advantage of the confusion matrix and Evaluation metrics. There are no test data of an approved outlier when creating the first outlier Support Vector Machine.

The new evaluation methods were to split the inliner training data into inliner_train and inliner_test with a 4:1 split. The program uses inliner_train to train and once it is done it tests

on both the inliner_train and inliner_test. To find the optimal values for nu and gamma I started in the range of 0.05 to 0.2 with range of 0.5 for both.

Table 22. Initial Outlier Results

nu	Gamma	Pass Train %	Pass Test%
0.05	0.05	69	39
0.05	0.1	68	28
0.05	0.15	55	25
0.05	0.2	52	24
0.1	0.05	71	39
0.1	0.1	66	30
0.1	0.15	66	26
0.1	0.2	63	25
0.15	0.05	70	28
0.15	0.1	61	20
0.15	0.15	66	26
0.15	0.2	63	25
0.2	0.05	69	37
0.2	0.1	62	28
0.2	0.15	65	27
0.2	0.2	67	24

These results show that the higher percentage of passing of both the training and testing data relies more on the gamma value. As the gamma value increases, the pass percentage drops. Looking at the nu column, the values 0.1 and 0.15 relate to the highest pass percentage. A new set of ranges is then derived with the aim of increased pass percentage. Nu has a range of 0.1 to 0.15 with step of 0.01 and gamma will have a range of 0.000001 to 0.1 with step of gamma * 10.

Table 23. Outlier Results

nu	gamma	Pass Train %	Pass Test %
0.100000	0.000001	90	89
0.100000	0.000010	89	89
0.100000	0.000100	90	91
0.100000	0.001000	89	88
0.100000	0.010000	83	70
0.100000	0.100000	65	31
0.110000	0.000001	88	89
0.110000	0.000010	89	88
0.110000	0.000100	89	89
0.110000	0.001000	88	89
0.110000	0.010000	83	70
0.110000	0.100000	68	27
0.120000	0.000001	87	87
0.120000	0.000010	87	87
0.120000	0.000100	87	88
0.120000	0.001000	88	87
0.120000	0.010000	83	70

0.120000	0.100000	61	27
0.130000	0.000001	86	87
0.130000	0.000010	87	87
0.130000	0.000100	86	85
0.130000	0.001000	87	87
0.130000	0.010000	82	69
0.130000	0.100000	66	30
0.140000	0.000001	85	86
0.140000	0.000010	85	86
0.140000	0.000100	85	87
0.140000	0.001000	85	85
0.140000	0.010000	82	70
0.140000	0.100000	63	30
0.150000	0.000001	85	88
0.150000	0.000010	85	84
0.150000	0.000100	85	85
0.150000	0.001000	85	83
0.150000	0.010000	81	69
0.150000	0.100000	61	28

From this table the best values, nu = 0.1 with gamma = 0.0001, were picked to be used to create the Outlier Support Vector Machine.

5.2.7 Failed outlier evaluation

As different evaluation methods were used to test the outlier Support Vector Machine, one approach was worked on that ended up not being used. This approach that was used to try to train the outlier Support Vector Machine is a set of uniform points from (0,0,0,0,0,0) to (180,180,180,10,10,10) called “uniform.csv”. Once the outlier takes in a point and produces -1 for an outlier and 1 for not an outlier. Running the outlier on “uniforms.csv” the boundaries can be found, by connecting all the points that are -1 which are next to points that are 1. This creates two ways to predict outliers. The second Support Vector Machine can be put on the mobile computer or the boundary data could be put on the computer. With these two approaches evaluated putting the Support Vector Machine onto the mobile computer was chosen. This was because it did not hinder performance and could give more accurate readings. The boundary data would lose some information as not all points were put under the Support Vector Machine to process, namely the space between the uniform points. If the boundary data is not needed, then the code to build it is not necessary and can be removed from the solution.

6. Future Work

The project was able to predict three activities to an acceptable level whilst also being able to recognise when it did not know the activity. However, several improvements could be made to both the hardware and software. On the hardware side, without the time limit I would be able to create a device that has a physically attached chargeable battery, with only the relevant sensors on board. The Raspberry Pi and Sense HAT had extra features that I did not need (for example, the pressure sensor.) Having to connect to an external battery meant it needed to be connected via wires which made it harder to place in a pocket. My device would also be smaller than the Raspberry Pi (due to not having the extra features), so it would be easier to fit into the trouser pocket. Beyond that, the device could be reduced in size and price so that it could be woven into trousers (or other tight-fitting clothes covering the thigh), turning the product from just a device to smart clothing. This would take a long time and might require a large amount of work.

