

Determining and Analysing the Emergent Behaviour from Context-Aware Devices



University of Fort Hare
Together in Excellence

A Dissertation Submitted In Fulfilment Of The

Requirements for the Degree

Of

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

By

Bandason Vivian

SUPERVISOR: Professor Mamello Thinyane

[September 2017]

Abstract

With the continued miniaturization of technology and the incorporation of Moore's law, smartphones are more powerful. These mobile devices contain technologies that add other functionalities to them. Technologies such as sensors constantly provide information to the device. The constant stream of information from these sensors often leads to information overload of relevant and irrelevant information. To work towards solving this problem context-aware computing was introduced. Our major concern is that context information in context aware computing is not completely being utilized. The aggregation of context information could unlock more possibilities. This research seeks to aggregate the context of multiple devices such that, through analysis, some emergent behaviour can be observed.


In this research context information from the sensors of devices is collected using an Android application and a central Server. The context information is used for pattern analysis. A pattern analysis algorithm is designed and used to observe patterns throughout the data set. It shows patterns that are similar within the dataset. In the case that the pattern observed has no similar pattern or few similar patterns this behaviour can be stated as emergent in the dataset. Further study of this emergent behaviour can be performed were a classifier can be used to give the exact activities that were being performed at that time. The research found this was possible and has many uses. One of these is in disaster prevention were the behaviour of a group of individuals may be monitored to observe any random changes such as masses running at the same time. This could be used as a first warning to natural disasters.

Keywords

Context-aware computing, Emergent behaviour, Context Aggregation, Pattern Analysis, Machine Learning, Supervised Learning.

Statement of Original Authorship


I, Vivian Bandason do hereby confirm that all the work contained in this dissertation has not been submitted for any other qualification at this or any other University. Furthermore I confirm that the dissertation does not contain any published information except where due reference has been indicated.

Signature: 

Date: 22- September 2017

Plagiarism Declaration

I.....Vivian Bandason.....student number.....201104622.....hereby
declare that I am fully aware of the University of Fort Hare's policy on plagiarism and I
have taken every precaution to comply with regulations.

Signature:..........

Acknowledgements

I would like to thank my father in heaven the Lord God Almighty. "I can do all things through Christ who strengthens me." (Philippians 4:13). He has kept me strong and focused all this time.

I would also take this opportunity to thank my supervisor Professor M. Thinyane for guidance, support and vast knowledge during the duration of this project. Special thanks go to the H.o.D of the Department of Computer Science for the opportunity he has given me to study in the department and for his continuous support during the project. Thank you to my class mates for the support, laughs and motivation. Heartfelt thanks to Edmore Chindenga who has been an outstanding friend and more of a brother to me since first year.

I would like to thank my family for the support during this project. My mother for constantly being there and supporting me, I could not have done this or got this far without her, I love you ma. My aunts for the support during and prayers through the academic years.

This work was done under NRF and they have supported my academic pursuit and this helped me immensely through the two years.

Table of Contents

Chapter 1: INTRODUCTION	1
1.1 Background.....	1
1.2 Problem Statement	2
1.3 Research Questions.....	3
1.4 Research Aims.....	3
1.5 Research Objectives	3
1.6 Motivation.....	4
1.7 Organization of Dissertation	4
1.8 Conclusion	5
Chapter 2: BACKGROUND AND RELATED WORKS	6
2.1 MOBILE PLATFORMS.....	6
2.1.1 Android Platform	7
2.1.2 iPhone Operating System (iOS).....	9
2.1.3 Windows Phone.....	10
2.2 CONTEXT-AWARE COMPUTING.....	11
2.2.1 Context-Aware Systems architecture.....	14
2.2.2 Context Management Models	16
2.3 Machine Learning	19
2.3.1 Supervised Learning.....	22
2.3.2 Classification.....	24
2.4 Swarm Intelligence.....	37
2.5 Pattern Analysis algorithms.....	41
2.5.1 K-Means	42
2.5.4 CHAMELEON	43

2.6 RELATED WORKS	43
2.6.1 Mobile crowd-sensing in the Smart City.....	43
2.6.2 Waze	45
2.6.3 Anagog	46
Chapter 3: METHODOLOGY	48
3.1 Definition	48
3.2 Overview of the Methodology.....	49
3.2.1 Research Conceptualization	51
3.2.2 System Requirements	51
3.2.3 System Design.....	51
3.2.4 Data Collection	52
3.2.5 System implementation and testing.....	53
3.2.6 Data Analysis and Results	54
3.3 Summary.....	54
Chapter 4: DESIGN AND IMPLEMENTATION	55
4.1 Research Design	55
4.2 System Design.....	58
4.2.1 System Requirements	59
4.2.2 Interface Requirements.....	59
4.2.3 Data Collection	60
4.2.4 System Architecture.....	61
4.2.5 System Use Cases	64
4.3 Testing and Results	67
4.4 Mobile Application	67
4.4.1 Development Platform	68

4.4.2 Android Activity	70
4.4.3 StartSensors Method	73
4.4.4 StopSensors Method	79
4.4.5 Manifest File	80
4.5 The Context Aggregation Server.....	80
4.6 Classification.....	85
4.7 Pattern Analysis	88
Chapter 5: TESTING AND EVALUATION.....	92
5.1 Classifier Testing	92
5.1.1 Classifier model complexity	93
5.1.2 Classifier Sample size	95
5.1.3 Classifier Dimensionality of features.....	97
5.2 Pattern Analysis Testing	99
5.3 Discussion.....	103
5.3.1 Model Complexity	104
5.3.2 Sample Size.....	104
5.3.3 Pattern Analysis.....	105
5.4 Conclusion	105
Chapter 6: CONCLUSION AND FUTURE WORK	107
6.1 Mapping of Findings to Research Questions.....	107
6.2 Recommendations and Future Work	109
6.3 Limitations.....	110
6.4 Conclusion	110
REFERENCES.....	111

List of Figures

Figure 2-1 Mobile platforms overview through the years (Renner 2011).....	7
Figure 2-2 The layers of the Android architecture (Meier 2010).....	8
Figure 2-3 Architecture of iOS (Okediran et al. 2014)	10
Figure 2-4 Windows phone architecture (Raghavendra & Vasantha 2016).....	11
Figure 2-5 Classification Framework using the architecture.....	16
Figure 2-6 Example training set passed through the hypoDissertation (N. Nillson 2005)	20
Figure 2-7 The training sets associated with each learning algorithm (Ayodele 2010)..	21
Figure 2-8 Workflow of a Supervised Learning algorithm (Paco 2014)	23
Figure 2-9 Classification process of Salmon and Sea bass (Duda et al. 2001).....	25
Figure 2-10 Process of feature selection before classification (Tang et al. 2014)	26
Figure 2-11 Learning and testing of algorithm (Tan et al. 2006).....	27
Figure 2-12 Simple decision tree structure (N. Nillson 2005)	28
Figure 2-13 Univariate and Multivariate splits (Weis et al. 2009)	29
Figure 2-14 Hyper-plane fitting (Joachims 2002)	31
Figure 2-15 Non-linearly separable data (Fletcher 2009)	33
Figure 2-16 Difference of classification in low dimension and high dimension (Paco 2014)	33
Figure 2-17 Transformation of input data (Joachims 2002).....	34
Figure 2-18 Polynomial function (Paco 2014)	35
Figure 2-19 RBF kernel separation (Paco 2014).....	35
Figure 2-20 Shows the training instance x within a circle showing $k = 5$ (Duda et al. 2001).	36
Figure 2-21 Shows the 1NN algorithm where the closest training instance is chosen (Duda et al. 2001).	37
Figure 2-22 Ants crossing the twig example (Eberhart et al. 2001b).....	38
Figure 2-23 sudo-code for K-Means (Duda et al. 2001)	42

Figure 2-24: The Smart Citizen website (Lendák, 2016).....	44
Figure 2-25: Screenshot of waze application.....	46
Figure 2-26: Anagog parking application (Lendák, 2016).....	47
Figure 3-1 Waterfall Model stages (Davidson 2004)	49
Figure 3-2 Incremental and Iterative Development (Davidson 2004)	50
Figure 3-3 Evolutionary Prototyping Model	53
Figure 4-1 Research Onion (Saunders & Tosey 2013)	56
Figure 4-2 Data collection stages.....	61
Figure 4-3 System Architecture	62
Figure 4-4 Starting the Application	65
Figure 4-5 Server interaction.....	66
Figure 4-6 Graph of sales of different Devices	68
Figure 4-7 Graph of platform market shares	69
Figure 4-8 Android Activity LifeCycle.....	71
Figure 4-9 Screenshots of application	72
Figure 4-10 Screenshot of running server	80
Figure 4-11 Server receiving context information.....	81
Figure 4-12 Extract of database information	82
Figure 4-13 Table Created by Clustering Algorithms.....	90
Figure 5-1 Confusion matrix of each of the classifiers.....	94
Figure 5-2 Graph of performance of classifiers	97
Figure 5-3 Graph of all points against the X, Y and Z axis of the accelerometer.....	100
Figure 5-4 Patterns in data set and the similar patterns	101
Figure 5-5 Multiple similar patterns	102
Figure 5-6 Multiple predictions showing where the pattern could go.....	102
Figure 5-7 Pattern recognition using 70% accuracy.....	103

List of Tables

Table 1: Mapping of Research Questions to Objectives.....	58
Table 2: Model accuracy	93

Table 3: Accuracy scores of models with cross validation.....	94
Table 4: Accuracy of models using different parameters for each classifier	95
Table 5: Changing training set size	96
Table 6: Accuracy scores using accelerometer only compared to all features	98
Table 7: Accuracy scores using light and proximity only compared to all features.....	99
Table 8: Accuracy scores using accelerometer X and Y axis only compared to all axis	99

List of Listings

Listing 1 Pseudo code for Decision Trees (Kotsiantis et al. 2006.....	32
Listing 2 Declaration of special variables	72
Listing 3 Location listener	74
Listing 4 Registering all sensors	75
Listing 5 OnSensorChanged method	76
Listing 6 Updating of Views	76
Listing 7 Runnable Thread	77
Listing 8 Post Method	78
Listing 9 Stop sensors method	79
Listing 10 Permissions used	80
Listing 11 Aggregating context	83
Listing 12 Conversion of table to dataset	86
Listing 13 Imports for Classifiers	86
Listing 14 Decision Tree model fitting	87
Listing 15 Scores used for testing	88
Listing 16 Code snippet of clustering algorithm	89
Listing 17 Code snippet of calculation of similar patterns	91

List of Acronyms

ACO - Ant Colony Optimization

AI - Artificial Intelligence
AOSP - Android Open Source Project
API - application programming interface
BPA - Behaviour Pattern Analysis
BSD - Berkeley Software Distribution
DT - Decision Tree
DVM - Dalvik Virtual Machine
GPS - Global Positioning System
GUI - Graphical User Interface
HTTP - The Hypertext Transfer Protocol
IDE - Integrated Development Environment
IDLE - Integrated Development Environment
JSON - JavaScript Object Notation
kNN – K-Nearest Neighbours
MS - milliseconds
NP - Non-deterministic Polynomial-time
OS - Operating System
OSGI - Open Service Gateway Initiative
PSO - Particle Swarm Optimization
RBF - Radial Basis Function
RF - Radio Frequency
SDK - Software Development Kit
SVM - Support Vector Machines
URL - Uniform Resource Locator
VM - Virtual Machine
WSGI - Web Server Gateway Interface
XML - eXtensible Mark-up Language

Chapter 1: INTRODUCTION

This chapter introduces the nature of the problem that the research is set to address. It discusses the background of the problem and highlights the research aims and objectives. The chapter sets a tone for the rest of the research and also outlines the research organization.

1.1 Background

Continuous changes in Information and Communications Technologies (ICTs) have led to the rapid change in the use, capabilities, and shape of computers. Numerous changes have been observed from the traditional desktop computers with minimum capabilities to the advent of mobile devices capable of a great number of functions (Costa et al. 2007). Computers and mobile devices tend to incorporate more and more technologies adding to the already large set of functionalities available in these devices. These technologies inaugurate new ways of accessing and providing information and services to users.

These new technologies include sensors housed within the devices. The sensors work together with the software to provide information to the device. These sensors can be found on most mobile devices currently being released on different platforms (Beach et al. 2010). The sensors are capable of recording motion, orientation, and various environmental conditions which change regularly as the device is being used (Nadeem 2012). Although the addition of sensors opens new horizons on the capabilities of mobile devices, it also serves as a problem. The constant stream of information from these sensors leads to information overload, making it difficult to distinguish relevant information and irrelevant information (Schmidt et al. 1999).

In an attempt to solve the above-mentioned problem, context-aware computing was introduced. Context-aware computing is the design and implementation of systems that adapt themselves to the ever-changing context of the user, device and surrounding environment (Hong et al. 2009). However, not all context information is relevant to a particular application. Context-aware applications are designed to collect information useful to the user based on the type of application being used.

The introduction of agents to context-aware computing can bring a solution in which synergy between the context-aware system and the user-relevant information is created. Context-aware agents are presently being implemented as recommender systems and other similar applications, but their actual intelligence is limited (Abbas et al. 2015). Context information collected by a single agent from its sensors has limited uses and applications, However, useful results can be achieved by bringing together multiple agents that can maintain their individuality while their context information is aggregated to achieve some goal. The aggregation of the context-aware agents can be used to find patterns and emergent behaviours of the agents. These patterns and behaviours can be analysed characterize user-activities.

1.2 Problem Statement

The awareness of the physical environment surrounding a user and their mobile or ultra-mobile device has been one of the prevalent concerns in context-aware computing. In recent work, this concern has been addressed by implementation of systems that use the sensors embedded on the devices. This has improved the way that context-aware applications on mobile devices function. Although this problem has been addressed the amount of contextual information that is extracted from a device can only be used in a limited number of ways. Expanding the scope of the applications of the collected contextual information can further understanding in context-aware computing. By aggregating contextual information collected from a multi-agent context-aware system, contextual information can be analysed to deduce emergent behaviour that can be associated with a group of agents in a particular area. This research seeks to develop a system to aggregate contextual information derived from a multi-agent system in a way to better understand emergent behaviours.

1.3 Research Questions

The research activities are guided by the main research question which is stated as follows;

“How can emergent patterns and behaviours of a context-aware multi-agent system be predicted based on the agents’ contextual information”.

The main research question is decomposed into the following sub-questions that will be addressed by this research:

1. What models of context-aware computing are currently available?
2. What context can be found using sensors on a device and how can an application harness this contextual information be developed?
3. How can the contextual information from different devices be collected and aggregated using a central service?
4. How can emergent behaviours and patterns be mapped to certain context?

1.4 Research Aims

The aims of the research are stated below:

1. To design a context aware application.
2. Use a user’s context to observe their behaviours.
3. Find a means to aggregate and observe behaviours of users.

1.5 Research Objectives

The specific objectives for the research are as follows:

1. Analyse context-aware computing models.
2. Design an application capable of acquiring information from a device’s sensors.

3. Design and implement a system that collects the context from multiple agents and aggregates the contextual information.
4. Analyse the emergent patterns that result from the analysis of the contextual information associated with the multi-agent system.

1.6 Motivation

The analysis of the context information of an individual device yields little information. It does not show the interactions of different individuals with others. An aggregation of context information from interacting individuals can yield more information regarding the environment or how individuals behave in certain contexts. The behaviours that emerge from the analysis of these agents in their specific contexts can lead to analysis of group behaviours in certain settings. This research can allow for work and social behaviour which are affected by several environment influences such as deadlines, the nature of your work, work relationship, responsibilities at home, relationship with family members and colleagues - to name a few to be observed in the context of the groups who work or associate with each other (Baer et al. 1968).

Behaviour Pattern Analysis (BPA) helps demonstrate the ability to flex or maintain consistent behaviour in various situations by identifying various patterns and styles that one assumes in different settings. This can also be useful in comparing work and personal behaviour styles leading to an analysis of reasons for conformity or lack of it.

1.7 Organization of Dissertation

The remainder of the dissertation consists of six chapters which are arranged as follows:

- **Chapter Two: Literature Review** – the chapter reviews literature on Mobile platforms, Context Awareness, Machine learning, Swarm intelligence and Pattern analysis.

- **Chapter Three: Methodology** – this chapter discusses the methodologies used for the research and why these were chosen.
- **Chapter Four: Design and Implementation** – this chapter discusses the research design and implementation of the system; the research design gives the techniques used to carry out the research. The implementation discusses and shows how each component of the system was coded.
- **Chapter Five: Results and Discussion** – details the tests performed on the system and the results associated with the tests. The chapter then discusses the results given by the system.
- **Chapter Six: Conclusion and Future Work** – gives the conclusions of the research, the recommendations and makes suggestions for future work.

1.8 Conclusion

This chapter presented the research focus. It outlined the background of the research problem and research motivation. It also outlined the research questions and objectives. The chapter concluded by outlining the dissertation structure.

Chapter 2: BACKGROUND AND RELATED WORKS

This chapter begins by presenting different mobile platforms available and gives the background behind context awareness and its models. It further describes the different machine learning techniques and swarm intelligence and concludes with the pattern analysis algorithms available. This chapter helps give the scope of the research work. The related works section looks at work that can be found that has been done by other authors and is relevant to the research. The work is analysed and then criticized in relation to this particular piece of work. The related works are used as stepping stones by observing how the work was done and how this may be useful in the development of this research.

BACKGROUND

2.1 MOBILE PLATFORMS

The growth of the mobile phones market has led to a subsequent increase in the number of mobile platforms. Mobile platforms are operating systems that are designed to run on mobile devices. The rapid increase in sales of smartphones in the last few years has spurred the emergence of multiple operating systems for mobile devices. This has led to an increase in competition in this market (Lettner et al. 2012). Many platforms are available for different mobile phones. Figure 2-1 shows some of the mobile platforms. Amongst those shown, Android, iOS and Windows phone platforms have dominated the markets throughout the recent past few years and are probably the most relevant (Renner 2011). For the purposes of this research, we discuss some the relevant mobile platforms.

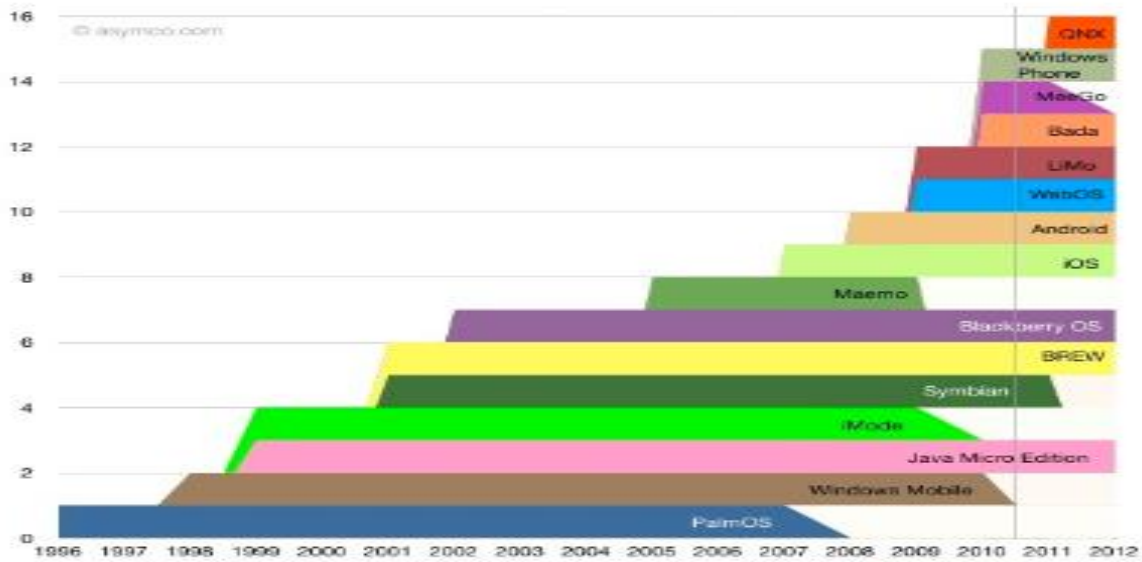


Figure 2-1 Mobile platforms overview through the years (Renner 2011).

2.1.1 Android Platform

The Android platform is an operating system based on the Linux 2.6 kernel. The kernel is used for hardware abstraction and is responsible for security, memory and process management (Raghavendra & Vasantha 2016). The Android platform is currently developed by Google and it is the most popular mobile operating system as of 2013 (Meier 2010). The Android operating system source code is released by Google under open source licenses (Meier 2010).

Most of the devices that run this operating system come with the custom software but also have some of the proprietary software. Some of the devices that run this operating system are cell phones, tablets, high definition televisions, microwaves and washing machines.

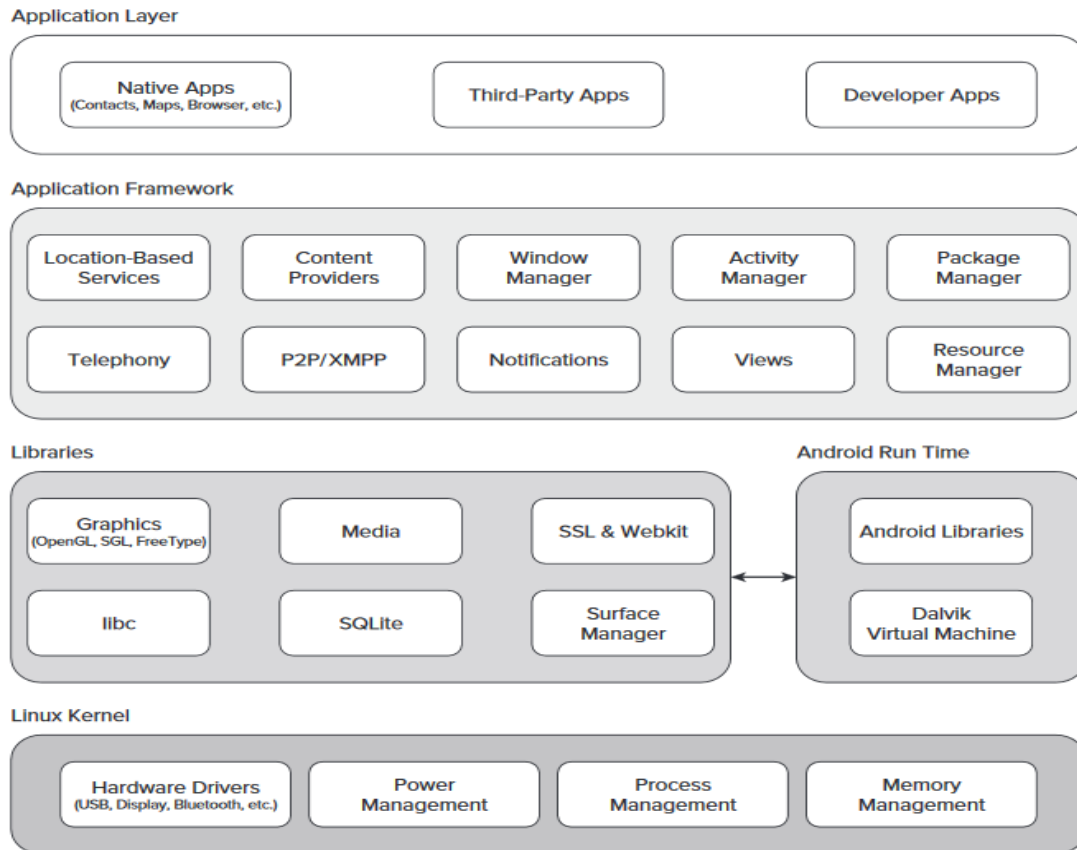


Figure 0-2 The layers of the Android architecture (Meier 2010).