On the software side, I would work to recognise additional activities. For my next activity I would like to classify falling over, as then the program would turn from being a health application to a safety application. In addition, the main problem in the project I wanted to solve was the classification problem, but with extra work it could have additional functionality to send information over the internet or Bluetooth. This would mean that if it could recognize falling over, it could then notify someone who could help the fallen user. With the connecting functionality it could connect to other applications, like the Apple Health app or a logger on the user's computer. This would still be possible using the Raspberry Pi as it has Wi-Fi capabilities but a new Python module may be needed to send the CSV file over the Wi-Fi.

To improve the model, the Raspberry Pi needs to stop the predicting script, and run the recording script. After the recording is done, the data are manually transferred to the PC where the models need to be re-created with the new data included. To improve this, I would like to improve the predicting script so that it could improve the prediction models alongside prediction. In order to do this, I would turn the outlier prediction model so that after a certain number of outliers recorded, it would turn a new Support Vector Machine model to try to find new activities from these outliers. On the next round of prediction, it would compare it against the new Support Vector Machine so that the new activity would show up as the activity and not an outlier. One problem to include this new activity recognition is that the activity map would need to be updated with the new activity. The current activity map allows the Raspberry Pi to show a letter that represents the activity. If a new activity was created, then there would be no logic to choose the letter to represent the activity. There would still be a need of manual updating.

More work would include removing the personal bias. All the data recorded and tested was on my own body. This means the Support Vector Models would have a bias towards me. To remove this, the training data would increase as the Raspberry Pi would be on Record with multiple people of different body types. This means the prediction would give better test results on different people. To understand how large the problem of personal bias test data was gathered from another consenting person. Below are the results with the support vector machines (created with my data) predicting the actions of test data (created with another person's data).

Table 24 Personal Bias Confusion Matrix

Actual		Predicted		
		Standing	Walking	Running
	Standing	516	2	0
	Walking	2	193	348
	Running	2	235	345

Table 25 Personal Bias Evaluation metrics

	Precision	Recall	F1-score	Support
Standing	0.99	1.00	0.99	518
Walking	0.45	0.36	0.40	543
Running	0.50	0.59	0.54	582
Avg / total	0.64	0.64	0.64	1643

From these results we see that personal bias does indeed have an impact on this project. Not for the activity of standing still as that is an universal activity. The activity of running and walking show poor results. This is because the activity of walking and running are broad. Different people have different paces and styles of running and walking. To improve the model, all styles need to be included in the training data.

7. Conclusions

In conclusion, my project was to create a solution that will support “Activity Recognition on Body Data”. My hypothesis was that a mobile computer with a machine learning classifier could recognise human actions. I then sought to find both the hardware and software solutions. The hardware solution must be both mobile and hold enough sensors to read the movement of the human body. The software solution must be able to classify the sensor’s readings. The hardware solution chosen was the Raspberry Pi with Sense HAT attachment connected to a mobile power source. The second hardware was a high-powered desktop with capabilities to create the classifier. The software solution became an architecture of two programs to run on the Raspberry Pi: Record and Predict. It also took control of the data management, pre-processing and classifying.

Much of the work was with data management and pre-processing compared to the actual coding. A classifier was quick to produce, a good classifier took longer. The training data was improved by doing more examples of each activity and the classifier was improved by exploring different ways to classify each activity.

The main motivation of this project was to capture the uncaptured data of human movement. The results achieved by this solution were acceptable. However, with the outlier activity it cannot work out the severity of the outlier. One motivation of this project was fall detection, when the machine could replace a panic button by calling a person if the user has fallen over. As the mobile machine does not know the difference between falling over and spinning around, it cannot send an emergency signal. One positive outcome of the outlier was removing false negatives in the exercise classifier. As the prediction machine holds the data points for all outliers, one can perform the classified activity (for example running). There should be no outliers in this period but if there are, the points can be placed in running training data. The ability to put false negatives back into the training process is very useful to create a more refined prediction model in supervised learning. Having less false negatives means the percentage of true negatives among all negatives increases. This means that even though the prediction machine cannot determine between the outliers, it is very good at determining if it is an outlier.

Another motivation was to keep the prediction model on a local device and not have to send to an external data centre. This was achieved to a certain extent. Once the two prediction models are created, then they can be moved via file transfer and do not need to be connected back to a central system. This achieves the goal as the prediction is done locally. However, to improve the prediction, both the prediction model and the outlier data (generated by the user) would need to come back to the central computer for the prediction models to be re-created. This has to be done manually so the user must give both their consent and physically give the memory card. No data are sent over the internet. This gives control back to the user and increases privacy as they only need the data shared if they decide that the prediction can be improved.