The operating system is composed of many different layers that make up its architecture. The components of the operating system are as follows:

- Linux kernel
- Open Source Libraries
- The Android Runtime
- Application framework
- Applications

The open source libraries are coded using C or C++ and are responsible for application development, including SQLite, WebKit and OpenGL (Meier 2010). The Android runtime is where the core libraries are found and it provides JAVA language functionalities that Android applications are mostly based on. The Android runtime uses a virtual machine known as the Dalvik Virtual Machine (DVM). The DVM has a feature which allows for the

concurrent running of virtual machines (Gopalakrishnan et al. 2012). The application framework is a layer which provides the essentials for application development (Renner 2011). It contains important classes such as the Activity manager, Location manager, and the Resource manager (Gopalakrishnan et al. 2012). The application layer is found at the top of the stack and is made up of the applications that run on the operating system (Renner 2011).

2.1.2 iPhone Operating System (iOS)

The iOS is the mobile operating system that is used on Apple's iPhone series of devices. It is a UNIX-based system which is the same as the Mac OS X. This means it has a Mach kernel and BSD interfaces (Lettner et al. 2012). Unlike the Android platform, iOS is only available exclusively to Apple devices. The iOS operating system is proprietary software owned exclusively by Apple. The iOS promoted direct manipulation of the user interface using touch gestures such as swiping, pinching and tapping (Okediran et al. 2014).

The architecture of the IOS operating system consists of the following layers (Renner 2011):

- Core OS
- Core Services
- Media Layer
- Cocoa Touch

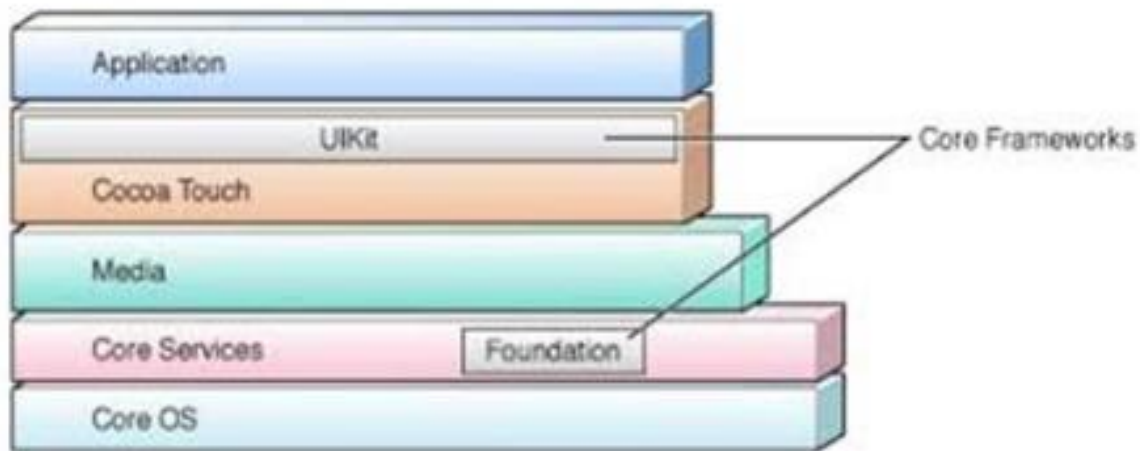


Figure 0-3 Architecture of iOS (Okediran et al. 2014)

The Core OS is the kernel of the operating system. The kernel includes functionalities such as file input/output, network sockets, hardware management, memory management, security, and system support (Okediran et al. 2014)(Gopalakrishnan et al. 2012). The Core Services are the core system services that are offered and are divided into different frameworks based on C programming language and Objective C. The Core Services is involved in providing the basic application services, such as accounts, managing contacts, data management, location, calendar events, store purchasing, SQLite, and XML support (Okediran et al. 2014).

The media layer is involved in rendering both 2-Dimensional and 3-Dimensional graphics. It is also used by the operating system’s audio and video technologies (Gopalakrishnan et al. 2012). Cocoa touch is an Objective C based framework that includes fundamental infrastructure that is used by applications (Okediran et al. 2014). It also provides different APIs that are used for application development (Gopalakrishnan et al. 2012).

2.1.3 Windows Phone

Windows phone is a mobile operating system that was developed by Microsoft succeeding the previous operating system, Windows mobile. It was developed for the consumer market with various mobile phone manufacturers developing mobile phones to run this operating system (Gopalakrishnan et al. 2012). The operating system makes use

of tiles to immerse the users in the application experience (Raghavendra & Vasantha 2016).

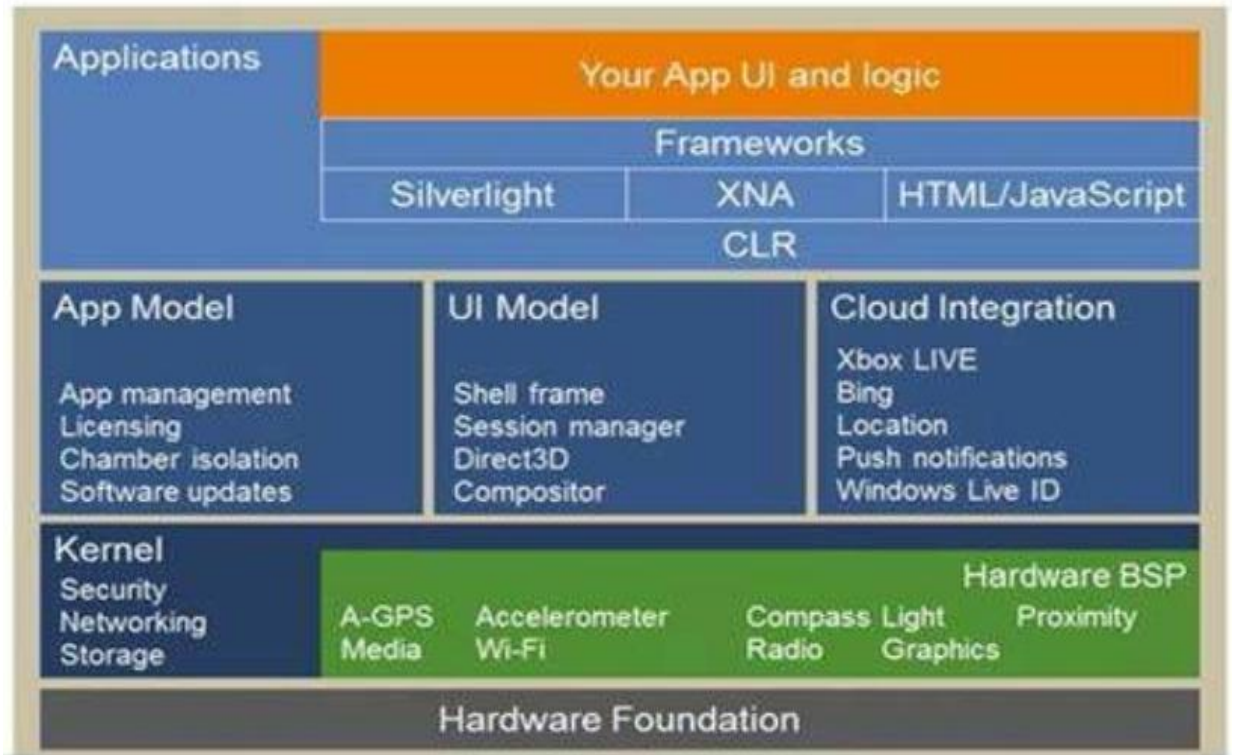


Figure 0-4 Microsoft Windows phone architecture (Raghavendra & Vasantha 2016)

2.2 CONTEXT-AWARE COMPUTING

Presently, the emerging Ubiquitous Computing technologies tend to offer a type of computing that can be known as ‘anytime, anywhere, anyone’ computing (Schmidt et al. 1999). Weiser’s vision of ubiquitous computing had two major observations. The first observation was that technologies that can work in the background as they are used were doing well, as they were part of the world we live in but out of sight. This means the separation of users and their physical devices instead of having different entities namely, the user, the device, and the environment.

The second observation was the effects of Moore's Law on technologies in pursuit of making computation small and cheap enough that a new economic and technical model for computation could emerge.

A user can no longer be tied to one device in one particular location as found with desktop computing. This made it possible to view a world where our day to day environment would be full of low power and low cost technologies (Weiser 1991). Currently, computing is done on the move and relates to how Weiser saw that his two observations were heading to a similar conclusion.

Depending on the situation and the final use of the context, there can be various definitions of what context actually is and different researchers have come up with their own definitions of context. Context can be expressed by the user's location, the people around a particular user, the temperature in the user's environment and the time of day. Dey et al (2001) (de Oliveira & Loureiro 2011) further define an entity as a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.

From this, we can breakdown context to three main sections, the computer environment, the user environment and finally the physical environment. The computing environment refers to the processors used and processing speed, the actual input devices accessible to the user and display and whether these are on the same device or separate, network bandwidth, connectivity and the cost of this bandwidth and computing. The user environment includes everything that directly involves such as the user's location, people surrounding the user and groups associated with that the user is in if any. The physical environment involves measurements recorded by the sensors such as the light, sound, and temperature

Context can also be explicitly defined as any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects and encompasses the above mentioned definitions.

A user's context can be composed of many different aspects each consisting of attributes such as user location, different conditions of their body, personal history and different patterns. We can further address context by focusing on how it is used in computing. The use of context on a daily basis is known as context-aware computing. Context awareness refers to the properties of a system that make it aware of its user's state and surroundings and help it adapt its behaviour accordingly.

A context-aware system can also be defined as capable of extracting the current context, interpreting it and using that context information then also adapting to the context information (Karypis & Han 1999). Location, identity, time, and activity are the primary context types for characterizing the situation of a particular entity. These context types not only answer the questions of who, what, when, and where, but also act as indices into other sources of contextual information (Dey et al. 2001). The identity of an individual and other relevant context information can be inferred to build a better understanding of the user such as information from social media, the individual's birthday, and their connections with other individuals in their biophysical space.

The location service allows for the discovery of the entities that are within the proximity of that particular entity and the kind of activities occurring close to it. Time allows for time-sensitive context such as notifications to the user about meeting or appointments that are found in the user's schedule, for example, a user goes to work then the grocery store on weekdays while on weekends they go for golf at a certain time also. Events may also occur together, for example the user may call some of his friends when they go for golf to confirm who is coming (Mäntyjärvi 2003).

The secondary pieces of context share a common characteristic: they can be indexed by primary context because they are attributes of the entity with primary context. For example, a user's phone number is a piece of secondary context and it can be obtained by using the user's identity as an index into an information space like a phone directory (Dey et al. 2001). Primary context is then a source of inference to find secondary context and this allows for rich context data in a context-aware system

A System is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. There is a

misconception between context-aware systems and self-modifying systems (Gangam 2009). Self-modifying systems are software systems which achieve their goal by rewriting themselves as they proceed with their computations. It is almost universally considered harmful now, but it used to be a neat trick in the toolbox of programmers of small or old computers (Tropeano 2006). Context-aware systems are meant to acquire and analyse context of a device in order to provide services that are appropriate to the environment (Hong et al. 2009).

They must sense more dynamic information in real-time from the user's environment including things such as the users state, their location and orientation and people that may be of relevance to the user (Dey et al. 2001). This is best observed in mobile devices that are being released integrated with sensors that can be used for this particular purpose. Mobile smartphones have been enhanced with a variety of sensors, such as accelerometers, microphones, cameras, and even digital compasses that can be mined to infer actions or even orientation. These sensors found on these devices can then be used to acquire context from the device's surroundings.

2.2.1 Context-Aware Systems architecture

The common approaches to designing context-aware systems are Direct sensor access, Middleware Infrastructure approach and Context Server approach.

Direct sensor access

This is one way of acquiring context data whereby the devices have the sensors embedded within the devices. This means the context-awareness system on the client device can collect the desired context information from the sensors directly. In this approach, there is no middleware component that is responsible for the processing of the context information that is collected such as standardizing and sorting of the context information (Baldauf et al. 2007). This approach cannot be implemented in distributed systems as it completely goes against the foundations of distributed computing because the approach enforces strong coupling between the sensors, sensor drivers and the software communicating with the sensors.

Context server

This distributed approach extends on the middleware infrastructure by introducing an access-managing remote component. This allows different clients to be able to use data from remote sources. The sensor information is collected and moved to the context server allowing for simultaneous access from different devices. This approach has two major benefits; reusability of sensors by all clients accessing the context server and relieves clients from carrying out resource intensive functions as this is carried out by the server (Baldauf et al. 2007). This approach is useful in context-aware systems that incorporate devices with limited computational power such as mobile phones and tablets.

Middleware infrastructure

This approach is based on the encapsulation software design method where functions and components are separate entities in the system communicating via messages. The approach makes use of a layered architecture. This means data is processed at each layer until it reaches the highest layer, usually the Graphical User Interface (GUI). Unlike the Direct sensor access approach, this approach allows for a more flexible system since the layers are loosely coupled and allow for reusability (Baldauf et al. 2007). The middleware infrastructure is usually composed of the following layers; the Network Layer, Middleware Layer, Application Layer and the User-Information Layer.

1. The Network Layer involves a network supporting context-aware systems and sensor collecting low-level of context information.
2. The Middleware Layer manages processes and stores context information.
3. The Application Layer provides users with appropriate service.
4. The User Infrastructure Layer manages the interface of context-aware systems to offer a suitable interface to users.

Figure 2-5 represents the architecture used for a classification framework developed for classifying the literature related with context. The framework uses the general layers of the architecture and at each level, the activities involved are shown. Network layer involves protocol, sensing, network requirement and network implementation.

Middleware layer is categorized into agent-based middleware, metadata-based middleware, tuple space-based middleware, Open Service Gateway Initiative (OSGI) based middleware, reflective middleware and sensor selection middleware (Dey et al. 2001).

The application and service layer are divided into the smart space, tour guide, information systems, communication systems, mobile commerce and web services. User infrastructure layer consists of interface and usability categories.

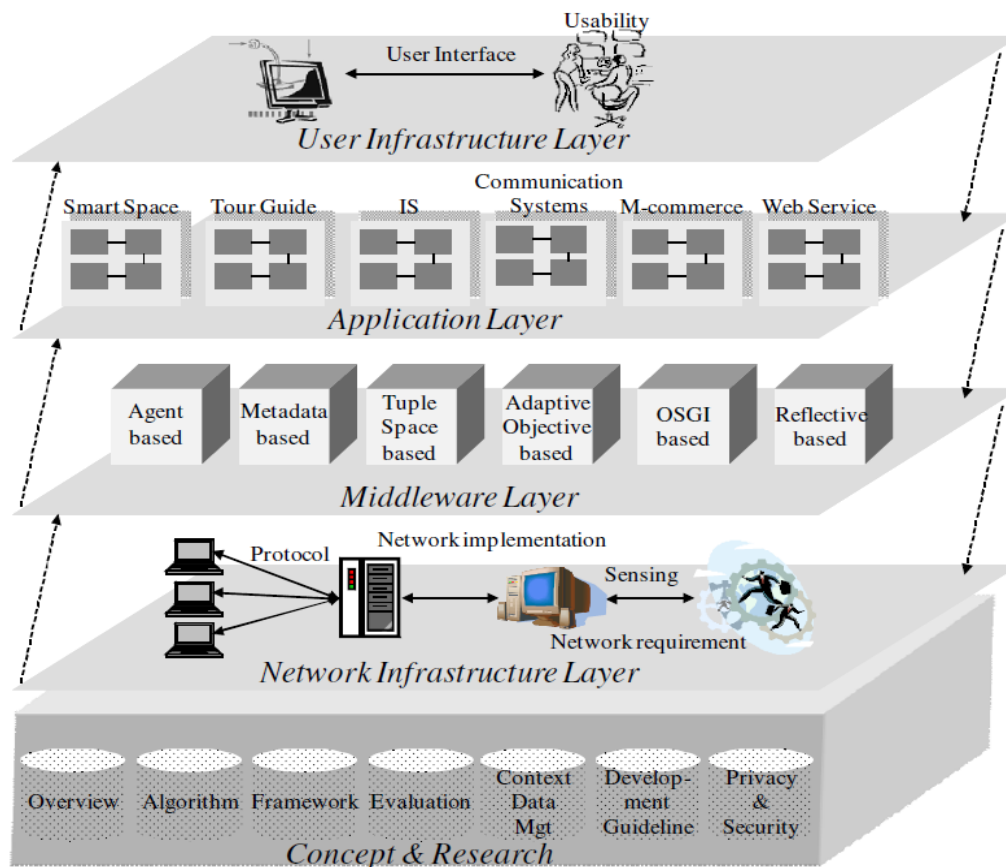


Figure 2-5 Classification Framework using the architecture (Dey et al. 2001).

2.2.2 Context Management Models

There are three widely used models for managing multiple processes and components. These models have drawbacks when it comes to efficiency, configurability, and

robustness. All technology has an efficiency matrix that it has to be compared to, in context-aware computing, efficiency usually refers to the time efficiency of the system or model. Some models impose many layers which delay the request-response turnaround time, while others can create fast paths and give responses with minimal lag. However increasing processing speed in most devices abstracts this drawback.

Configurability is more difficult to measure as it does not follow Moore's law. Configurability can be somewhat defined as the effort required to configure systems that include multiple processes and devices. In complex systems, the components work effectively once they have been configured, but adding or modifying components is complicated and prone to breakdowns. As computing has moved towards a network-centric environment with dynamic addition and removal of processes, re-configurability has become an increasing concern (Winograd 2001).

Robustness refers to the ability of a computer system to cope with errors during execution. Robustness can also be defined as the ability of an algorithm to continue operating despite abnormalities in input or calculations. In this case, the system must also be able to function with abnormalities since contextual information is very dynamic.

The first model is known as widgets. Widgets give a process-centric view and are known to be the most efficient model yet not robust to component failures. The second model is known as network services. Network services resemble context server architecture and consist of a service oriented model. Network services model is not as efficient as the widgets model but it is more robust. Lastly, the Blackboard model provides a data-centric view and requires a centralized server to host the Blackboard.

Widgets

These are GUI elements that can be found on devices. Widgets abstract the functionalities of the input device and in this case the sensor. A widget can also be defined as a software component that provides a public interface for a hardware sensor (Baldauf et al. 2007). They do this by allowing a separation of components of the application hiding the complexity of the actual sensors. Due to the encapsulation found in widgets, it is possible to exchange widgets which provide the same kind of context data.

An example can be the location of a user that can be sensed using Active Badges floor sensors, a radio frequency (RF) based indoor positioning system or any combination of these positioning methods. The use of either of these methods should not affect the actual functionalities of the system or application. Widgets also allow for the abstraction of context information such that it is presented in a suitable manner. The manner of presentation is dependent on the design choices made by the programmer, less significant changes in a user's context can be ignored and not reported while more significant changes can then trigger a response of some sort by the widget (Dey et al. 2001).

Widgets do not need to show the low-level details of each of the sensors being used. Instead, they give the sensors as abstract devices with certain properties which allow for reusability during application development. This abstracting of low-level details means sending messages and notifications to widgets and callbacks is done in the same way and the implementation of distinct notifications is not necessary.

Network Services

Unlike the centralized approach used in the widgets model, network services represent the context server architecture and give a more flexible approach compared to widgets. This model uses discovery techniques to find the networked services which are unlike the Widgets whereby a widget manager is used to keep track of all active widgets. This service based approach is also dependent on complex network based components that make the model additionally expensive than others when it comes to time and communication. With careful selection of specialized protocols this can be surpassed but is a factor to consider (Baldauf et al. 2007).

Blackboard model

The blackboard model is characterized by its data-centric view which is unlike the other models previously discussed which are process-centric and service-oriented. Unlike the service-oriented model where requests are sent to distributed components and subsequent responses are received from them, a process posts messages to a shared media or message board, known as the Blackboard, and can subscribe to receive new

messages matching a specified pattern. All messages pass through the Blackboard and are managed by a Blackboard manager making communication between components less complex than in a client-server architecture (Korpipää 2005). The benefits of the model are that context sources can be easily added and the configuration is also very simple. One drawback is that a centralized server is required so as to host the Blackboard. Time and resource usage is inefficient as two hops per communication are necessary from source to Blackboard then back to the source or other destination (Baldauf et al. 2007).

2.3 Machine Learning

Machine learning is a growing field in computing that has become a vital part of everyday activities. Machine learning is subtle yet effective in the use of the internet, for instance when running Google searches. Google uses web page ranking which involves learning which pages are relevant to the search and arranging them in an order from most relevant to least relevant. Another use of machine learning is in online shopping. In this case, the system learns from previous purchases and uses these to recommend a new purchase that may be useful to the user. These are just a few examples of the uses of machine learning. The keyword used in these examples is that the system must “learn”. What then is machine learning?

Machine learning is mainly concerned with the question of how to implement systems that will improve their performance over time without being explicitly programmed (Mitchell 1997). A machine can also be said to learn if it changes its internal workings such as program or structure due to the inputs it receives or other factors and its expected performance improves over time (N. Nilsson 2005). A broad definition of a machine learning states “a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E” (Mitchell 1997). A machine learning algorithm must take in data that will allow it to gain the experience E.

The data must be in the form of a feature vector. A feature vector is in the form of a vector containing the features that will be processed by the model “h”. Figure 2-6 shows the process of learning observed from the program. The final product will be $h(X)$ which is the hyperplane. The hyperplane is thought to come from the larger set of functions H and is chosen to best fit the data (N. Nilsson 2005).

Machine learning draws inspiration from many different fields, for example, Biology, Artificial Intelligence, and Statistics. Machine learning is closely associated with the field of artificial intelligence (AI). In AI “agents” may perceive their environments and decide on their set of actions. This perception of their environment can be regarded as a way of learning.

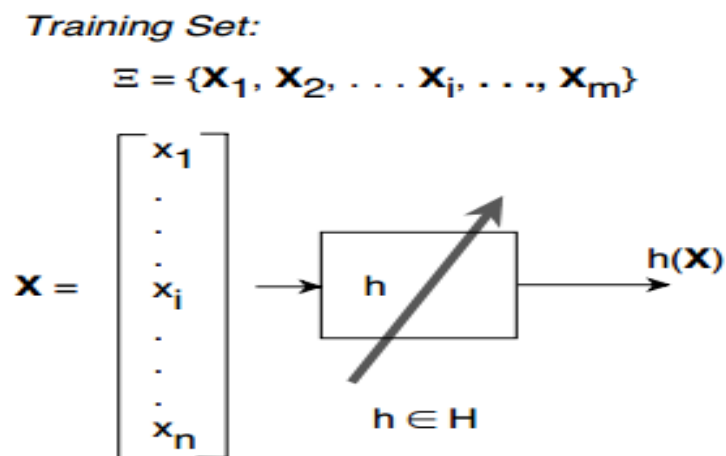


Figure 0-6 hyperplane being fit to a training set (N. Nilsson 2005)

In machine learning, there are many types of learning algorithms that use different methods depending on the data available, some of these are listed below:

1. **Supervised learning** – the algorithm is responsible for creating a model that best maps the inputs to the outputs. An example of supervised learning is best found in classification problems
2. **Unsupervised learning** – the algorithm is responsible for modelling given inputs and is not given labels or output values for it to learn from. It is usually used to best describe some hidden structures within unstructured data. Approaches that are used in unsupervised learning involve clustering and neural networks.

3. **Semi-supervised learning** – this brings together techniques of supervised and unsupervised learning combining labelled and unlabelled data to generate an appropriate model to be used. It is mostly used when a smaller portion of the available observations has corresponding labels.
4. **Reinforcement learning** – learns a policy that it can use to address the environment it is in together with the actions it can perform. Each decision and action that the algorithm makes will also cause some action of effect by the environment, this can be in the form of feedback that guides the learning algorithm.

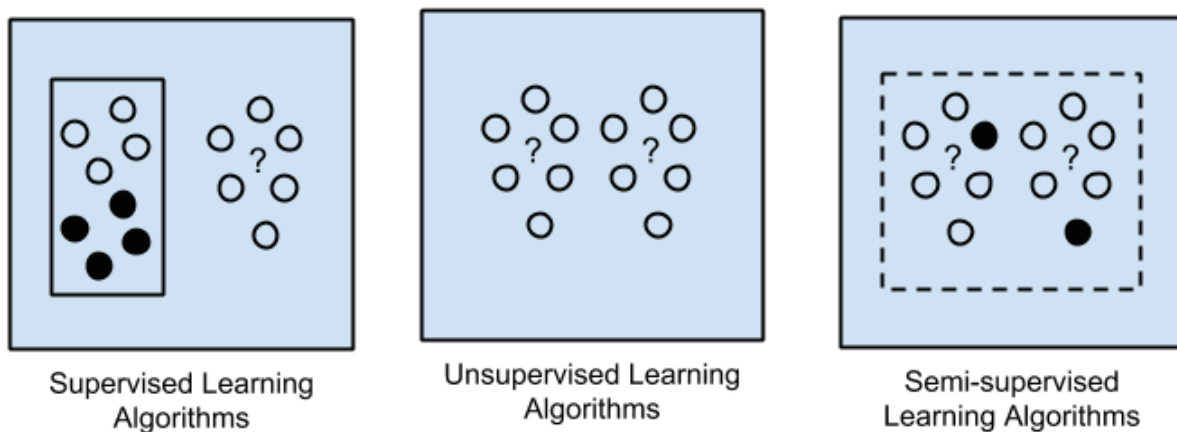


Figure 2-7 The training sets associated with each learning algorithm (Ayodele 2010)

Figure 2-7 shows how some of the algorithms work. Supervised learning having labelled data, unsupervised learning having no labelled data and semi-supervised having some data labelled. Machine learning is implemented on problems that are complex to program. These can be daily human tasks that can be easily programmed well to be a set of steps. Learning from experience makes this easier to achieve results using computers. Machine learning can also be used for complex problems that cannot be done by a regular human being. Data mining is one such example of complex problems. A human might find it hard or impossible to analyse a large data set while a machine learning algorithm combined with the high computation speeds of a computer can infer new knowledge. Adaptability is also a key feature of machine learning, the system is capable of changing itself to best suit its environment (N. Nilsson 2005). Programmed tools, on the other hand, are rigid and cannot adapt to their environment. This is good in a constant environment but the state

of the user and their environment can change at any given time. All these changes make it difficult for such a rigid tool to work at its best in a constantly changing environment.

2.3.1 Supervised Learning

In supervised learning, the algorithm has both the inputs and the expected outputs. The algorithm has to learn from this data and can then create a model or function that best fits the given data. Supervised learning can also be defined as the inference of some function that maps the given inputs to the expected results (Noh, 2015). The basic idea is that the model is given a training and a testing set. The inputs with labels or expected outputs are the training set. The model is then given a testing set which it then tries to identify using its knowledge from the training set. The identification that it does on the training set should be done with the highest level of accuracy (Erik 2014).

Typically, a learning algorithm has access to the following, the domain set, the Label set and finally Training data.

- Domain set – this refers to the set known as X which has the observations that are to be labelled. This could be, for example the different types of cars that may be on offer at a sale. The domain set is represented in most cases by a vector containing all the features of the set. Each feature may also be known as an instance of X .
- Label set – this is the set that has the labels for our data. This set can be referenced by the letter Y . Depending on the complexity of the problem the elements may be more in this set, for example, a two-element set would be something like $\{0,1\}$. 0 may represent a cheap car while 1 may be used to represent an expensive car.
- Training data – this is a set composed of a sequence of the pairs of x -values and the y -values. This basically means it contains a set of labelled values of the Domain set. The notation for this set can be generalized to be $S = ((x_1, y_1) . . . (x_m, y_m))$ meaning from the first training example set (x_1, y_1) to the last one represented by

(x_m, y_m) . A training set may be composed of a type of car and its corresponding speed rating for example (Mercedes A45, fast).....(VW Gti, slow).

The last part of the training algorithm is the hyperplane. The hyperplane is a function that maps X to Y and is also used to predict Y for new X-values in the testing set. The notation used for the hyperplane is $A(S)$, this means the hyperplane of the algorithm A will fit the data after being trained on S which is the training set (Ben-david & Shalev-Shwartz 2014).

Supervised Learning Workflow

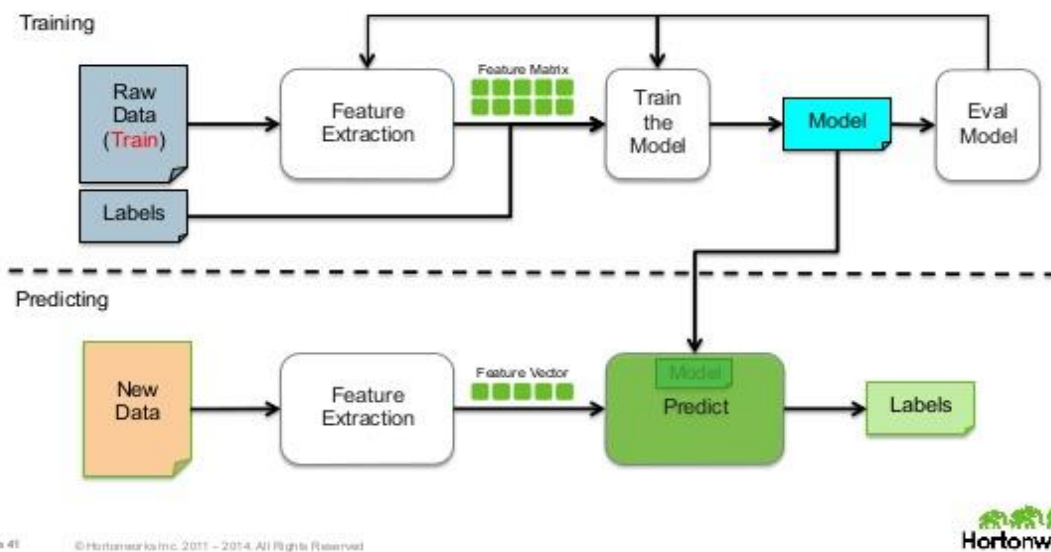


Figure 0-8 Workflow of a Supervised Learning algorithm (Paco 2014)

Figure 2-8 shows the workflow of a supervised algorithm. The first steps show the data acquisition and pre-processing. This involves transforming the data and addressing factors such as missing data. When the features and their labels have been correctly extracted then the feature vector can be used to train the model. The training of the model allows the hyperplane to be decided and fit to the data. The models are validated and verified on certain parameters and scored. If the performance measures are not to the satisfaction of the designer, both the data and the model re-evaluated. The data may contain abnormalities that hinder the model from fitting a proper hypoDissertation to the

data. If the model is satisfactory then the model is used to predict labels from new real world data.

2.3.2 Classification

Classification algorithms are a type of supervised learning algorithms. They are made of a group of algorithms that are able to perform classification on given data. Classification can be defined as assigning different objects into categories that have already been defined (Tan et al. 2006). A typical classification problem example is if a system is given appropriate training examples can the system then determine a set of unlabelled data points (Weis et al. 2009). Classification is made up of two phases:

- A training phase
- A testing phase

The training phase uses the training data points and the labels to allow the model to learn. The training data points are a collection of features that describe a particular entity. An example maybe a set of plants, some may be of different colour, different length, different leaf type, different uses and may bloom in different seasons. These are the features of the plants and each plant is described by these features. The labels are the instances of the plant such as a flower is a daisy, violet or rose.

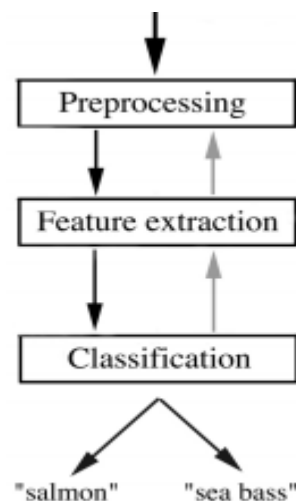


Figure 0-9 Classification process of Salmon and Sea bass (Duda et al. 2001)

Another example is that of fish such as salmon and bass. These can be used to classify a school of fish into these two categories. The different features such as length, scales, and more features can be used to distinguish them using classification (Duda et al. 2001). Figure 2-9 shows the classification stages of the bass and salmon problem and how the classification process would work.

Dimensionality reduction is a method that is used to remove noisy data within the training data. Dimensionality reduction techniques can be categorized mainly into feature extraction and feature selection (Tang et al. 2014). This involves the removal of the features that may be deemed redundant or irrelevant. This can include features that when the classification is performed they have little to no influence on the results (Kotsiantis et al. 2006). Feature extraction involves the transformation of features into features with lower dimensionality and also new features can also be created that are an aggregate of features that were originally in the training. There are a number of techniques available for features extraction such as:

- Canonical Correlation Analysis.
- Linear Discriminant Analysis.
- Principal Component Analysis.

Feature selection aims to make the training data more relevant to the labels used for training. It does this by only selecting the set of features that reduce redundancy and increase the relevance of the features to the label (Tang et al. 2014). Techniques for feature selection are as follows:

- Information Gain.
- Lasso.
- Relief.
- Fisher Score.

Feature selection is an important phase as that is when the features that will be used for the classification are chosen (Weis et al. 2009).

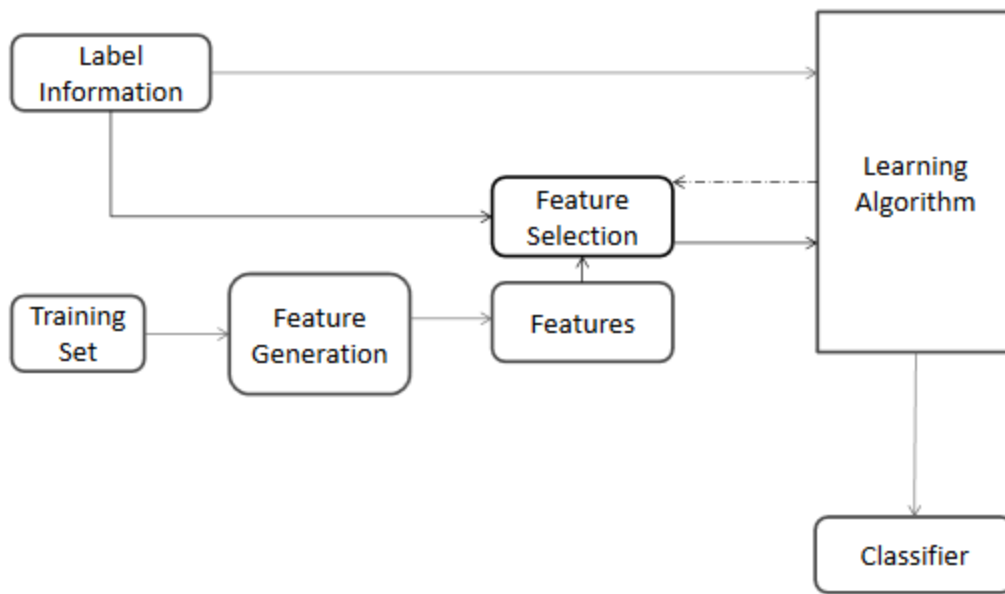


Figure 0-10 Process of feature selection before classification (Tang et al. 2014)

There are various considerations to be made during feature selection these may include:

- Commensurate data.
- Knowledge of the domain.
- Time.
- Pruning of data.

Commensurate data is data that is of the same size. The data will require normalization before classification can be done. Knowledge of the domain can also help in feature selection. This will be useful in figuring out which features are critical for the classification. Noisy data can affect the accuracy of the classification, for example, labels may not be clear for some instances. Time depends on the type of classification such as if the classification is part of a project and time is limited. The data may require pruning to maybe better fit the classifier or to satisfy computational limits (Singh et al. 2011).

When the data is ready for the training of the classifier it is used by the model to learn the features and their labels. The testing phase is used to give labels to unlabelled data. The classifier is given a set of features and must return the labels.

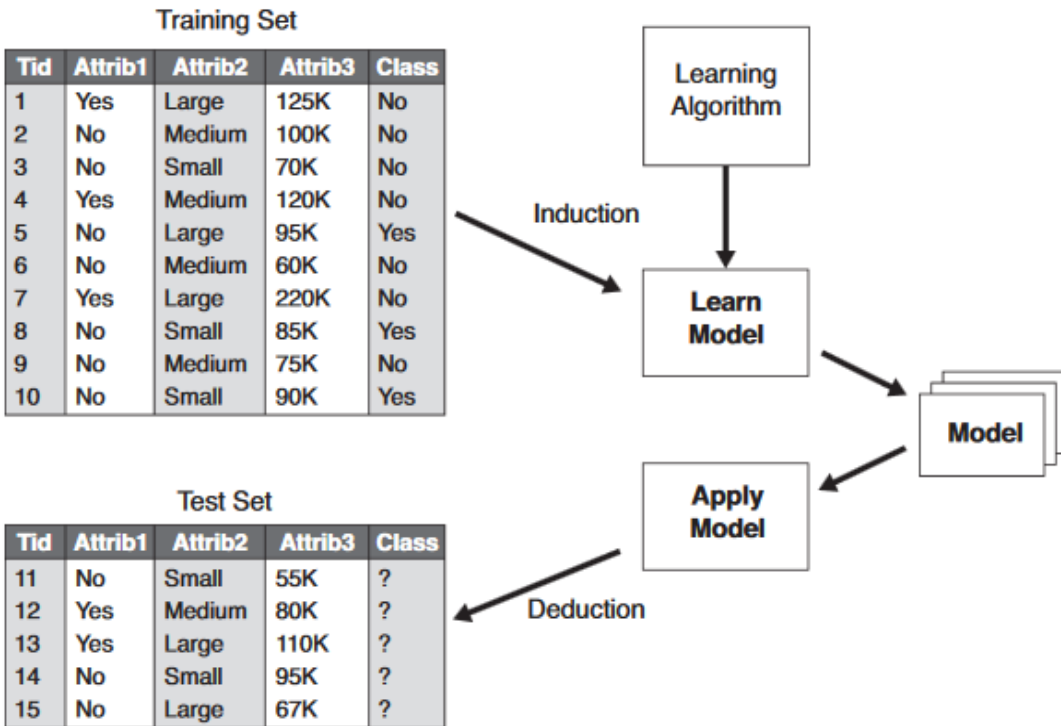


Figure 0-11 Learning and testing of algorithm (Tan et al. 2006)

Figure 2-11 shows the process of introducing the training data points with labels to the learning algorithm hence creating the model that best fits the data. The model is then applied to the test set. The labels for the test set are kept separate from the test set and can later be used to compare the predicted values to the actual values. This comparison is used as one of the metrics for classification algorithms.

2.3.2.1 Decision Trees

Decision trees are a type of classifier that classifies the given data using the features of the data. The trees are made of nodes that may have branches, each branch will represent a value that the given node can have (Kotsiantis et al. 2006). A decision tree can also be described as a data structure that is shaped like a tree hence the name given. Each of the nodes describes a decision that the classifier must make based on a particular feature (Weis et al. 2009). During feature selection, the most relevant features are chosen and one such method for selecting features is dimensionality reduction. This allows for the most relevant and the least relevant features to be chosen although this is not the

only method. The node that is at the beginning of the tree is known as the root node. The most relevant feature is used for the first split of the node into its branches (Kotsiantis et al. 2006).

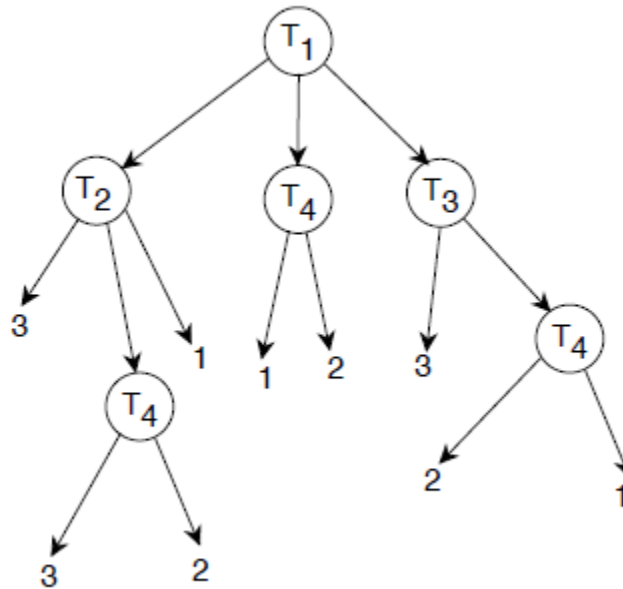
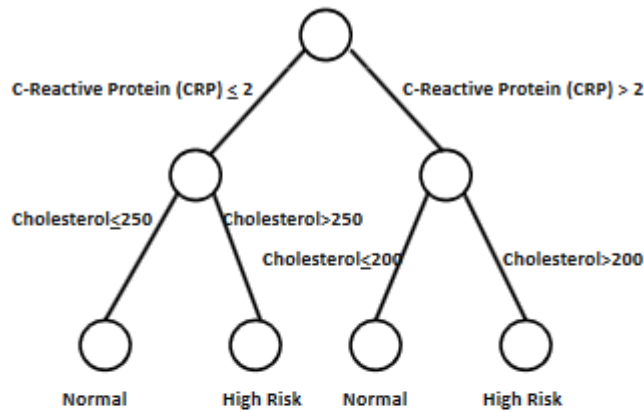


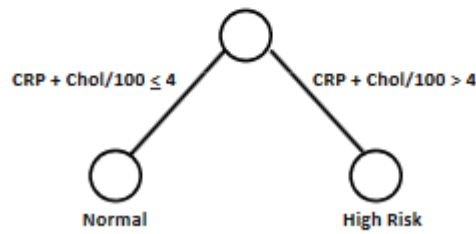
Figure 0-12 Simple decision tree structure (N. Nilsson 2005)

Figure 2-12 shows a simple decision tree with test shown by T. An instance will start at T1 and depending on the output of the test it will either go to T2, T3 or T4. At T2, for example, it will be tested again on a different feature and will either be classified as belonging to class 1 or class 3. In the case that further classification is required it might be sent down to T4. Each instance in the test set will go through this process and will be given a class label in this manner. This process of going from test to test is recursive as the instance goes from each subset to the next (Weis et al. 2009). The tests can be univariate or multivariate, univariate means the tests are on one feature while multivariate means the tests are on several different features (N. Nilsson 2005). As seen in figure 2-13 the univariate tests are only using one conditional statement while multivariate uses the two conditional statements. In some decision trees, there are only two branches for each node and these are known as binary trees. One major problem with implementing

an optimal binary tree is that it is an NP-complete problem. NP-complete problems are generally regarded as hard problems.



(a) Univariate Splits



(b) Multivariate Splits

Figure 2-13 Univariate and Multivariate splits (Weis et al. 2009)

The abbreviation NP means non-deterministic polynomial-time (Chuang et al. 2009). The NP problem in decision trees has led many researchers in trying to design heuristics that attempt and optimize decision trees (Kotsiantis et al. 2006). Listing 1 gives the pseudo-code for designing a decision tree. As seen in the code the function DT() is a recursive function as it calls itself again in the case that the node is not a leaf and hence builds a subtree of that node.

```

DT(Instances,Target_feature,Features)
If all instances at the current node belong to the same category
  then create a leaf node of the corresponding class
else
  {
  Find the features A that maximizes the goodness measure
  Make A the decision feature for the current node
  for each possible value v of A
    {
    add a new branch below node testing for A = v
    Instances_v := subset of Instances with A = v
    if Instances_v is empty
      then add a leaf with label the most common value of
        Target_feature in Instances;
    else
      {
      below the new branch add subtree
      DT(Instances_v,Target_feature,Features - {A})
      }
    }
  }
}

```

Listing 1 Pseudo code for Decision Trees (Kotsiantis et al. 2006)

After a decision tree has been constructed and trained it can then be evaluated. Decision trees can be evaluated using their generalization error. This is the probability that given a new instance to classify the tree will misclassify it (Almuallim et al. 2002). This allows the tree to be evaluated based on its ability to accurately classify an instance rather than on its ability to learn. The evaluation can be done by validating using a training set and a testing set. This validation may lead to different error estimates when different partitions of the data are used for validation. It is mostly used when a limited amount of data is available. This type of validation gives a validation estimate while splitting the data several times and using these as tests then finding an average gives a cross-validation estimate (Arlot & Celisse 2010). The most common algorithm used in decision trees is the C4.5 (Quinlan 1993). A study on the many different algorithms has shown that the C4.5 algorithm performs well given its speed and its lower error rate. Decision trees are usually simple and easy to visualize (Kotsiantis et al. 2006) (Weis et al. 2009)(Lim et al. 2000).

2.3.2.2 Support Vector Machine

Support Vector machines (SVM) use what are known as hyper-planes. The SVM attempts to find the hyper-plane that can separate the data perfectly into the two classes (Joachims 2002). From this definition, SVM is clearly a binary classifier. If the classes can be separated by the hyper-plane, then the function can be determined (Weis et al. 2009). SVMs uses a margin which exists on both sides of the hyper-plane which is used to separate the two classes. SVMs attempt to increase the size of the margins between the hyper-plane and the classes and this gives the optimal hyper-plane (Kotsiantis et al. 2006).

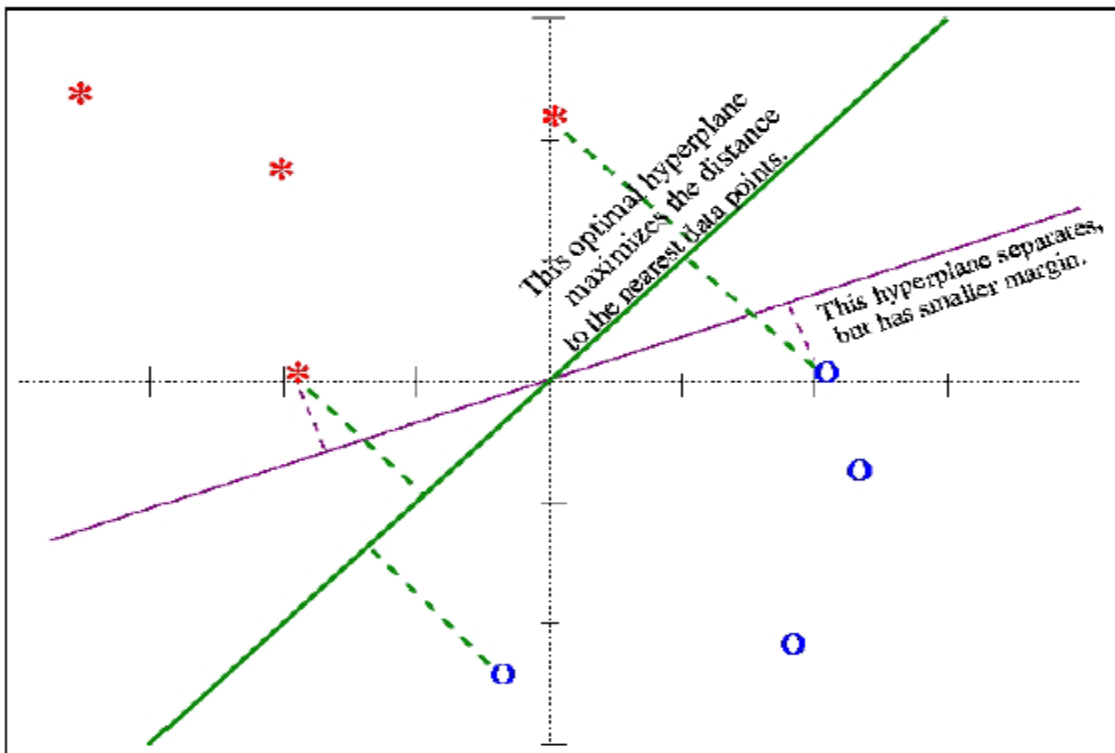


Figure 0-14 Hyper-plane fitting (Joachims 2002)

Figure 2-14 shows the fitting of a hyper-plane to a dataset. The two classes are shown by the blue and red markers. The line shaded in purple can be set as a hyper-plane as by definition it separates the two classes. The main problem is that it does not give the largest distance on both sides. The green line shows the optimal hyper-plane which has

the largest distance to the nearest points on both sides (Joachims 2002). Once this optimal hyper-plane has been found the support vector points are chosen. Support vector points are the points that are found lying on the margin of the hyper-plane. These points are the only points considered by SVMs and all other points are not (Kotsiantis et al. 2006). The probability of the SVM misclassifying an instance is bound by the ratio of the expected value of the number of support vectors to the number of instances in the training set. This is shown in the equation below:

$$E[Pr(error)] \leq \frac{E[\text{number of support vectors}]}{\text{number of training vectors}}$$

The error bound then allows for the optimal hyper-plane to be constructed from fewer support vectors in comparison to the number of instances found in the training set (Cortes & Vapnik 1995). This is one of the major advantages associated with SVMs, they work well in a situation where there are many features in the training instances showing high generalization ability (Kotsiantis et al. 2006).