With regard to the higher goal of exploring the benefits of cyborg organisms, I have broadened the scope of the human-computer interface via an attached computer that can recognise the physical activity that the human body is performing. This adds to the solution of one of the biggest problem two conversing objects can have: that of accurate communication.

8. Reflection on Working

When starting the project, I had very little experience of machine learning and had not written a report of this size. Having little experience in machine learning meant I had limited knowledge of how I was going to implement a solution or know what problem I was trying to solve. I knew you could put a lot of data through an algorithm and it would produce some code that could tell you something about similar data, but I did not know the different types of machine learning algorithms. I thought everything was to do with classifying, with my knowledge stemming from the dog vs cat example: when sending a lot of pictures labelled cat and dog, the computer could tell the difference in new photos. Not knowing the difference between classifier problems and clustering problems was a hurdle in the first couple of weeks as I compared k-means clustering to support vector machines. I now know which solution should be used for which problem.

In this project, unlike my other experience with coding problems, not all the work was in coding, but also in gathering training data. This meant I had to learn how to obtain training data correctly. Learning to obtain training data includes making sure the activity is performed correctly when the device is recording. It is also about how much data should be collected for training data and how much of a split of the data should be turned into test data.

This is the first project I worked on a Raspberry Pi with the Sense HAT add-on, so working under the limitations of the smaller computer was new to me. Previously I would always code on a high-powerful computer, which meant I was never under any restriction in terms of computational power. With the aim of the computer being mobile, it brings along the limitation of being less powerful. I learnt to make sure all my different 3rd party Python packages were the same. I also had to learn the Linux crontab editor to run programs on reboot.

Being a personal project and completed only by myself brought its own challenges. I had the freedom to do what I wanted but I missed the support structure of working in a group. I did have, and am thankful for, my half-hour weekly meetings with my supervisor, but this contact time percentage wise was small in regard to my overall work load. To complete my project, I needed to make use of my connections I had made at my time at university to motivate me. With little contact time and not needing to go to the university buildings, challenged my motivation to complete the project. What kept my interest up was going to optional intellectual meetings, both inside and outside university to keep my motivation high.

Appendix

All training data will be attached within “data.zip”

All the code will be attached with “code.zip”

References

Anon., 2019. *Dictionary.cambridge.org.* [Online]
Available at: https://dictionary.cambridge.org/dictionary/english/big-data
[Accessed 29 April 2019].

Apple, 2019. *www.apple.com.* [Online]
Available at: https://www.apple.com/uk/watch/
[Accessed 07 May 2019].

Ashton, K., 2009. That "Internet of Things" Thing. *RFiD Journal,* pp. 97-114.

Bennett, J. and Lanning, S., 2007. *The Netflix Prize,* s.l.: s.n.

Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Springer,* 20(3), pp. 272-297.

DeepAI, 2019. *DeepAI.* [Online]
Available at: https://deepai.org/machine-learning-glossary-and-terms/neural-network
[Accessed 1 May 2019].

dict., O., 2019. *Oxford Dictionaries | English.* [Online]
Available at: https://en.oxforddictionaries.com/definition/run
[Accessed 29 April 2019].

Dictionary, C., 2019. *Cambridge Dictionary.* [Online]
Available at: https://dictionary.cambridge.org/dictionary/english/activity
[Accessed 04 May 2019].

Dix, Finlay, Adowd & Beale, 2004. *Human-Computer Interaction.* 3rd ed. London: Pearson.

FitBit, 2019. *FitBit Charge.* [Online]
Available at: https://www.fitbit.com/no/charge
[Accessed 07 May 2019].

Google, 2019. *machine-learning/glossary/.* [Online]
Available at: https://developers.google.com/machine-learning/glossary/

HAT, S., 2019. *Raspberry Pi.* [Online]
Available at: https://www.raspberrypi.org/products/sense-hat/
[Accessed 05 May 2019].

Hinton, J. & Sejnowski, T., 1999. *Unsupervised Learning: Foundations of Neural Computation.* s.l.:MIT Press.

RaspberryPi.org, 2019. *Projects.raspberrypi.org.* [Online]
Available at: https://projects.raspberrypi.org/en/
[Accessed 29 April 2019].

Steinhaus, H., 1956. Sur la division des corps matériels en parties. 3(4), pp. 801-804.

Turing, A. M., 1950. Computing Machinery and Intelligence. *Mind,* LIX(236), pp. 433-460.