In some cases, the SVM may not be able to find an optimal separating hyper-plane or even find a hyper-plane at all in the dataset. This may be caused by misclassified data points. The constraints associated with constructing the hyper-planes may be relaxed a little to allow for these misclassifications. This is known as using a soft margin (Fletcher 2009). Figure 2-15 shows two classes that cannot be linearly separated with an optimal hyper-plane. Using soft margin the data points that may be misclassified are penalized depending on the distance from the margin boundary.

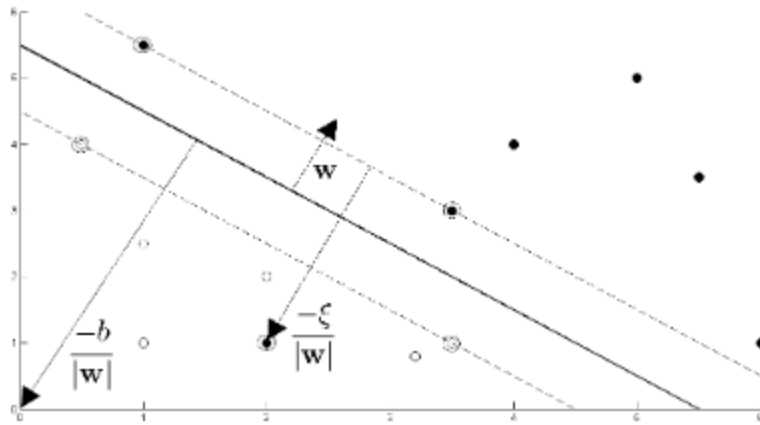


Figure 2-15 Non-linearly separable data (Fletcher 2009)

One other solution for the problem of non-separable classes of data is the mapping of the data into a feature space of a higher dimension.

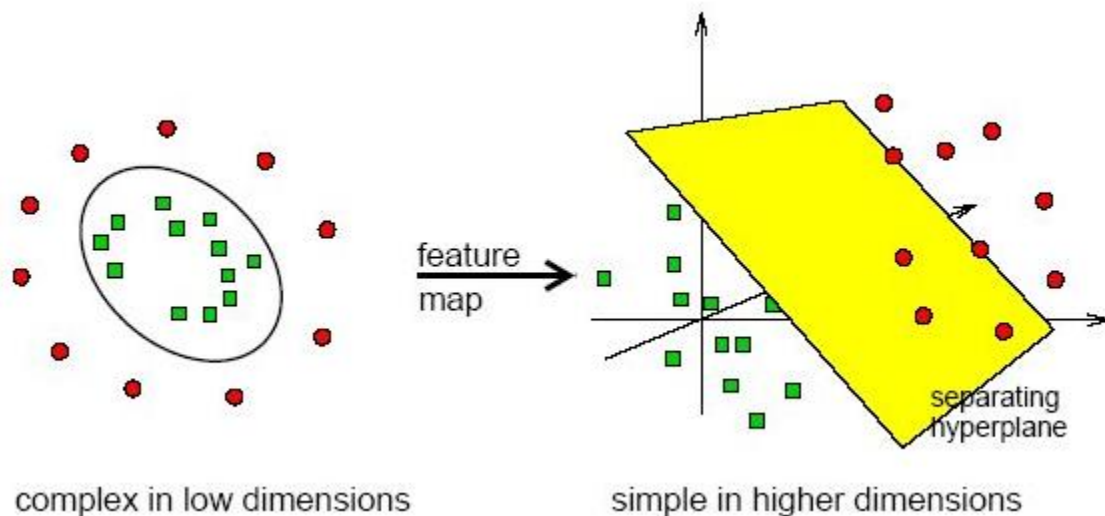


Figure 2-16 Difference of classification in low dimension and high dimension (Paco 2014)

Depending on the feature space chosen, the data can become separable (Kotsiantis et al. 2006). Figure 2-16 shows how the data can be easily separated when it is in a new feature space of a different dimension. A hyper-plane for this data can then be constructed and this allows for classification to take place.

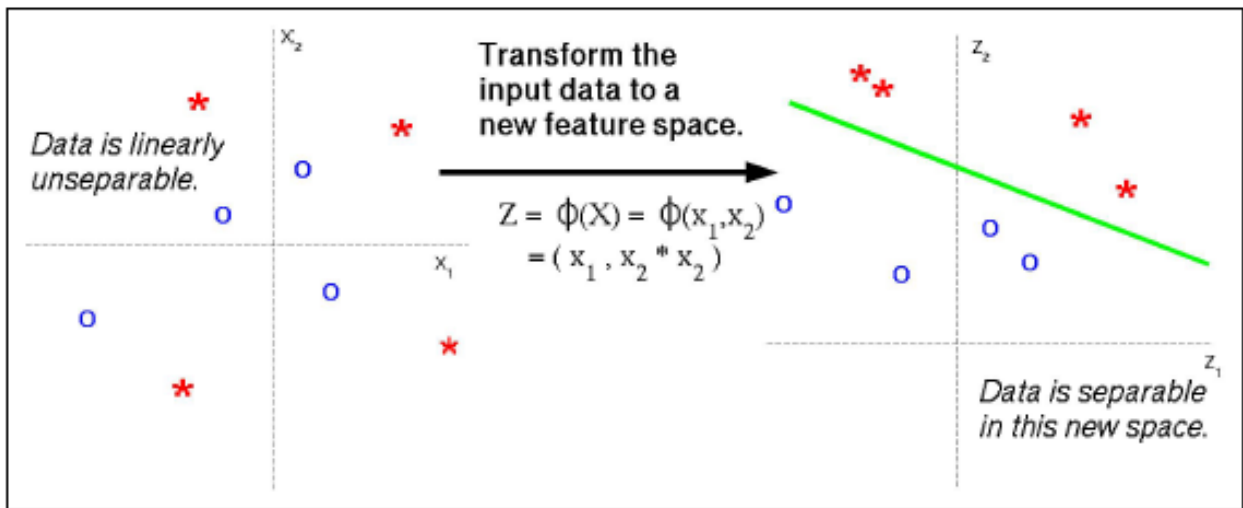


Figure 0-17 Transformation of input data (Joachims 2002)

Figure 2-17 shows the transformation of the input data to a feature space. The vector X is transformed by the function $Z = \phi(x)$ to a higher dimension. The function $\phi()$ is to be chosen such that the new training data in the feature space is able to be separated using a hyper-plane (Joachims 2002).

Kernel functions can be used to directly calculate the $\phi(x_i)$ within the feature space. The mapping of the data shown in figure 2-18 to the feature space will then not be necessary. The kernel function can then be used to map the new points into the feature space for classification once the hyper-plane has been constructed (Kotsiantis et al. 2006). Only the hyper-plane will be in the feature space while in the input space the data is separated by a curved contour.

The SVM uses the polynomial and the Radial Basis Function (RBF) kernels for non-linearly separable data (Joachims 2002). The polynomial function uses a non-linear separation boundary as shown in figure 2-18. While the RBF uses a separate space to separate the data classes as shown in figure 2-19.

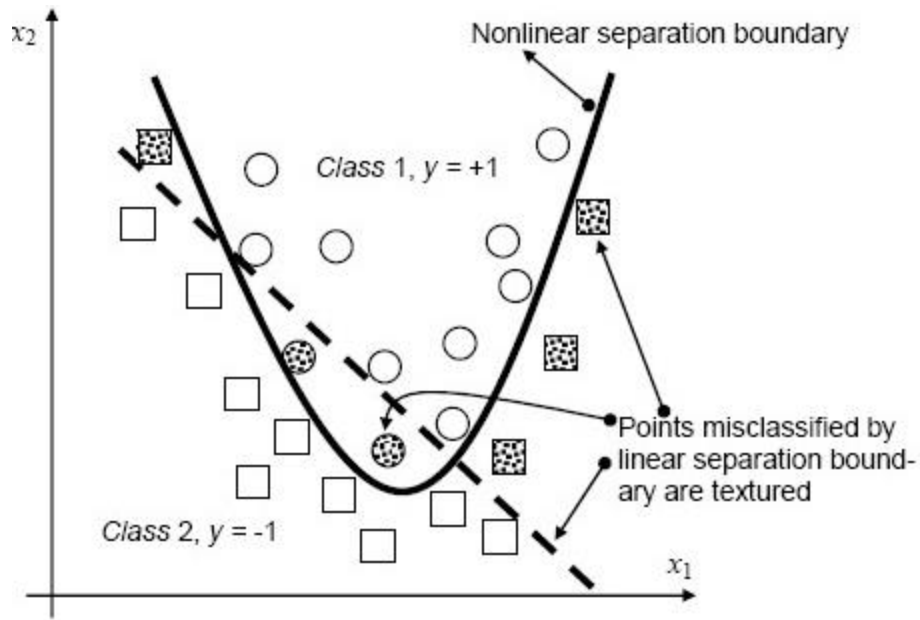


Figure 0-18 Polynomial function (Paco 2014)

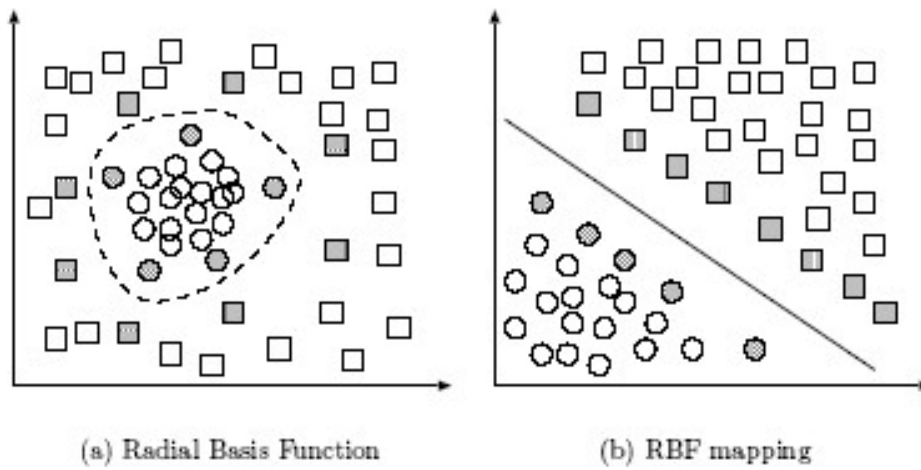


Figure 0-19 RBF kernel separation (Paco 2014)

2.3.2.3 KnearestNeighbors

The algorithms discussed above are known as “eager” learning algorithms and require more computation time to perform classification. K-nearest neighbors (kNN) algorithm is known as a non-parametric “lazy” learning algorithm meaning it requires less computation time (Kotsiantis et al. 2006). Non-parametric means that the kNN algorithm does not make any assumptions on the distribution of the data that it is given. kNN is a classification algorithm that is part of the instance-based algorithms (Kotsiantis et al. 2006). In this algorithm, the k nearest data points to the test instance in the training set are first selected and their class labels are collected. The most common of these class labels is the one that is then given to that test instance.

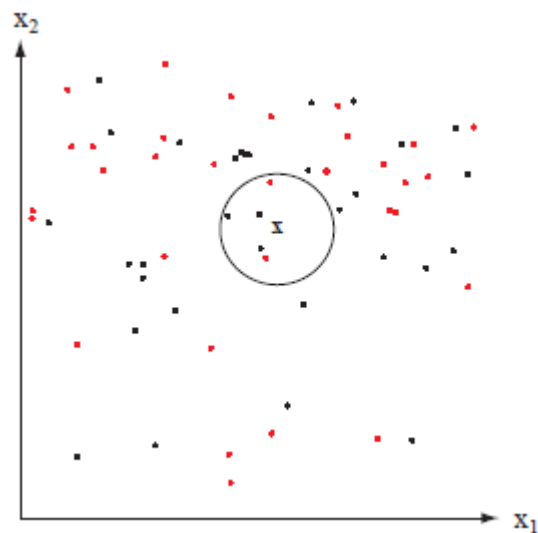


Figure 0-20 Shows the training instance x within a circle showing $k = 5$ (Duda et al. 2001).

The k number of neighbors used for the algorithm should be odd, this allows for a majority vote to be possible (Weis et al. 2009). This process is used for the k -nearest-neighbour rule, where k neighbors are chosen. There are also some situations where 1NN or the nearest-neighbour rule is used. This is where the closest point to the test instance is found and the instance is assigned this value (Duda et al. 2001).

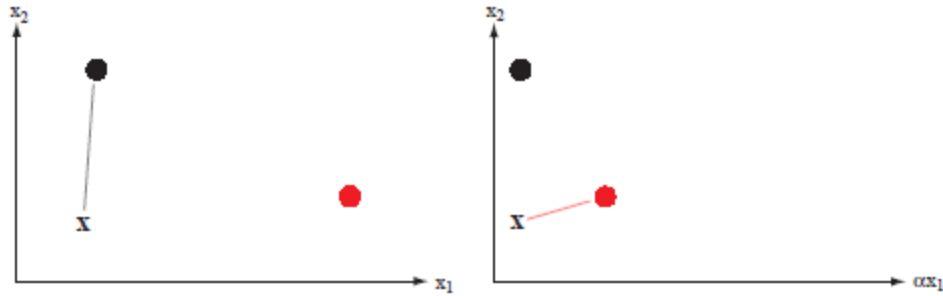


Figure 0-21 Shows the 1NN algorithm where the closest training instance is chosen (Duda et al. 2001).

The performance of this algorithm is highly dependent on k and the distance used. A larger k may improve the performance of the algorithm but also increases the cost. The cost of storage is a major factor in k NN. The k NN algorithm stores all the training examples in memory and uses a large part of the training data during testing (Kotsiantis et al. 2006). The k NN algorithm usually uses the Euclidean distance but other distances may be used.

2.4 Swarm Intelligence

Swarm intelligence can be regarded as having its origin from nature. One major inspiration that has led to the advancement of artificial intelligence in computer science is the way in which natural organisms behave in their respective groups. Considering any group of organisms such as ants, bees or a flock of birds, biologists have shown that groups of animals can behave in a particular manner that the individual members cannot attain on their own.

This has led to the design of algorithms in Computer Science that have been implemented in some of the systems in use currently (Eberhart et al. 2001a). Swarm intelligence can be described as the collective behaviour emerged from social insects working under very few rules (Abraham et al. 2006). Swarm intelligence to be fully defined has to follow the following rules:

1. Useful behaviour must emerge from the cooperative efforts of the group of individual agents.
2. The individual agents are largely homogeneous.

3. The individual agents act asynchronously.
4. There should be minimum to no centralized control.
5. Communication between the individual agents in the group is largely affected by some form of stigmergy.
6. The final behaviour should simplify the original problem faced by the group.

Swarm intelligence is about individuals cooperating in order to achieve a definite goal. Such as, ants finding the shortest path between their nest and a food or in another scenario bees finding the best sources of nectar taking into consideration the distance between the hive and the sources. These two processes have led to the major algorithms that have proved to be very substantial contributions to the sciences of computational optimization and machine learning. For example, with ants, the initial behaviours of a colony of ants when they require a new food source is for the individual worker ants to wander randomly in their environment. If a particular ant in the colony stumbles on an appropriate food source it will return to the nest all the while leaving a pheromone trail. The ants that follow will decide their path based on the pheromone trails left by the other ants before them. Over time the pheromone trails evaporate and only strong ones will remain. These are the paths to close by food sources or to a path most travelled by other colony ants.

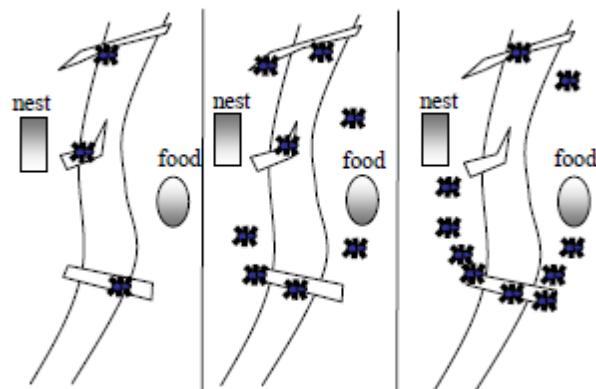


Figure 0-22 Ants crossing the twig example (Eberhart et al. 2001b)

In figure 2-22, ants need to cross a narrow stream of flowing water to get to the food source and there are three places for them to use to cross these are the fallen twigs. Initially, each ant is likely to try each twig without anything to change its decision. Each of these ants lays its trail of pheromone as it makes its journey towards the right and after a while, the path in the middle twig becomes less laid with pheromone. This is because it is the path less traveled by other ants and so the pheromones evaporate leaving a weak trail or none at all. Eventually, several ants are following the path using the lower twig attracted by the ever-increasing strength of the trail left by the others.

Although the upper twig provides a fairly short path this twig is narrow, and several ants cannot manage to make the trip because they fall off before being able to strengthen the path. This type of swarm behaviour is known as natural navigation (Eberhart et al. 2001).

There are broadly two types of natural process that are associated with the term "foraging", and in turn provide sources of inspiration for optimization methods. The idea is based on the emergent behaviour whereby the swarm locates the good sources and utilizes them methodically. The swarm exploits the food sources and then moves to others as the current ones become exhausted. The energy of the swarm is also used in an efficient manner unlike a brute force search of the entire environment of the swarm. The first type of foraging is found in bacteria. The bacteria move in their environment in a "run" or "tumble" manner. The run is a straight line movement in the direction they are facing and is usually for shorter distances. The tumble, on the other hand, is a movement in a random direction leaving the bacteria in new locations in its environment is usually longer than the runs.

In the case that the bacteria find a high gradient of food, the bacteria will start to make longer runs, and tumble shorter distances. It will keep moving upwards along the nutrient gradient while maintaining an element of stochastic exploration. The bacteria can sometimes leave some chemicals that other bacteria can follow. The other style of flocking is best represented by bees. When bees go scouting for nectar sources one may stumble on a source and will return to the hive. When it is at the hive, the bee performs a dance for the other bees in the hive. The better the quality of the food source the longer the dance. The bees in the hive will decide to go or not, they will taste the nectar to judge

if it is a quality source, the bees which decide to go to the source are said to have been recruited by the dancing bee. The direction and distance of the source are all communicated in the dance. The longer the dance the more the bees that will see it and can be recruited. When the source is being exploited each bee will come back and do a dance also but it will gradually get shorter as the source is also depleted.

Since the research in the field of swarm intelligence started in the late 1980s the field has had time to grow and find its applications in many other fields of Computer Science and other fields. Swarm intelligence has its applications in conventional optimization problems, sometimes used for library materials acquisition, dataset classification, dynamic control, communications, tracking and prediction (Eberhart et al. 2001).

To model the broad behaviours from a swarm, we introduce several general principles for swarm intelligence (Bonabeau et al. 1999):

Proximity principle - The basic units of a swarm should be capable of simple computation within space and time. Here computation is regarded as a direct behavioural response to variance in space, such as those triggered by interactions among agents. Depending on the complexity of agents involved, responses may vary greatly. However, some fundamental behaviours are shared, such as living-resource searching and nest building.

Quality principle - Apart from basic computation ability, a swarm should be able to respond to quality factors, such as food and safety.

The principle of diverse response - Resources should not be concentrated in the narrow region. The distribution should be designed so that each agent will be maximally protected facing environmental fluctuations.

The principle of stability and adaptability - Swarms are expected to adapt environmental fluctuations without rapidly changing modes since mode changing costs energy. This principle in some literature is separated into two different principles, the principle of stability and the principle of adaptability.

In computing swarm intelligence deals with some specific behaviours of organisms to which the algorithms found in swarm intelligence are based some which have been mentioned above. Ant Colony Optimization (ACO) algorithms and Particle Swarm Optimization (PSO) algorithms are the two major algorithms that have been developed from swarm emergent behaviour. ACO draws inspiration from the observation of ants and how they solve creating routes to food sources as shown in Figure 2-22. PSO is inspired by flocking and swarming behaviour of organism which has a certain goal or final state. Each particle in the search space has a goal to search for the optimum and since it's moving in the search space it also has a velocity associated with it (Bonabeau et al. 1999) (Abbas et al. 2015). The particle also maintains its previous position in memory as its personal best. The particles in the swarm co-operate by exchanging information about what they've discovered in the places they have visited. The co-operation is very simple and can be loosely explained as this:

1. A particle has a neighbourhood associated with it.
2. A particle knows the fitness's of those in its neighborhood and uses the position of the one with the best fitness.
3. This position is simply used to adjust the particle's velocity.

2.5 Pattern Analysis algorithms

What is pattern analysis? This field of study can be defined in many ways. It can be defined as a field concerned with machine recognition of meaning regularities in noisy or complex environments. It can also be defined as a classification of input data via features selection from a lot of noisy data (Dutt et al. 2012). Finally, it can be defined as a scientific discipline of machine learning (or artificial intelligence) that aims at classifying data (patterns) into a number of categories or classes. Based on these definitions, a pattern can be any entity of interest which one needs to recognize and/or identify (Kpalma & Ronsin 2007). Pattern analysis can be divided into two major tasks:

1. The analysis or description that extracts the characteristics from the pattern being studied.
2. The classification or recognition that enables us to recognize an object or pattern by using some characteristics derived from the first task.

There are two major techniques used in pattern analysis, these are Partition and Hierarchical techniques. Partitioned clustering attempts to break a dataset into K clusters such that the partition optimizes a given criterion. Hierarchical clustering algorithms produce a nested sequence of clusters, with a single all-inclusive cluster at the top and single point clusters at the bottom. Some algorithms used in pattern analysis are as follows:

2.5.1 K-Means

K-means is one of the oldest algorithms in use today. It is regarded to be part of clustering algorithms class. Clustering is a division of data into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification (Berkhin 2002). It is a prototype-based clustering technique defining the prototype in terms of a centroid which is considered to be the mean of a group of points and is applicable to objects in a continuous n-dimensional space. The figure below shows how the K-means algorithm functions:

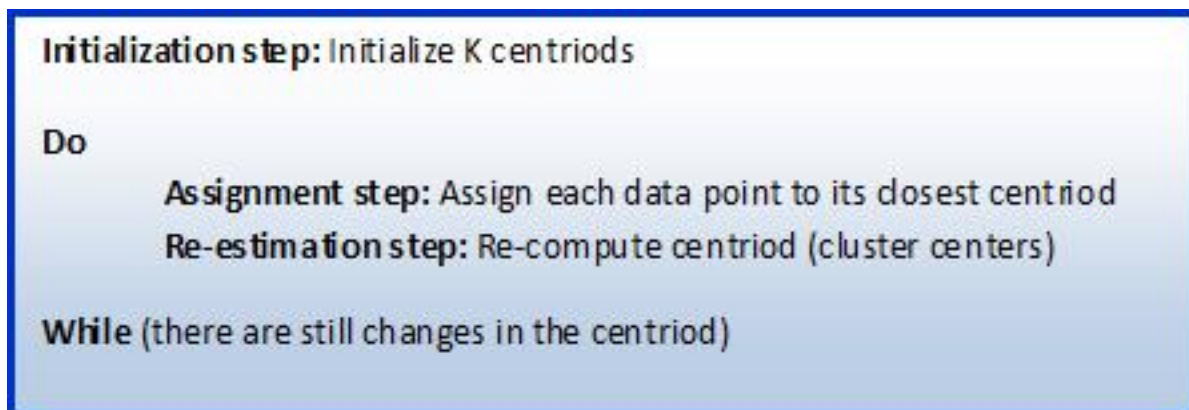


Figure 0-23 pseudo-code for K-Means (Duda et al. 2001)

K-means uses the squared Euclidean distance to allocate objects to clusters. Because the squared Euclidean distance is used, the K-means algorithm proceeds to try to find a minimum for the Error Sum of Squares. K-means is a nice method to quickly sort your data into clusters but local optima in K-means can derail final results.

2.5.4 CHAMELEON

This algorithm falls in the class of hierarchical clustering algorithms. Hierarchical clustering builds a cluster hierarchy or, in other words, a tree of clusters, also known as a dendrogram. Every cluster node contains child clusters; sibling clusters partition the points covered by their common parent. Such an approach allows exploring data on different levels of granularity. Hierarchical clustering methods are categorized into agglomerative (bottom-up) and divisive (top-down) (Berkhin 2002). CHAMELEON operates on a sparse graph in which nodes represent data items, and weighted edges represent similarities among the data items. This sparse graph representation of the data set allows CHAMELEON to scale to large data sets and to operate successfully on data sets that are available only in similarity space and not in metric spaces. CHAMELEON finds the clusters in the data set by using a two-phase algorithm. During the first phase, CHAMELEON uses a graph partitioning algorithm to cluster the data items into a large number of relatively small sub-clusters. During the second phase, it uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly combining together these sub-clusters (Karypis & Han 1999)

2.6 RELATED WORKS

2.6.1 Mobile crowd-sensing in the Smart City

A smart city is defined as a city that connects to its citizens in novel ways taking advantage of the shift to IoT technologies. A smart city citizen has multiple technological solutions to their disposal. Smart cities can manage the city's resources in a more efficient manner,

such resources are schools, transportation systems, hospitals, power plants, law enforcement, and other services provided by that particular smart city (Musa, 2016). According to Sam Musa, “The goal of building a smart city is to improve the quality of life by using technology to improve the efficiency of services and meet residents’ needs”.

The city interacts with users by collecting real-time data from the citizens by using sensors and in some cases actuators. With all this the city can be made safer and lives of the citizens also made easier by improving services (Musa, 2016). In mobile crowd-sensing the citizens of the Smart City collect, share and jointly use services based on the sensed data, e.g. the Waze application for optimized car-based navigation, the Smart Citizen project for collecting meteorological measurements.

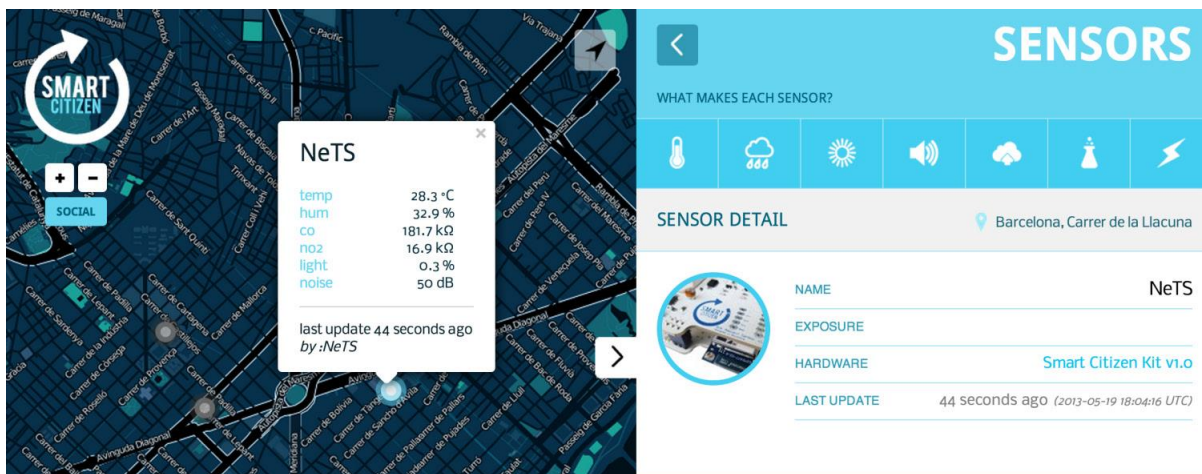


Figure 2-24: The Smart Citizen website (Lendák, 2016)

The developers of this platform discovered that there were limitations associated with the mobile devices in use. The limitation was that the mobile devices have a limited arsenal of sensors for collecting information about the natural environment. This was mitigated by instead building their own sensing hardware. The Smart Citizen project consists of

- The Smart Citizen Kit
- A mobile application
- The Crowd

The smart citizen kit can be placed anywhere high and where it can charge using solar power. Username and geographic location can be grabbed from the screen, which negatively impact smart citizen privacy. This can although be solved by only removing the username to keep user information anonymous (Lendák, 2016).

Using the mobile applications and the crowd the system allows for many applications. Citizens within the smart city can communicate and discuss information using other applications such as traffic information using Waze.

This system managed to solve the limitations on the sensors readily available on a mobile device. This allowed them to use the surrounding natural environment of the user for the system. This gives a much larger view of context being collected. They also solved a power issue making it very affordable to run the smart citizen kit using solar energy.

2.6.2 Waze

Waze is arguably the world's largest crowd-sensing-based traffic application. It is by definition a navigation application allowing users to move from one point to the next. The reason for its growth is that it allows the users of the application to share traffic events with other users on the application. These updates vary from road works, accident or detours. The system then aggregates such contextual information and shows this on the local machine. Waze is not strictly a smart city only application but can be used in any community (Lendák, 2016).

Waze relies highly on the activity of its clients for it to be efficient in communicating information and for the updates. This motivated Waze to create a way of giving users motivation to use waze. They use a levelling-up system that makes use of different ranks and titles for users from baby to king. The number of pints awarded to a user are based on the active use and on the issued reports. Waze uses the sensors on a device as follows:

- GPS for the user's location
- Camera in the case that a user takes a picture of the event

- Accelerometer to detect certain automated events



Figure 2-25: Screenshot of waze application (<https://www.pcmag.com/article2/0,2817,2493448,00.asp>)

As can be seen in figure 2-25 a user can see all the other users in that particular area. This is not limited to just friends but everyone and can prove quite dangerous to users of the application. Another limitation of the application is that individual can give false reports to the application. Such message fabrication attacks might be used to cause havoc in traffic systems (Jeske, 2013).

Waze does not fully utilize the device sensors and focuses on GPS and accelerometer. The use of all sensors to completely try and observe the environment lacks in this application and hence the information it can give is limited. Also anonymity is a feature that can be easily added to reduce the risk each user has.

2.6.3 Anagog

Anagog is a new application that has found its way among the context-based applications. Anagog makes use of the available sensors on the device using just about 8 of the available sensors on smartphones (<https://www.anagog.com/how-it-works/>). The company has developed and SDK that allows for the learning of user behaviour from the

sensors. One of the major features is parking assistance. The application learns when users park and leave parking allowing for anagog to notify another user of free parking.



Figure 2-26: Anagog parking application (Lendák, 2016)

This application uses the data from Waze to find and the share any parking events e.g. “left parking” or “entered parking”. This application then learns how the users park their cars, where and when (Lendák, 2016). The main advantage of this application and what sets it above the rest is that anagog has developed an SDK that they built the application on. The SDK is called JedAI, according to the anagog website the jedAI SDK is the first on-handset AI engine. It also reduces the battery consumption and maximises the privacy.

Chapter 3: METHODOLOGY

The main objective of this chapter is to discuss the steps taken by the researcher from the conceptualization of the research topic to the conclusion of the research. This chapter will discuss how the literature is reviewed, how the data is collected, how each component of the system is designed and how they is tested and against which metrics.

The choice of methodologies used is decided through literature review and applied to the research in a manner that closely fits the research. This chapter begins by giving the different methodologies reviewed in the research and how they were used to give the methodology of this research. The chapter discusses each of the methods of the research and concludes by giving a brief summary of all aspects discussed.

3.1 Definition

Research methodology refers to how the research addresses the problem statement. This section discusses the methods by which the research is designed and results analysed to answer the research questions and also achieve the objectives of the research. According to Polit and Hungler (2004), methodology refers to ways of obtaining, organizing and analysing data. Henning (2004:36) describes methodology as a coherent group of methods that complement one another and that match the data and findings to the research question and suit the research purpose.

The methodology is selected on whether it will give sound and plausible results that address the research. The methodology selected gives the researcher adequate principles and methods to perform the research in the best possible way. The methodology is referred to at each point of the research so as to maintain order and keep track of the research.

3.2 Overview of the Methodology

This study focuses on the aggregation of the context of a group given the changes of their activities based on sensor information on their devices.

The methodological research design of this research uses a multi-method quantitative approach. The main strategy used for the research is an experimental one where the results found are used for answering the research questions. This research makes use of the Waterfall model in conjunction with Iterative and Incremental development techniques. The use of combined models was found to be best suited for this research because using the Waterfall model only has many fallouts. The Waterfall model relies highly on a “One-Time” approach (Davidson 2004). Development and documentation are all carried out in that one project life-cycle using pre-set rules with the idea that it will be successful at the end. This may be the case in some situations but carries a high risk whereby if the development fails at a certain stage, the whole development has to be redone from the top. This wastes resources and time which are valuable for the project and risks the quality of the final product.

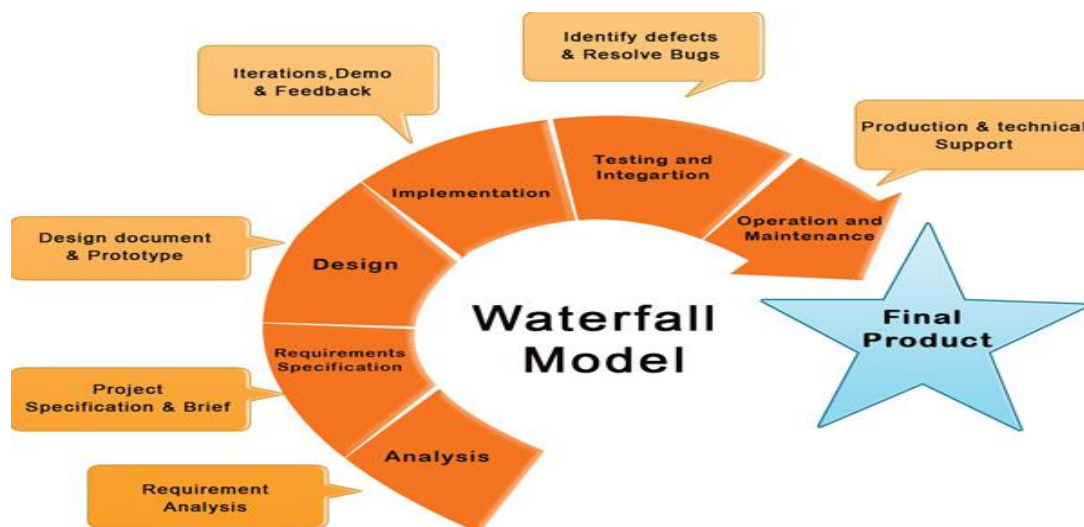


Figure 3-1 Waterfall Model stages (Davidson, 2004)

As shown in Figure 3-1 the waterfall model does not repeat any steps which becomes its major fallout. Repetition gives the opportunity to modify the final product presenting the best possible product with the available resources. Repetition allows for most of the practical requirements to be achieved. The research will follow a set of rigid steps but will also be highly iterative and flexible for system development. Iterations can also be based on how the system prototypes perform based on the metrics which will be discussed later in this chapter during analysis. The Waterfall model has its advantages such as keeping track of the research. The model has a linear set of phases to be followed that are sequential and rigid. This rigidity gives the researcher the opportunity to reference the stages of the research with the project plan, timelines and required deliverables. Iterative and incremental development breaks the project into sections that can be done sequentially. At any point during the project, a section can be revisited or repeated entirely. The different components of the System will be designed and implemented in this way.

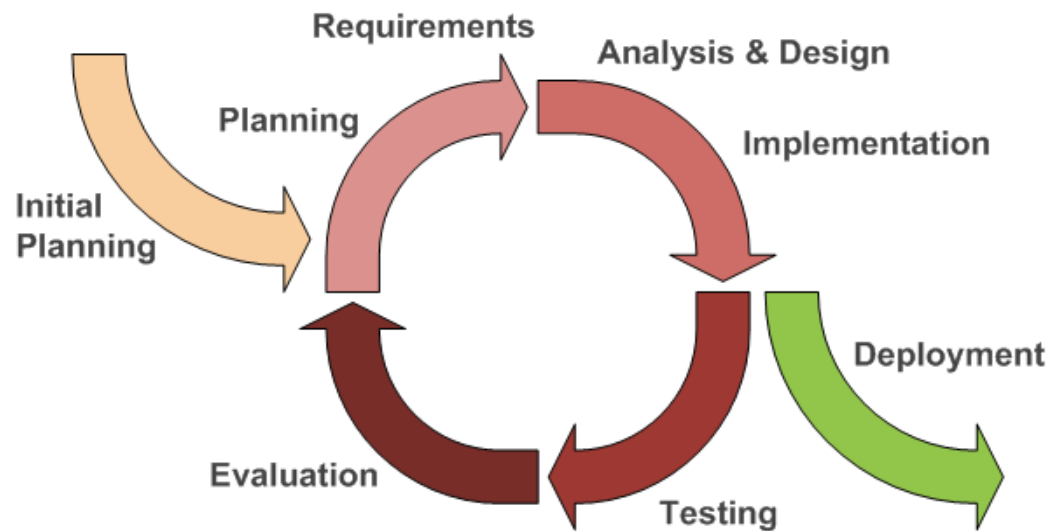


Figure 2-2 Incremental and Iterative Development (Davidson, 2004)

Figure 3-2 above shows the lifecycle of incremental and iterative development. The planning, implementation, testing and evaluation phases make the core of the lifecycle in an iterative manner before development. Figure 3-1 showed the rigid structure of the

Waterfall model, these two models were used to come up with the methodology stages of this research. The write up of the research is done as each of these stages is completed in a sequential manner.

3.2.1 Research Conceptualization

This research involves many different fields of study that were focus areas during the literature review. The related work is work that has been done by other researchers or developers that have explored the same domain as the work discussed in this research. The work previously done has been observed and points to build on observed. Most systems observed work well together but not as individual stand-alone systems. The systems show that the research can be done and value can be found within the research. Anagog attempts to use learning to find user behaviour but this is for personal use and cannot be used by organizations or have other applications. This step of exploring the problem gives form to the research and its aims.

3.2.2 System Requirements

The focus of the research is to build a system. The system requires information that will be used in the design process. This information is the documented in the system requirements. The requirements of the system in this research are found from the research questions and the objectives. These are expanded on to find the particular components of the system and define how each of these interact with each other. The process is iterative and incremental and so with each iteration if new requirements were found or a requirements feasibility is in doubt, new requirements could be added

3.2.3 System Design

This describes the system requirements, operating environment, the architecture of the system and its components, database schema, interfaces of the system and how the system interacts with its environment. The system that is developed for the fulfillment of the research is comprised of different components that function as a unit. The system

requirements of each of the components such as the Android application and the server need to be elicited and documented. The database schema depended on the format of the data collected during the data collection phase. The system design gives the different interfaces of the system and how the user will interact with them and this is done using use case diagrams.

The components of the system are as follows: A Sensor application for collecting context information, a server for aggregating the data and storing it in the database and learning algorithms for classifying and pattern analysis. The system design chapter documents how each of the components is to function and the high-level logic of its functionalities. This is shown using different design tools such as sequence diagrams and flow charts.

3.2.4 Data Collection

The data collection methods are dependent on the sources of the data, which data from these sources will be collected, how will the data be collected and within which intervals will the data be collected. Data was collected from the sensors of the user's device and from these sensors the data which will be collected will need to be decided. In the case of the accelerometer data the values are needed within an interval that does not miss slight changes in movement. The data for light and location is collected over a certain period of time. This time interval depends on how the sensor value may affect the results of the research. A method of collecting the data from the sensors and storing it in the server database was designed. A tool for collecting the data will also was designed in the form of the Android application. Data collection needs to be done at the appropriate intervals for the system to give accurate results. The decision on the intervals was done through iterations were a prototype for the system is implemented and tested using data collected at different intervals. The best-suited interval was selected in this way as it will give the better results from the other intervals.

3.2.5 System implementation and testing

At this stage, the actual coding of the components of the system is done. This is done in an iterative manner; each component is implemented and then tested against the requirements. The feasibility of some of the requirements is tested here as some may not be possible given the resources of the project. At such a point, we revisit the requirements and attempt to adjust them so they may be more suitable for the system. Different integrated development environments are used at this stage to develop the system components. Different programming languages need to be used for the implementation of each component. Different reasons led to the choices of programming languages to use. The availability, cost and how the language integrates with the system as a whole are some of the factors taken into consideration in the selection process.

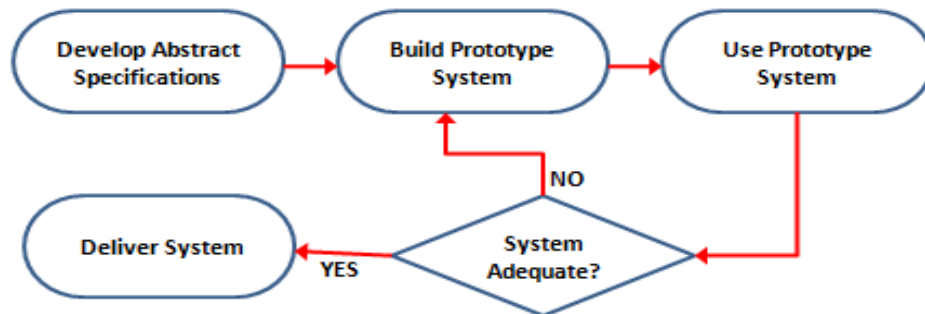


Figure 3-3 Evolutionary Prototyping Model

Figure 3-3 shows the stages of evolutionary prototyping. During the implementation, evolutionary prototyping is used to develop each of the system components. The prototype is developed in an incremental and iterative manner. With each increment, the designed component is tested thoroughly and if changes are required they are implemented and the cycle continues until the final system is developed. The final system goes through testing of its own before being deployed. Functional testing is used to verify the different functions each component is supposed to perform. The methods used to test that are requirement-based testing and then at a later stage business rules-based testing.

Requirement based testing tests how well the system components satisfy the system requirements. Business rules-based testing is concerned with the testing of the system in real-world scenarios and observing its behaviours.

3.2.6 Data Analysis and Results

At this stage, the system was deployed into a controlled environment and left to run in different scenarios. Different iterations are undertaken on two devices as this was what was available. The results of each iteration are stored as a new device each time. The system is in a controlled environment and so minimum transformations are required. The results are used for pattern analysis and this also returns its own set of results used as the base of the research. Other results include those from the accuracy information of the classification algorithm during its optimization stage. The results of the research are finally used to support the aims of the research.

3.3 Summary

The methodologies mentioned are used together to come up with the set of steps to be followed in the research. The chapter gives an overview of how the research and the system development is to be performed. The requirements gathered from the literature review and the research objectives are used to design the system used in the research. From the design, the system can then be implemented using different tools and technologies. The final product is tested against the requirements and the results are collected and analysed. Functional requirement-based tests are used to verify the functions of the system while business rules-based tests verify its functionality in various scenarios. We conclude by mentioning how the results are analysed and tested.

Chapter 4: DESIGN AND IMPLEMENTATION

The research design gives a detailed overview of how the research work will be carried out. The system design involves a number of activities that help in designing an architecture of the system. The architecture made in the design attempts to give the details of how the system and its components will function and interact. The interactions involve the software components, network operations that may be required, the user interface and the specific database and its tables that will be needed. A number of the design decisions about the system and its required functions were made in the conceptualization of the research. These are mainly discussed as some of the objectives of the research. The design philosophy and the design strategies are chosen in this chapter. The philosophy gives an abstract approach taken by the researcher, their beliefs, personal encounters and their understanding of the problem space determine how the research is performed. The strategies determine how the researcher will answer the research questions and achieve the objectives of the research.

Implementation refers to the development of the system using different programming tools. The implementation makes use of the available requirements and also the specified design of the system. This implementation chapter shows how the system was developed after completion of the design. The design was used to show the architecture of the system and the system requirements. Development of each of the components of the system discussed in the architecture will be documented in the chapter. This chapter constantly references on the design chapter to test the prototypes against the system requirements.

4.1 Research Design

This research is done from the point of view of the researcher and assumptions made in the research are at the discretion of the researcher. A research is aimed at finding a certain truth, and the three major paradigms for social sciences are Positivism, Interpretivism, and Critical theory. These three paradigms can be taken and applied accordingly as a research philosophy. The research philosophy taken by the researcher is important for how the research will be designed. The research philosophy used for this

research is “positivism. This means the researcher designs the research based on scientific methods. Positivism is a paradigm that closely relates to the natural sciences in that it is empirical, deterministic and generalized. Scientific methods usually involve testing the hypothesis of the research using a cause-and-effect approach. This philosophy distances the researcher’s personal values from the study allowing the research to be objective.

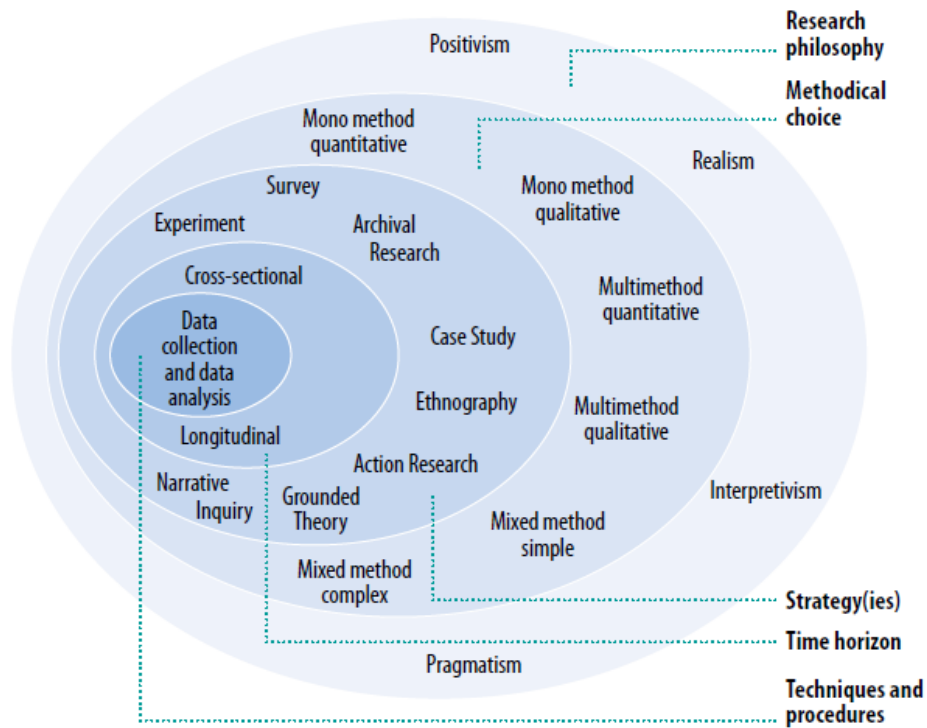


Figure 4-1 Research Onion (Saunders & Tosey 2013)

Figure 4-1 shows the research onion which was used to design the research methodology and decide on the approach to the research. The research philosophy has already been discussed, the next decision is the methods used. The methods used for the research are multi-method and quantitative. This is because the data collection methods are quantitative and so quantitative data allows for quantitative research. Quantitative data allows for objective decisions and scientific assumptions. From the methodical choice the next layer is the strategy. A research strategy is a tool for answering the research questions. The most appropriate strategy for this research is an experimental strategy. The research is experimental and gives results used in addressing the problem statement. With the research philosophy, methodical choice and strategies decided, the

research design must also be able to bring the link between the research questions and the objectives. The objectives are mapped to the research questions, this is known as the usability of the research design. Table 1 shows how each of the research questions will be answered and by the objectives and their deliverables.

Table 1: Mapping of Research Questions to Objectives

RESEARCH QUESTION	RESEARCH OBJECTIVE	EXPECTED RESULTS
One	One	
What are models currently available for context-aware computing?	Analyse context-aware computing models	An understanding of the models and a well explained a section of these in the Literature Review Chapter.
Two	Two	
What context can be found using sensors on a device and can an application using this context be developed?	Design a simple agent capable of acquiring information from a device's sensors.	An Android application implemented using JADE and can perform the functions mentioned in the objectives
Three	Three	
Can the context information from different devices be	Design a system that collects the contexts of a multi-agent system and	A multi-agent system that will collect and aggregate

collected and aggregated using a central service?	aggregates the context information.	the information sent by the agents.
Four	Four	
Can behaviours and patterns be mapped to certain context?	Analyse the patterns that result from the analysis of the contexts associated with the multi-agents.	A well-documented evaluation of the results of the analysis by the multi-agent system

Research question one involves the study of the different context-awareness models; this is addressed in the literature review chapter where different models are explained in detail. The relation between the models and the system developed in the research is also highlighted in this stage. Research question two involves an investigation of if sensor data can be collected and used as context information. This is addressed by the design of a mobile application capable of collecting and using sensor data as context. The design of the mobile application is discussed later on in this chapter. The third and fourth research questions are addressed by the development of a back-end system that is capable of aggregating and then analysing the context information of a group of individuals. The architecture of the back-end system is discussed later in this chapter under the system design.

4.2 System Design

This section gives the system requirements of the system and specifies the components functionalities. The methods for data collection are also specified in this section. It concludes by giving the system architecture and the system use cases.

4.2.1 System Requirements

The system requirements are statements that give how the different components of the system should function. Each requirement is taken into consideration before the system can be designed. The functional requirements of each of the components are stated below:

Mobile application

1. The application should be able to use the sensors of the device it is deployed on.
2. The system should be able to collect the values from the sensors.
3. The application must have a selection of all the activities that can be performed by a user.
4. The application must be able to send the context information to the aggregation system.

Back-end System

1. The system must be able to receive the context from the mobile application.
2. The system must be able to aggregate the context of different devices.
3. The system must be able to store the context of the devices.
4. The system must be able to classify the context to a particular activity.
5. The system must be able to find patterns associated with the context of multiple devices.

4.2.2 Interface Requirements

This interface is only for testing the system. The interface is only be available during the testing phase and beyond that, the application runs in the background. This then means the application does not need to interact with the user using an interface but only needs to be launched to start functioning. The training phase interface requirements are as follows:

1. The mobile application must show the sensors being used.
2. The mobile application must show the changes in the sensor values.
3. The mobile application must have a way of starting and stopping the sensors.
4. The mobile application must have a way of selecting the activity being performed.

4.2.3 Data Collection

The source of data for the system is sensor data that was collected from the user's mobile device for training purposes. This data is the data that was used in the research to train the selected learning algorithm. The learning algorithm then gave the results used later in the research in the form of predictions of activities being done. Two methods of data collection were proposed:

1. Collecting sensor data at different intervals for each sensor.
2. Collecting sensor data from all the selected sensors simultaneously.

The first proposed method will get the sensor data from each of the sensors at different pre-set intervals. Some sensors will not change over short periods of time while some will. For example, the accelerometer is affected by the acceleration due to gravity and so can change at any time while temperature may not change that frequently. Activities such as jumping take a fraction of a second to complete. The changes in sensor values at each point of the jump to increase accuracy for the predictions. This method also puts into consideration the issue of battery life of the device. Constant probing for sensor data may compromise the devices battery life.

The second proposed method disregards the differences in how often the sensors change values. This method collects all the data at the same time with a constant interval across all sensors. Again, these intervals are decided by the researcher. This means all sensors are probed for their relative values simultaneously. This allows for all sensor data to be available for each data point that will be used during the learning stage. These two

methods will both be tested based on storage in the database and compatibility with the learning algorithm. Figure 4-2 shows a model of how data collection was performed during the training phase of the system.

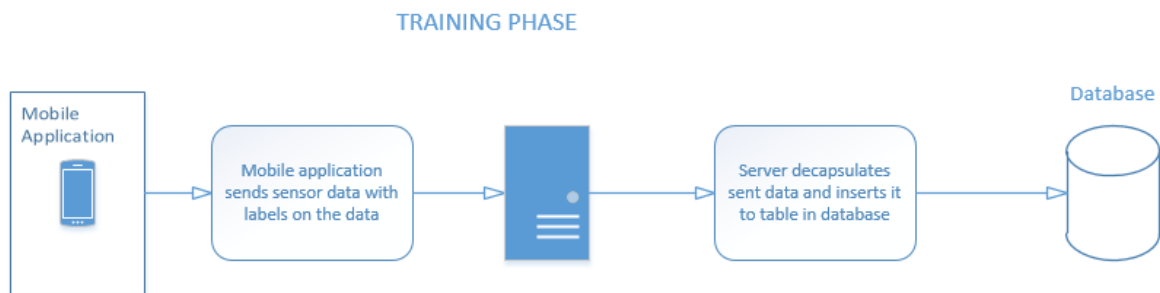


Figure 4-2 Data collection stages

The mobile applications send the data using one of the proposed methods. The mobile application then encapsulates the data for sending through the internet to the server. When the server receives the data the server decapsulates the data and can now insert this data into a table in the database. Some data transformation is performed at a later stage to make the data more compatible with the learning algorithms. The tables in the database were converted to comma separated value files. These can be easily taken in by the models using pandas or numpy libraries.

4.2.4 System Architecture

The system architecture is a conceptual model of the high-level structures of a software system. The system architecture gives a view of the system before it is built. This allows for design decisions that would have otherwise been costly to make at the implementation stage to be performed. The architecture shows how the different structures of the software system interact with each other. The system designed for this research is composed of the mobile application and the back-end system. The back-end system is composed of the server, a classifier, a database and a pattern analysis algorithm. The architecture shows how these components communicate with each other and achieve the aggregation

of context. Figure 4-3 shows the different components of the system and how they communicate.

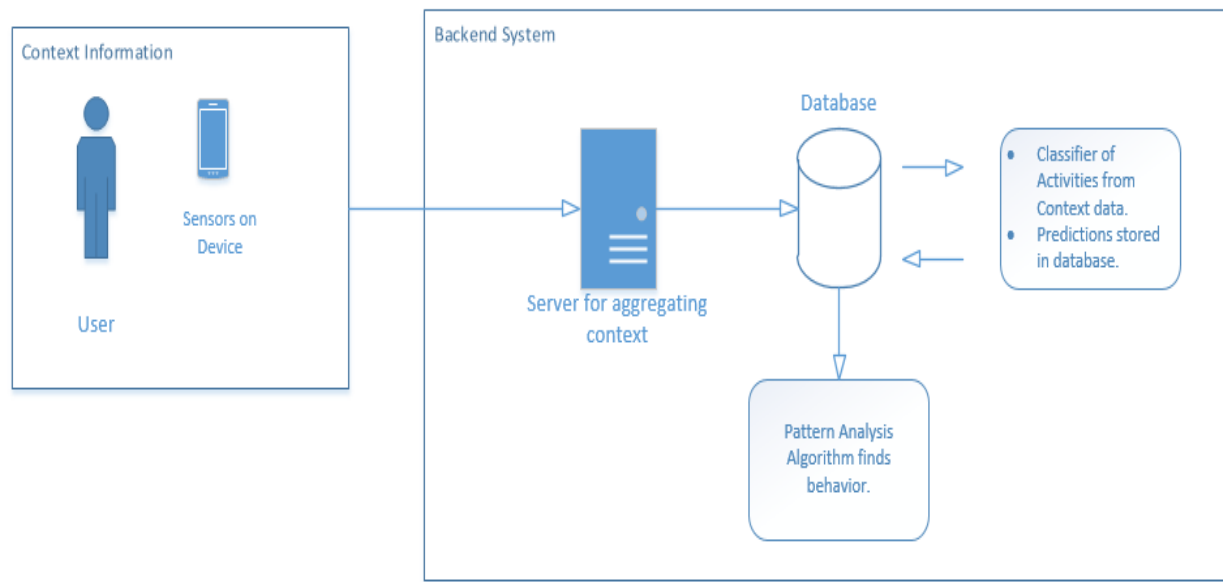


Figure 4-3 System Architecture

The first component the mobile application is a tool designed for acquiring and gathering the context information. The sensor data it acquires from the device sensors is used as the context information. As stated in the methodology, evolutionary prototyping is used; a prototype of the application is first developed. The prototype is used for collecting training data in a controlled manner. When the initial data collection is complete then the prototype is modified to develop the final product. The final mobile application is capable of running as a background application without hindering the user's normal activities on their device. The next structure of the system is the back-end system. The back-end system is responsible for collecting and aggregating the context of multiple devices and analysing the data for patterns. The back-end comprises of four major parts working together:

1. The server
2. A database
3. A classifier

4. A Pattern Analysis algorithm

The programming languages chosen for the back-end will be based on a review of the literature and the skill set of the researcher. The server is responsible for receiving the encapsulated context data from the Sensor application on the mobile device. It is responsible for adding the context data to the appropriate table in the database. The database is used to store the data in an organized manner. For the training phase, the database table includes a Class label that is used for training the algorithms. After the training phase, the tables do not have a class label and the classifier must deduce the label from the learned information. The classifier is an algorithm that performs classification of certain phenomena. A classifier can also be defined as an algorithm that maps input data to a category. The system uses the classifier to map the context information to the activity being performed by the user. The classifier is trained using the training data that has labels which were collected during the training phase of the mobile application. The classifier takes the tables as datasets and converts them to datasets. The classifier outputs the predictions it makes and will be tested on its accuracy and the confusion matrix.

The accuracy score is the percentage times that the classifier managed to correctly classify a particular feature. The confusion matrix is a matrix that shows the correctly classified features and the features that were not. The matrix shows where the incorrectly classified features were placed and this can help in observing if there may be a problem with the data itself. Three different classifiers are then tested, the Decision Tree (DT), KnearestNeighbours (KNN) and Support Vector Machines (SVM). The classification results given by the classifiers will be discussed in the results chapter.

The last structure of the system is the pattern analysis. Pattern analysis is defined as a field of machine learning concerned with recognition of regularities in noisy or complex environments. It can also be defined as a classification of input data by extracting important features from a lot of noisy data. For the purpose of this research, noisy data refers to the different activities that are found and the system attempts to find only the important features. The pattern analysis algorithm will be used to observe these features

and this will be documented as graphs, reports, and tables. This stage answers the final research question of the research and testing of the system as a whole is done.

4.2.5 System Use Cases

The use cases show the different interactions between the user and the system. The use cases also show the interactions between the system and its components. In some instances, a component can be an actor triggering a function in another component.

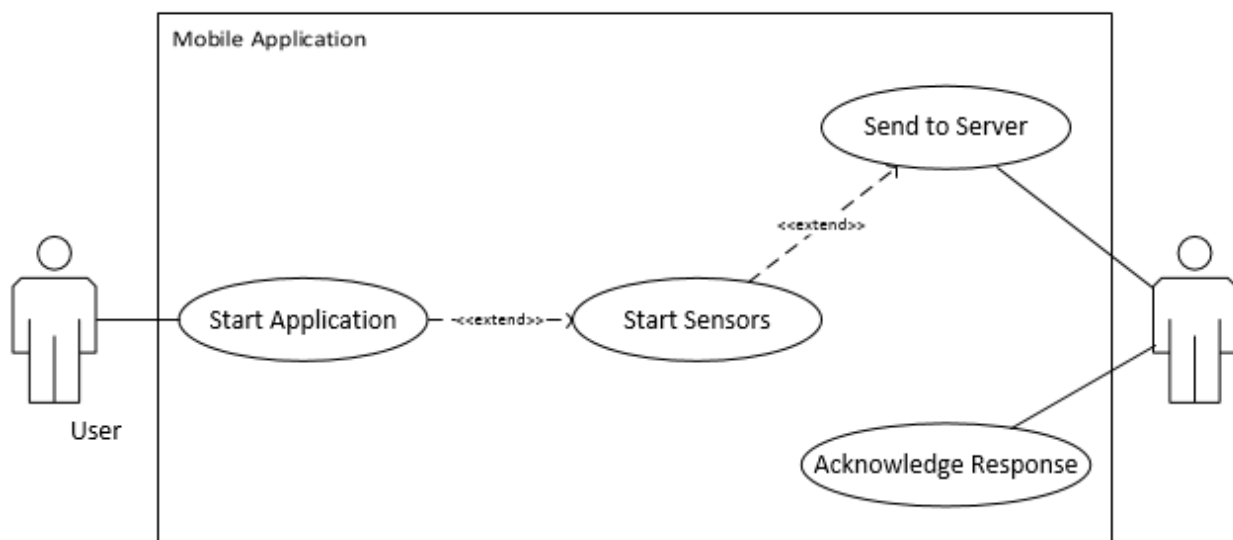


Figure 4-0 System use case

Figure 4-4 shows the use case diagram as a whole of the user interacting with the mobile application and the server. The actors involved are the user and the server. The user launches the application when the application starts the sensors will also be started and their data sent to the server. The use cases breakdown the above system use case into the user use case and the server use case.

Use case 1

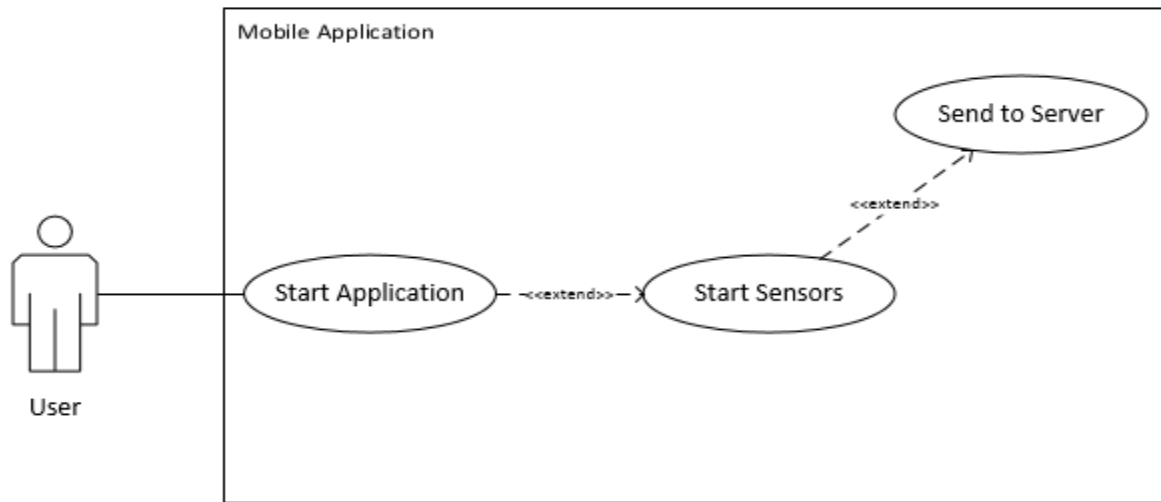


Figure 4-3 Starting the Application

Name	Start Application
Brief Description:	The user must have the application installed on their device. The user launches the application by clicking on the icon. The application will launch and start the relevant sensors. The sensor information will then be sent to the server for aggregation.
Precondition(s):	Android Device. Internet Availability.
Assumption(s):	Device has all required sensors
Step-by-Step Description:	<ol style="list-style-type: none"> 1. User launches application. 2. Application registers sensors. 3. Sensor data Encapsulated. 4. Encapsulated data sent to server.

Post-Condition(s)	Steps 3 and 4 may be repeated or Application Disabled

Use case 2

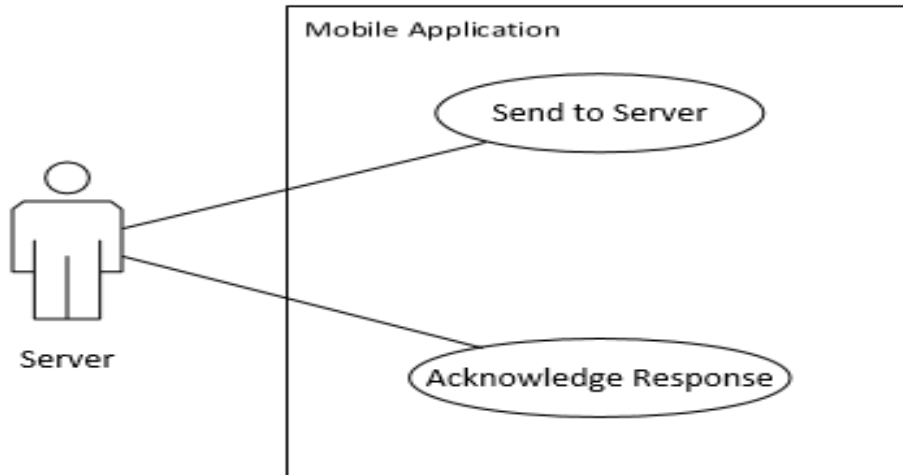


Figure 4-5 Server interaction

Name	Server Interaction
Brief Description:	The server gets information sent to it from the mobile application. The server stores the information in the database for analysis. The server then sends a response to the application acknowledging receipt of the information.
Precondition(s):	One or more devices Connected
Assumption(s):	Database successfully connected
Step-by-Step Description:	<ol style="list-style-type: none"> 1. Connects to application. 2. Receives sent data. 3. Decapsulates data. 4. Stores data in the database.

	<ul style="list-style-type: none"> 5. Send a response to the application 6. Disconnects to application
Post-Condition(s)	Disconnects to the device to allow more devices to connect meanwhile.

4.3 Testing and Results

The system is tested using functional testing. The system is tested as individual components against their individual requirements. The research uses an iterative and incremental development hence if a requirement was not satisfied the system is redesigned and tested again. The testing criterion for the mobile application is the satisfaction of its requirements.

The back-end is also tested on the requirements but its individual components must also be tested first. The percentage accuracy will be used to test the algorithms the better the score the more accurate the classification. The classifiers are tested against each other and the results are discussed in the testing and results chapter. The system is tested in a real world environment and then the results were studied and any observations were noted. The results are shown using graphs and tables, some are discussed from the discovered observations. The results should answer the last research question and also satisfy the problem statement of the research. The system is used as proof of concept of the research in relation to the last objectives. The results lead to a discussion at the end of the research to address the problem statement and how the results can be used to develop the system for future work and its possible applications.

4.4 Mobile Application

As mentioned in the design, the mobile application is developed in two phases. The first being as a data collection tool and the second being as a background application. The mobile application is required so as to be able to use the sensors on the device to collect

their values. The values are then used as context information of the user and sent to the back-end system.

4.4.1 Development Platform

The mobile application was developed on the Android platform. This platform was chosen based on the availability of the platform to the researcher and its general availability to users. Google and Apple have taken over the mobile device market. They constitute over 90% of smartphone sales worldwide. Google developed Android and has had a major impact as a mobile platform accounting for 80% of sales including their high range and low range smartphones.

The iPhone Operating System developed by Apple accounts for approximately 13% of smartphones sales globally. This means these two platforms alone constitute 93% of smartphone sales globally making them the most common mobile platforms available. The market for smartphones has also supported the growth of these two platforms. Smartphones have gradually become more popular than laptops and tablets. With continued miniaturization and effects of Moore's law smartphones can now offer the same functionalities of these devices.

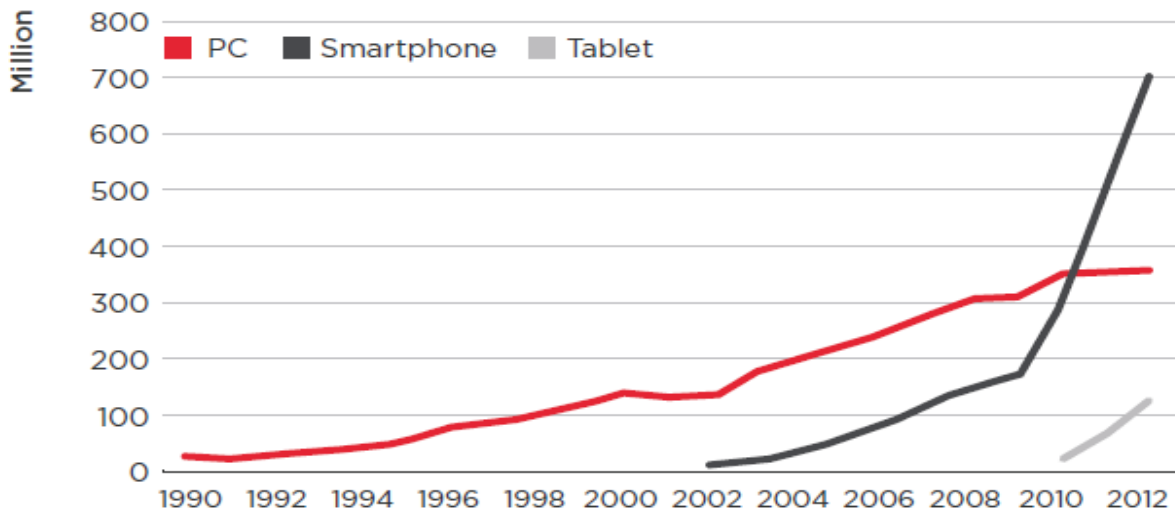


Figure 0-6 Graph of sales of different Devices (Gsm Intelligence 2014)

The Android OS started with very little to no share of the market in the early 2009s. With time this grew substantially and has continued to grow. Figure 4-7 shows the mobile platform share evolution showing the different growth patterns of iPhone and Android. Many mobile device companies use the platform on their devices such as Samsung, HTC, Fire Phone, Sony, and others. What remains impressive is iOS although dealing with mostly high-end devices has maintained its share and even grown over time. It's the sole major competitor of Android OS as other manufacturers have been left to share a small portion of the shares.

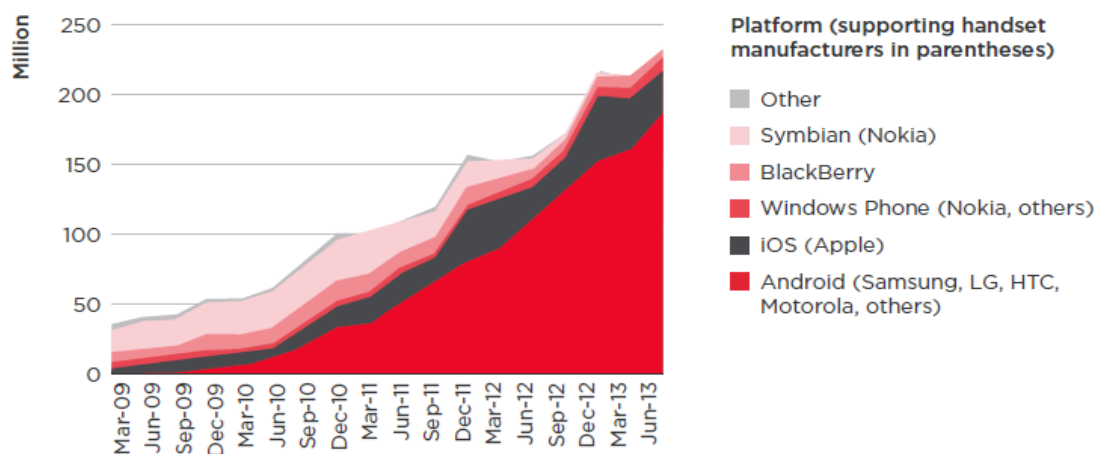


Figure 0-7 Graph of platform market shares (Gsm Intelligence 2014)

Besides being a popular OS, one of the major reasons for choosing Android OS is its availability to many users. Android can be found on high-end phones which are expensive and the cheaper low-end phones. This makes it available to a large number of individuals. iOS, as mentioned, holds a considerable amount of sales but it compensates having a smaller market share with mostly offering high-end devices. iPhone prices, for example, have maintained a price range much higher than Android. What this basically means is that only a select group of individuals can afford to buy such a device making them not available to many people. The system developed for this research is meant to be available to a large number of users to get more complex behaviors.

Another reason for choosing Android over iOS was that it is open-source. The Android OS is based on the Android Open Source Project (AOSP). This means that people can take the source code and modify it to create custom operation systems. Different devices

can be running Android but different custom systems. This gives users a choice on which Android-based operating system they would prefer increasing its popularity. The openness of Android has attracted developers to design applications. Google has given developers many tools, APIs and SDKs thereby making development easier on many platforms such as Windows, Ubuntu, and Mac. iOS on the other hand is closed-source and developers can only use Xcode to develop mobile applications for the Apple devices. Xcode is best used on a Mac computer and these can be expensive. The other options of developing IOS mobile applications is by renting a remote Mac computer or using a Virtual Machine (VM). Using a VM is complicated for a person without the technical skills to set it up and load the Mac OS. Another issue with using the virtual machine is that it is illegal; Apple does not sell their OS as a stand-alone product. The reasons explained above gave rise to the decision to use Android as the development platform instead of iOS and other platforms. The application was developed using the Android Studio Integrated Development Environment (IDE). Android studio is the official IDE from Google and includes most of the tools for Android application development. The IDE is available for all major operating systems, Windows, Mac, and Linux. The application components are explained in this chapter.

4.4.2 Android Activity

The Android activity is a component in an Android application that is the core thing that the user can do. The user interacts with the activity in some way. Interfaces are linked to the activity and how they function is implemented in the activity. The activity in some way controls the functionalities of the application. When building an application, the application must extend the activity class. Within the activity class there are a few core methods namely `OnCreate()`, `OnStart()`, `OnResume()`, `OnPause()`, `OnStop()` and `Onrestart()`.

The `OnCreate()` method is called when the application first instantiates. It initializes the views and other core functions. For the mobile application created under the name “Sensor App” the first thing done is to create the Sensor App class that extends the activity class and implements the `SensorEventListener` interface.

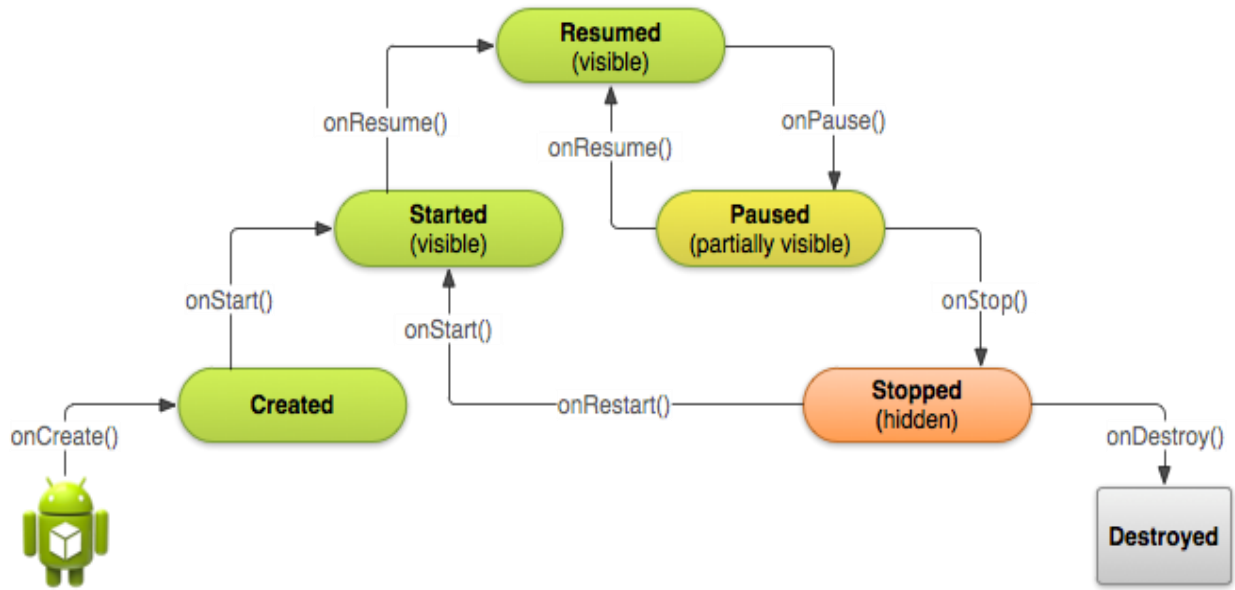


Figure 4-8 Android Activity LifeCycle

The required imports were done also before the application began development. The next stage was to declare all the necessary variable names. The variables will be used to store values and some as class objects.

```

public class SensorActivity extends Activity implements SensorEventListener {

    private SensorManager sensorManager;
    private SensorManager lsensorManager;
    private SensorManager psensorManager;
    private LocationManager locationManager;
    private LocationListener locationListener;
    private float x = 0;
    private float y = 0;
    private float z = 0;
    private float lux = 0;
    private float prox = 0;
    private Handler handler;
    private TextView currentX, currentY, currentZ, currentlight, currentprox;
    Sensor accelerometer;
    Sensor lightsensor;
    Sensor proximity;

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}
  
```

Listing 2 Declaration of special variables

Listing 2 shows the different declarations of variables for the application. The Sensor manager variables are used later in the application to get an instance of the Sensor manager class. The Sensor manager class lets the application get access to the device's sensors. The float variables will be used to store the numerical values received from the sensors. The variable names “x”, “y” and “z” represent the gravitational forces on the x, y and z planes of the device, “lux” for the light intensity and “prox” for the proximity sensor. Each of the sensors being used is also declared. One of the first things the onCreate() method does is to call the setContentView(). This method creates the window in which the layout that contains the interface of the application. Figure 4-9 shows the interface of the application during the training and data collection phase. The interface is made up of two buttons and some checkboxes for selecting the activity performed.

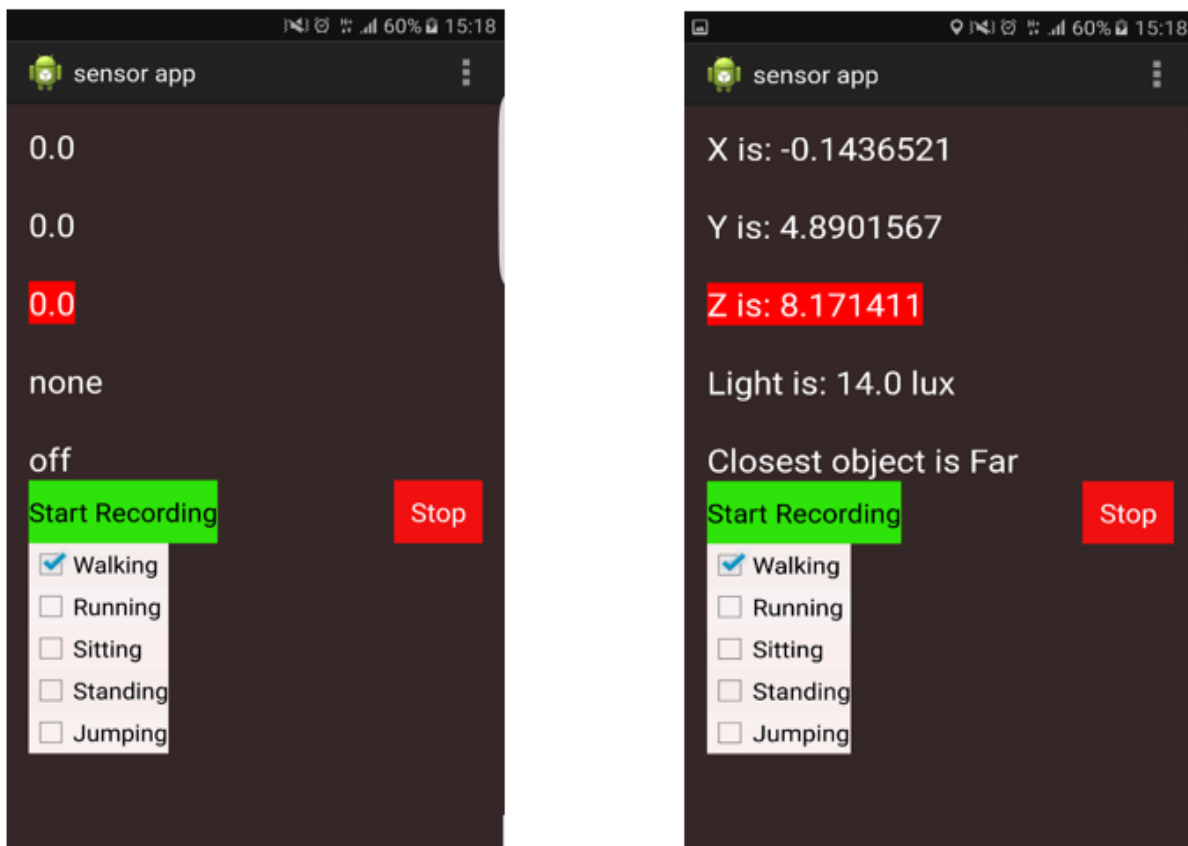


Figure 0-9 Screenshots of application

One button is for showing the sensor values and the other for stopping the showing of the values. These buttons are declared in the `OnCreate()` Method and use the `setOnClickListener()` method to perform their functions when clicked. The checkboxes are selected by the user based on the activity they may be performing. Only a few activities were chosen for the research and are all based on basic physical activities most users perform. The interface shows the values of each of the sensors. The X, Y and Z of the accelerometer the intensity of the light in lux and finally the proximity of the closest object, whether it's far or near.

4.4.3 StartSensors Method

The `startSensors()` method is invoked when the start recording button is clicked. This method uses the sensor manager to get access to the sensors on the device. The instances of the sensor manager class are initialized using the `getSystemService()` method. An instance of this class is made for all the sensors that are going to be used in the research. The sensors chosen for the research are the Accelerometer, Proximity sensor, Light sensor, Global Positioning System (GPS).

The location manager cannot be initialized directly like the other sensors. The constructor for this class is hidden and cannot be seen by the IDE. This is because unlike the other sensors the location manager requires platform specific configuration. The class is initiated and the values are updated by the `onLocationChanged` method. Listing 3 shows the steps taken in acquiring the values. The variable `location` of type `Location` is used to get the current Longitude and the Latitude.

The next stage is to register a listener for each of the sensors. This is done using the `registerListener()` method. Before the listeners can each be registered the application must first check if the sensors are available on the device. This is done by using if-statements as shown in Listing 3, the if-statements check when the application requests the sensor it will not receive a null response.

```

locationListener = new LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        if (location != null) {
            Toast.makeText(getApplicationContext(),
                "Location changed : Lat: " + location.getLatitude() +
                " Lng: " + location.getLongitude(),
                Toast.LENGTH_SHORT).show();
        }
        lat = location.getLatitude();
        lng = location.getLongitude();
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onProviderDisabled(String provider) {
    }
};

```

Listing 3 Location listener

If the sensor is available, then it the listener is registered using the sensor managers. When all the relevant sensors are registered to their respective listeners the next stage is to have the sensor values displayed. The sensor manager needs a way to communicate to the application that the sensor value has changed. This is done by the method `OnSensorChanged()`. It takes in a sensor event and will perform the appropriate functions when a sensor value changes.

```

locationListener = new LocationListener() {...};
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
//gps sensor activated

if (sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null) {
    // success! we have an accelerometer

    accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

    sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_GAME);
    //vibrateThreshold = accelerometer.getMaximumRange() / 2;
}
if (lsensorManager.getDefaultSensor(Sensor.TYPE_LIGHT) != null) {

    lightsensor = lsensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);

    lsensorManager.registerListener(this, lightsensor, SensorManager.SENSOR_DELAY_GAME);
}
if (psensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY) != null) {

    proximity = psensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);

    psensorManager.registerListener(this, proximity, SensorManager.SENSOR_DELAY_GAME);
}

```

Listing 4 Registering all sensors

The application requires the method to first check which of the sensor values has changed whether it is the accelerometer or another sensor. It does this using if-statements checking which sensor the event belongs to. Once the sensor has been determined then using the event object the application can get the value and store it in the correct variables for that sensor. If it's the accelerometer it stores the values in the variables x, y and z.

```

public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        // clean current values
        displayCleanValues();

        x = event.values[0];
        y = event.values[1];
        z = event.values[2];
        // display the current x,y,z accelerometer values
        displayCurrentValues();
    }
}

```

Listing 5 OnSensorChanged method

Once the values have been stored the method `displayCurrentValues()` is called. This method is responsible for updating the values displayed by the user interface. The method updates the `TextView`s of the changed values and displays them to the user. Listing 6 shows a section of the code that is used to display the new values on the user interface.

```
currentX.setText("X is: " + Float.toString(x));
currentY.setText("Y is: " + Float.toString(y));
currentZ.setText("Z is: " + Float.toString(z));
currentLight.setText("Light is: " + Float.toString(lux) + " lux");

if (prox <= 0) {
    currentProx.setText("Closest object is Near");
} else {
    currentProx.setText("Closest object is Far");
}
```

Listing 6 Updating of Views

The last stage of the `StartSensors()` method is a `Runnable` method. The `Runnable` method allows for the piece of code to be executed in another thread. A thread is a unit of code that is executed concurrently to the other threads. Generally, an application is executed in the main thread but can have other threads that are run concurrently to it. The threads are useful in order to keep the time intervals and send the sensor information within these intervals. The threads are best used in asynchronous environments where a regular time relationship is associated with a particular event. The `Runnable` class is used to run a piece of code that is used to send the sensor values to the back-end system. The back-end system is discussed in the system architecture. The `Runnable` class makes use of a handler that is used to delay the sending of the sensor data to the back-end. A handler is bound to a single thread and its message queue. It delivers messages and `Runnable`s to the thread's message queue to be executed. In Android, it can be used to perform actions scheduled to be done in the future.

The handler uses an interval of 500 milliseconds (ms) between each time it sends sensor data. The choice of the time will be discussed later when the classifiers and the datasets are discussed. The `Runnable` class calls the `AsyncTask` method after every 500ms. The `AsyncTask` method takes the Uniform Resource Locator (URL) of the method on the server that receives the sensor data as context.

```

Runnable runnable1 = () -> {

    //the runnable starts after another 500ms
    handler.postDelayed(this, 500);
    //insert code here for sending the sensor information to server
    new HttpAsyncTask().execute("http://10.42.0.1:5002/addaccelerometer"); //10.42.0.1 172.20.56.66/
}

```

Listing 7 Runnable Thread

Listing 6 shows the runnable method and how the handler calls the postDelayed methods and sets the 500ms time interval. The httpAsyncTask() method calls the execute method which takes the URL and gives the method for the server. httpAsyncTask() extends the Async Task class. This method has a method called doInBackground() that executes another method called POST. Once the httpAsyncTask has executed the code that is in its body the second method of this class onPostExecute() is called. The method is responsible for executing code after the httpAsyncTask has executed its doInBackground() code. The onPostExecute() gives a toast to notify the user that the data has been sent to the server.

The POST method is responsible for the sending of the sensor values to the server. The method handles the client-server request and response aspects of the application. It begins by creating an HTTP client then the HTTP post instance from the httpPost class. For the research, the data will be sent as a JavaScript Object Notation (JSON) object. JSON is a data inter-change format that is easy for machines to parse and generate. JSON objects use text format which in turn makes it language-independent making it the best option as a data-interchange language. The POST method handles sending the data as a JSON object. The JSON object is created by using the accumulate() method and then appending the different values of the sensors. An added field for the class label is also appended for the purpose of having labeled data in the dataset used for the Classifier. Within the POST method, the values for each of the sensors are found from the variables containing these. The label variable is declared in the POST method and is given a value depending on the checkbox that is marked. If no checkbox has been checked then the application will create a toast to asking for a checkbox to be marked.

```

// 3. build jsonObject
JSONObject jsonObject = new JSONObject();
jsonObject.accumulate("accelerometerX", x);
jsonObject.accumulate("accelerometerY", y);
jsonObject.accumulate("accelerometerZ", z);
jsonObject.accumulate("Light", lux);
jsonObject.accumulate("Proximity", currentprox.getText());
jsonObject.accumulate("longitude", lng);
jsonObject.accumulate("latitude", lat);
jsonObject.accumulate("label", label);

// 4. convert JSONObject to JSON to String
json = jsonObject.toString();
Log.v(TAG, "json= " + json);

// 5. set json to StringEntity
StringEntity se = new StringEntity(json);

// 6. set httpPost Entity
httpPost.setEntity(se);

// 7. Set some headers to inform server about the type of the content
httpPost.setHeader("Accept", "application/json");
httpPost.setHeader("Content-type", "application/json");

```

Listing 8 Post Method

A log message is used by the researcher to verify if the information being sent is correctly formatted. Each time a new JSON object is created the object's contents are logged in Android Studio's Log cat. The next stage is to add the JSON object to a string entity that will be used and finally add headers. A string entity refers to the raw data that is sent to the server. This raw data is sent as part of the request to the server and the server can then parse it and give the appropriate response. The string entity is then added to the httpPost object. This is the object that will be sent to the server. The headers are also added to the httpPost object. The headers tell the server what type of data it is receiving. The application will now just need the response from the server to show if the data was sent successfully. After the execution of the POST method and the AsyncTask the handler sets another delay of 500ms before the next execution. With this being done the methods of the StartSensors method have all been completed.

4.4.4 StopSensors Method

This method is invoked when the Stop button is clicked. This method is responsible for clearing the interface of all sensor values and also for stopping the sensor manager class from sending updates of the sensors. This allows the user to change the activity they are performing or to stop the application from running. The method takes the View as its parameter.

```
public void stopSensors(View v) {  
    sensorManager.unregisterListener(this);  
    lsensorManager.unregisterListener(this);  
    psensorManager.unregisterListener(this);  
    locationManager.removeUpdates(locationListener);  
    locationManager = null;  
    displayCleanValues();  
}
```

Listing 9 Stop sensors method

The accelerometer, light, and proximity sensors are all unregistered using the `unregisterListener()` method. Listing 8 shows each of the sensor manager using the `unregisterListener()` method to stop the sensors. The method takes the context as a parameter, in this case, it's the activity. Since the location manager was instantiated in a different way to the other sensors, stopping the sensor is done differently.

The `removeupdates()` method is used first, this method removes the updates from the location manager. The method takes in the location listener as its argument. This alone was discovered to not be enough to completely stop the sensor. It was found that the manager was still bound to the sensor. Setting the location manager value to null then forces the application to stop using this sensor. With this, all the sensors are stopped but the values on the screen must be reset to default. The method `displayCleanValues()` is called. This method is responsible for clearing the values and setting default values.

4.4.5 Manifest File

The Manifest file is one of the most important documents in the Android application. It contains information about the application that is highly important to the Android system. The Android system requires this information before it even starts to execute any of the code. It is an XML file. It contains the package name of the application which uniquely identifies it. It gives the minimum Android API that is required to run the application. It describes the different components of the application such as Activities, Content providers, and Services.

```
<uses-permission  
|   android:name="android.permission.ACCESS_FINE_LOCATION" />  
<uses-permission  
|   android:name="android.permission.INTERNET" />
```

Listing 10 Permissions used

The Manifest file also gives the permissions that the application needs to access certain APIs. The permissions allow the application to access protected parts of some of the APIs. Such permissions are required in this application. The application requires sending sensor data to the server over the internet. The application also needs the location of the user and this requires permission also. Listing 10 shows a code snippet of the Manifest file showing the permissions required by the application. The uses-permission tag specifies the permission that is needed within its body. The ACCESS_FINE_LOCATION allows for the application to get the precise GPS coordinates of the user. The INTERNET permission allows the application to get onto the internet or perform network operations.

4.5 The Context Aggregation Server

The Server is responsible for receiving the context information from the devices running the Android application. The server must receive the information as JSON and parse it to perform the required functions. The server for the back-end system is run on a desktop

computer. The server implementation was approached using two different methods. In the design chapter, the Server was proposed to be implemented as a web server using Apache or using Python. Apache was discovered as the more advised method. This was discovered by reading forums by other programmers. Apache is more scalable and can handle multiple requests at the same time. The major problem was for the purpose of the research not a lot of devices are used. It was then decided to use a lightweight web server and implement the Apache web server for future work. For the purpose of the research Flask was chosen. It is a micro-framework for python and allows for the creation of a lightweight web server. The Flask micro framework is based on Werkzeug. Werkzeug is an HTTP and WSGI utility library for python and Jinja 2 which is a templating language. Flask web servers can be implemented in one python document and does not come with a native database. Flask can provide a connection to a MySQL database to overcome not having a native database.

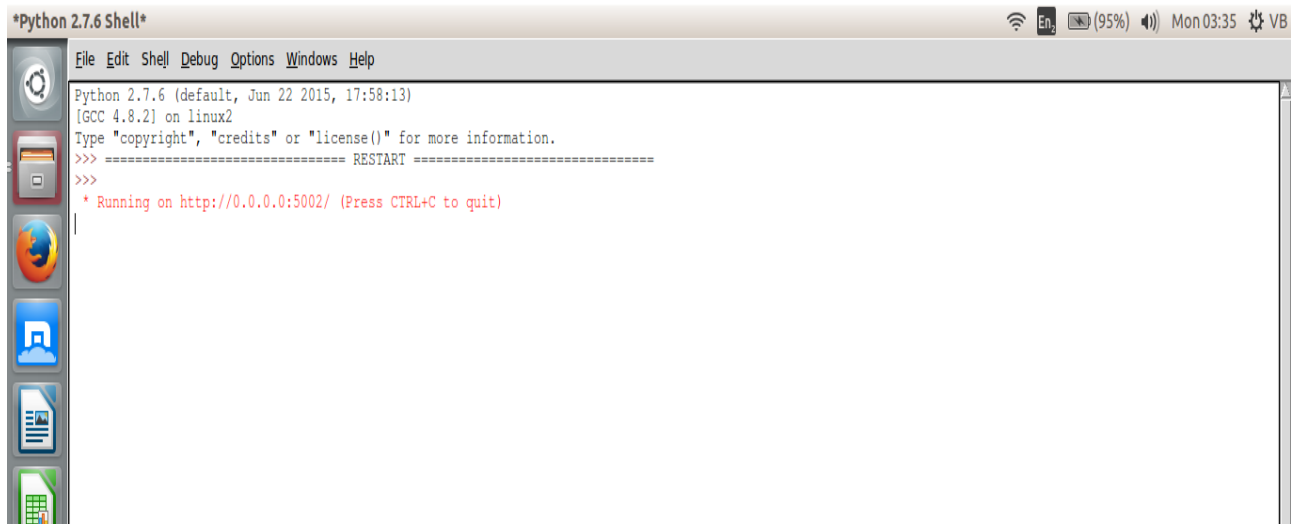
The server was implemented completely using the python programming language. Coding was done using IDLE which is an IDE for python running on Linux. The first thing was to import the required modules including Flask and its modules one of these being the MySQL connector. The MySQL connection was configured to allow the web server to perform database operations. The Listing 10 below shows the configuration of the MySQL connection.

```
# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = '*****'
app.config['MYSQL_DATABASE_DB'] = 'Backend'
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
mysql.init_app(app)
```

Listing 10 Database Configurations

The MySQL database requires a password and since it's running on a desktop computer then the host is the localhost. This then allows the web server to access the database and perform additions and/or removals from the tables within the database. A test page is created to check if the web server is online. The page shown just prints the word "Welcome" and the default route leads to this page.

Figure 4-10 show the server running with the address on port 5002. The 0.0.0.0 address means all the ipv4 addresses that are available on the host. If the host has two IP addresses, then the server will still be available on both addresses. At this point no requests have been made to the server hence it just shows its running status.



```
*Python 2.7.6 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
* Running on http://0.0.0.0:5002/ (Press CTRL+C to quit)
```

Figure 4-10 Screenshot of running server

The next application route is the one responsible for handling any POST or GET requests. The POST method submits data to be processed to a specified resource. GET is used for viewing something, without changing it, while POST is used for changing something. This is the route that is added to the URL when the mobile application is sending the context information. Listing 11 shows the method and how it is structured. The URL for the server would be the IP address of the host, the port, and the app route which in this case would be `addaccelerometer_`

```

# Add the Sensor data to the Database
@app.route('/addaccelerometer',methods=['POST','GET'])
def add_entry1():
    try:
        conn = mysql.connect()
        cursor = conn.cursor()
        print('connected')

        req_json = request.get_json()
        print('request.get_json() method',request.get_json())

        cursor.execute("INSERT INTO backend (xAxis, yAxis, zAxis, lux, proximity, lable)
VALUES (%s, %s, %s, %s, %s, %s)",
        (req_json['accelerometerX'], req_json['accelerometerY'], req_json['accelerometerZ'], req_json['Light'], req_json['Proximity'], req_json['label']))
        print('insert code done')

        conn.commit()
        print('New entry was successfully posted')
    except Exception as e:
        print(e)
        print('failed to add')

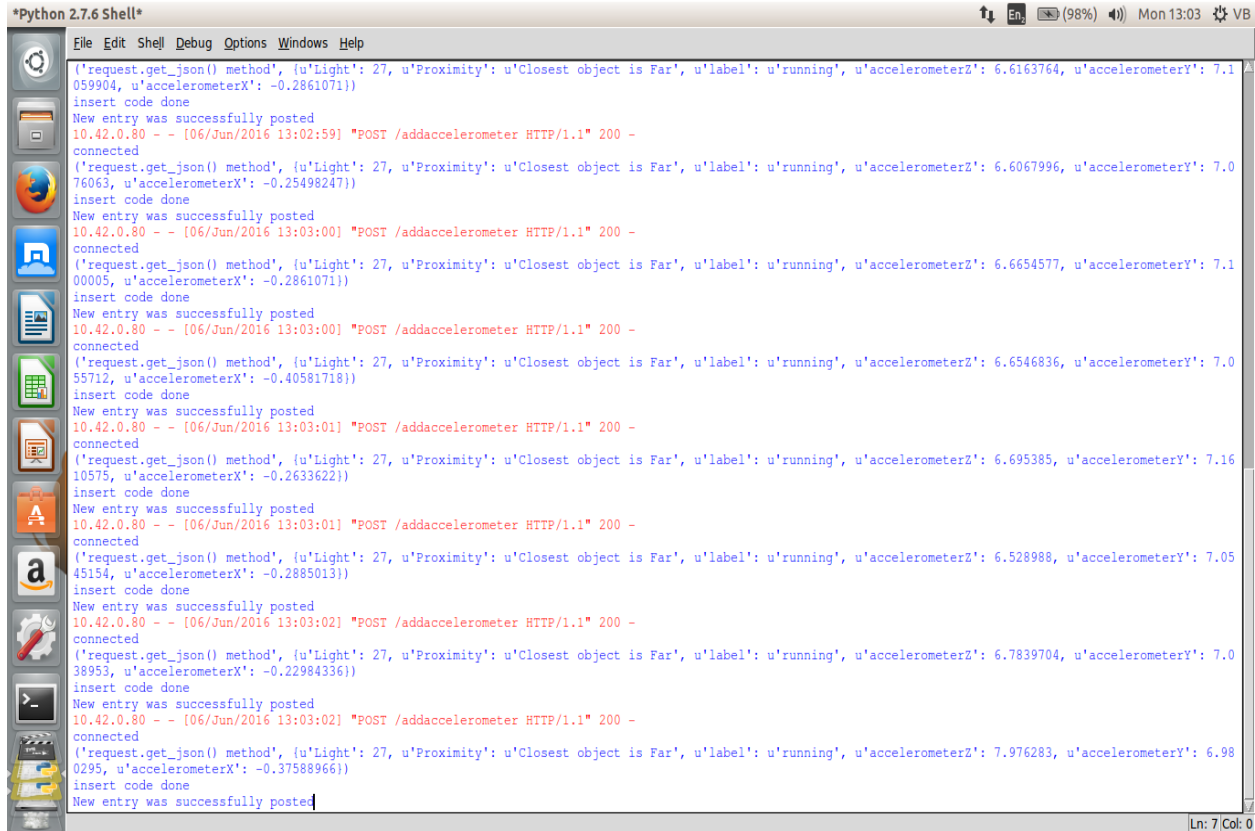
    finally:
        cursor.close()
        conn.close()
        return "Added to Database."

```

Listing 11 aggregating context

The database connects to the localhost because the server and database are on the same physical machine. The connection to the database is done within a try block in the case that an error occurs with the connection or insertion of data an exception will be thrown. This is useful in this case because instead of the entire server crashing because of the error, the server will show only the error but keep running and accepting requests. A cursor is used to traverse the rows in the database and allow for querying to be done easily. The request object is used to extract the JSON object. The request is sent by the mobile application and contains the sensor data and headers that give the content type to be received by the server. The server then prints out the JSON object, this is used to verify that the content has been successfully been sent from the mobile application. This is shown in Figure 4-11. The blue text represents the print commands in the method. The first line after every request shows that the device is connected to the server. The next line is the JSON object with all the attributes. The cursor is then used to insert the sensor values in the correct table and in the correct fields. Using the execute() method the values are inserted into the table back-end on the database. The last two lines in the server show this when each value has been inserted in the in each of the columns it prints “insert code

done”. Lastly, the commit command then finalizes the insertion to the table and changes are made to the database. When the commit is successful with no errors then again the server prints “New entry was successfully posted”.



```
*Python 2.7.6 Shell*
File Edit Shell Debug Options Windows Help
('request.get_json() method', {'u'Light': 27, u'Proximity': u'Closest object is Far', u'label': u'running', u'accelerometerZ': 6.6163764, u'accelerometerY': 7.1059904, u'accelerometerX': -0.2861071})
insert code done
New entry was successfully posted
10.42.0.80 -- [06/Jun/2016 13:02:59] "POST /addaccelerometer HTTP/1.1" 200 -
connected
('request.get_json() method', {'u'Light': 27, u'Proximity': u'Closest object is Far', u'label': u'running', u'accelerometerZ': 6.6067996, u'accelerometerY': 7.076063, u'accelerometerX': -0.25498247})
insert code done
New entry was successfully posted
10.42.0.80 -- [06/Jun/2016 13:03:00] "POST /addaccelerometer HTTP/1.1" 200 -
connected
('request.get_json() method', {'u'Light': 27, u'Proximity': u'Closest object is Far', u'label': u'running', u'accelerometerZ': 6.6654577, u'accelerometerY': 7.100005, u'accelerometerX': -0.2861071})
insert code done
New entry was successfully posted
10.42.0.80 -- [06/Jun/2016 13:03:00] "POST /addaccelerometer HTTP/1.1" 200 -
connected
('request.get_json() method', {'u'Light': 27, u'Proximity': u'Closest object is Far', u'label': u'running', u'accelerometerZ': 6.6546836, u'accelerometerY': 7.055712, u'accelerometerX': -0.40581718})
insert code done
New entry was successfully posted
10.42.0.80 -- [06/Jun/2016 13:03:01] "POST /addaccelerometer HTTP/1.1" 200 -
connected
('request.get_json() method', {'u'Light': 27, u'Proximity': u'Closest object is Far', u'label': u'running', u'accelerometerZ': 6.695385, u'accelerometerY': 7.1610575, u'accelerometerX': -0.2633622})
insert code done
New entry was successfully posted
10.42.0.80 -- [06/Jun/2016 13:03:01] "POST /addaccelerometer HTTP/1.1" 200 -
connected
('request.get_json() method', {'u'Light': 27, u'Proximity': u'Closest object is Far', u'label': u'running', u'accelerometerZ': 6.528988, u'accelerometerY': 7.0545154, u'accelerometerX': -0.2885013})
insert code done
New entry was successfully posted
10.42.0.80 -- [06/Jun/2016 13:03:02] "POST /addaccelerometer HTTP/1.1" 200 -
connected
('request.get_json() method', {'u'Light': 27, u'Proximity': u'Closest object is Far', u'label': u'running', u'accelerometerZ': 6.7839704, u'accelerometerY': 7.038953, u'accelerometerX': -0.22984336})
insert code done
New entry was successfully posted
10.42.0.80 -- [06/Jun/2016 13:03:02] "POST /addaccelerometer HTTP/1.1" 200 -
connected
('request.get_json() method', {'u'Light': 27, u'Proximity': u'Closest object is Far', u'label': u'running', u'accelerometerZ': 7.976283, u'accelerometerY': 6.980295, u'accelerometerX': -0.37588966})
insert code done
New entry was successfully posted
Ln: 7 Col: 0
```

Figure 4-11 Server receiving context information

In the case, an error occurs with any of these commands then the exception is printed out with the text “failed to add”. The point of these logs is that in the case the exception is thrown then the error in the server can easily be found. This is done by checking how far the execution of the try block went using the logs. The last step is to close the cursor and the connection. Using the final keyword makes the closing commands to always be executed by the server allowing other transactions to be done. The data is stored in that database in the format in Figure 4-12.

xAxis	yAxis	zAxis	lables
-10.4411	0.353145	4.43047	sit
-9.086	10.8553	-5.42885	walking
-8.13071	14.0528	-2.478	walking
-7.91643	-1.87466	5.96276	sit
-7.88889	-1.02113	7.00184	sit
-7.77517	-0.849942	6.91445	sit
-7.66623	-2.28646	5.79636	sit
-7.64828	-2.16915	5.67067	sit
-7.58722	-2.27808	5.78439	sit
-7.58483	-2.23858	5.82988	sit

Figure 4-12: Extract of database information

Since the data was being fed directly to the database in a monitored environment it is sorted and has no missing values to it. The data is used in the other steps of the system as is, no transformations were required

4.6 Classification

Once the data has been sent to the server and aggregated the back-end system should then predict the activity being performed. The system uses a classifier for this functionality that will first be trained then tested on its accuracy and other metrics. The research could not fully rely on one classifier as some work best with certain types of data. The classifiers selected for this research are as follows:

1. Support Vector Machine
2. Decision Tree
3. KnearestNeighbors

The focus of the research is not specifically on the internal functionalities of the classifiers. This lead to the decision of using the classifiers offered by Scikit-learn. Scikit-learn is open source and offers tools that can be used for data mining and analysis. It is built on the following libraries:

1. NumPy

2. SciPy
3. Matplotlib

These are downloaded and added to the python libraries. They are later imported for use during the coding of the classifier. The three classifiers are implemented using one python file and the same training data sets. This is done to make sure that the results that the classifiers give are not biased towards on classifier getting a dataset easier to learn. The data sets are created using the table from the database but the tables must first be extracted from the database. To do this a script was designed that uses a python library known as a dataset. The dataset is an easier way to load databases or tables to python. The table from the back-end database is extracted and converted to a .dot file which is compatible with the classifiers.

```
1 import dataset
2 db = dataset.connect('mysql://root:*****@localhost/Backend')
3 # export all data
4 result = db['backend'].all()
5 dataset.freeze(result, format='dot', filename='backend.dot')
6 print('done!')
7
```

Listing 12 Conversion of table to dataset

This file that's produced is a flat file that will be loaded to the classifier. All the sensor information is stored in this file for one device. Another column is added which gives the class label that will be used in training for the classifier. The class label is the one that was selected by the user when using the application. Once the dataset has been created the next step is to create the classifiers. Imports of the required classifiers are made from the sklearn library, the Decision tree is imported from the tree module in Scikit-learn. The KnearestNeighbors algorithm is imported from the neighbors module. The SVM algorithm is imported from the SVM module.

The metrics by which the classifiers will be tested are imported from the metrics module. The metrics used are the accuracy score and the confusion matrix. For training purposes, the data was split into training and testing data. This means part of the data will be used by the classifiers to learn based on the users' data. The next stage is to use some of the data to then test just how well the classifier classifies a user's activity based on sensor data alone. The accuracy score and the confusion matrix show the performance of the

classifier. Listing 13 shows the different imports and the libraries the modules are found in. most of the modules come packaged with Scikit-learn.

```
from matplotlib import pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Listing 13 Imports for Classifiers

An important import is from matplotlib, this module allows graphs to be designed using the data and can give some insight on the dataset. For the splitting of the dataset, a method from the cross-validation module in python is imported known as train_test_split. This method takes in the features and the targets/classes, the ratio is then set and is at the discretion of the researcher. For the research different ratios are tested and the performance of the classifiers is noted.

The Pandas library is used to read the dataset into the python file and assigned to a variable. This variable is used to interact with the dataset. A list of lists created that contains the features, the features are the different sensor values that were collected. A list of the class labels is also created using the dataset and these are the target variables. From these variables are created that will contain the features for testing and those for testing. Variables that contain the class labels for training and those for testing are also created. A model was created that will be used for the learning algorithm. The training features and their training target variables are fit to the model. Once fitting the data is complete then the model can be tested using the features for testing. The model has not given the target variables in this case and so has to give its own predictions of the activities.

```
model = DecisionTreeClassifier()

fittedModel = model.fit(featureTrain, targetTrain)

predictions = fittedModel.predict(featureTest)

print predictions
```

Listing 14 Decision Tree model fitting

Listing 14 shows a code snippet of a model for the decision tree. This is a basic decision tree with standard parameters from Scikit-learn. The Decision tree can be edited to add more parameters to give better results. The method fit() takes the training features and the training target variables and is used for the model to learn from these. The predict() model takes only the testing features and from what the model has learnt it gives its predictions. The predictions are printed out to get an idea of the different predictions of activities the model has made. The predictions are tested against the target variables that relate to the testing features. Using this comparison, the confusion matrix can be shown and the accuracy score can also be calculated as shown in Listing 15.

```
print confusion_matrix(targetTest, predictions)
print accuracy_score(targetTest, predictions)
```

Listing 15 Scores used for testing

This is done for all the classifiers and gives the different metrics for each. The classifiers are trained on changing ratios of testing data and the results are collected. The classifiers are then given parameters that allow the classifiers to be optimized. The first stage of optimizing was to change the parameters of the classifiers. The different parameters were tested such as the depth of the decision tree, the number of neighbors that are used by the KnearestNeighbors and the kernel used for SVM. The next stage was to attempt to use ensemble techniques on the classifiers. One such ensemble technique is the bagging classifier which takes the classifier being used as a parameter.

4.7 Pattern Analysis

Pattern analysis is the final stage of the system where the behaviors associated with the context information are found. This stage involves the creation of an algorithm that allows the different patterns to be observed. The researcher used clustering to attempt to find these behaviors using the data found from classification. The algorithms used in sklearn do not accept string values as labels. This was discovered after testing the dataset with

k-means and Mean Shift. The decision was taken to convert the labels to integers representing each of the states of the activities:

1. Sitting
2. Standing
3. Walking
4. Jumping
5. Running

The list above gives the activity and the integer value it was associated with. This means after the predictions have been done by the classifier then the integer value is the one stored in the database for use in classification. The activities are passed to the clustering algorithms in the form of a list of the labels. The clustering algorithms were imported from the Scikit-learn Cluster module in python. To extract the values from the file with the data the `genfromtext()` method in numpy was used creating a numpy array to be used as the dataset.

```
ms = KMeans(n_clusters=5)
ms.fit(X)

centroids = ms.cluster_centers_
labels = ms.labels_

print(centroids)
print(labels)

colors = ['r.', 'g.', 'b.', 'c.', 'k.', 'y.', 'm.']

for i in range(len(X)):
    print("coordinates:", X[i], "labels", labels[i])
    plt.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)

plt.scatter(centroids[:,0], centroids[:, 1], marker = "x", s=150, linewidths = 5, zorder = 10)

plt.show()
```

Listing 16 code snippet of clustering algorithm

Listing 16 shows the fitting of the data to the model using KMeans. The centroids are used to cluster the data. The colours used are used to uniquely identify the different clusters that are found by the algorithm. The algorithm then plots each value in the dataset in its appropriate position with the colour of its cluster. Each of the centroids is plotted onto the scatter graph and is represented by a large 'X'.

Figure 4-13 shows the graph that is shown by the clustering algorithms. The graph did not show much information for the purpose of the research. The results are difficult to analyse as this clustering graph as the plots are aligning between the devices and the activities.

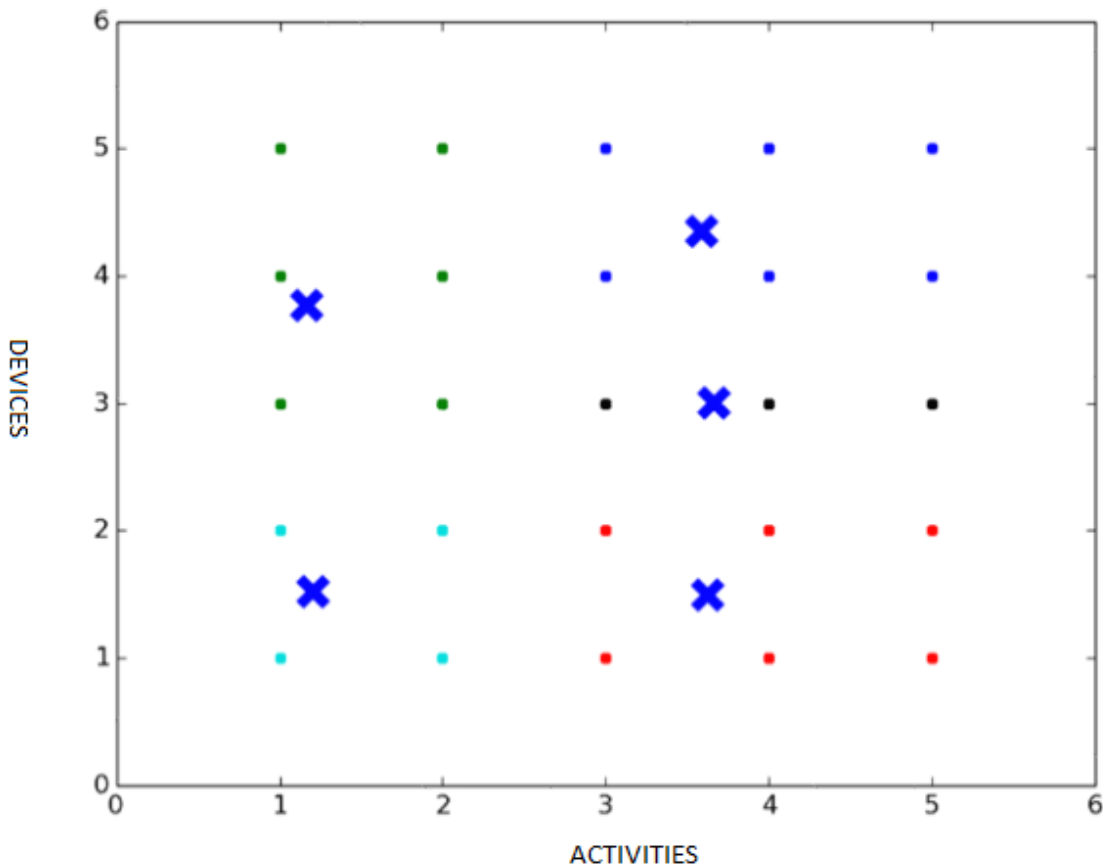


Figure 4-13 Table Created by Clustering Algorithms

By analysing the data, the researcher found that the reason for this was that the variety of the labels was not extensive such that the values could be easily scattered. The values

were either a 1 to 5 and only whole numbers were used so decimal points were not used and so only 5 usable values were available to the clustering algorithms.

The classifier can then be used in the case that the system intends to find which behaviors exactly are being performed by the individuals. The pattern analysis can be used to find which behaviors are associated with the context data. This means that the pattern analysis can be done specifically on the context data. Tests done on the classifier discussed in the next chapter were used to select the context data that is most relevant for use during pattern analysis. Another pattern analysis algorithm was then used using the accelerometer values as they were the ones mostly affected by the changes in an individual's activities. The algorithm looks for different patterns in the dataset and checks within the dataset if any other patterns like this can be found. Using this information, it then predicts the direction the pattern may follow. The algorithm makes use of percentage change between the different data points. For each of the pattern the algorithm checks how similar each of the points is to each other then takes the average of the points.

```
for eachPattern in patternAr:
    sim1 = 100.00 - abs (percentagechange (eachPattern[0], patForRec[0]))
    sim2 = 100.00 - abs (percentagechange (eachPattern[1], patForRec[1]))
    sim3 = 100.00 - abs (percentagechange (eachPattern[2], patForRec[2]))
    sim4 = 100.00 - abs (percentagechange (eachPattern[3], patForRec[3]))
    sim5 = 100.00 - abs (percentagechange (eachPattern[4], patForRec[4]))
    sim6 = 100.00 - abs (percentagechange (eachPattern[5], patForRec[5]))
    sim7 = 100.00 - abs (percentagechange (eachPattern[6], patForRec[6]))
    sim8 = 100.00 - abs (percentagechange (eachPattern[7], patForRec[7]))
    sim9 = 100.00 - abs (percentagechange (eachPattern[8], patForRec[8]))
    sim10 = 100.00 - abs (percentagechange (eachPattern[9], patForRec[9]))
```

Listing 17 Code snippet of calculation of similar patterns

Listing 17 gives 10 points that are being compared for similarity with the other pattern that needs to be recognized. Patterns that are similar to the chosen pattern are then plotted with the pattern in one graph. The predicted point is then also plotted to show where the algorithm is expected to go. The plot also shows where the actual plot went. If the difference is far greater than expected, then that change in expected with actual values may be regarded as an emergent behaviour.

Chapter 5: TESTING AND EVALUATION

This chapter presents the tests that were performed on the different components of the systems described in chapter four. The chapter continues to evaluate the results of the different tests on the system. The chapter is sectioned as follows, 5.1 describes the tests that were done on the classification algorithms and the pattern analysis algorithm, 5.2 presents the results associated with each of the tests on the algorithms, 5.3 discusses and analyses the results of the experiments and 5.4 concludes the chapter. The test produces a series of graphs and tables that are presented in this chapter and can be analysed to give the overall performance of the classifiers and pattern analysis on the dataset. The performance of the algorithms is affected by the sample size, the model complexity and the overall features chosen for the algorithm's features set.

5.1 Classifier Testing

A number of experiments were performed on the algorithms used in the research. The first set of experiments was on the classifiers used for classifying the activities being performed. For each of the chosen classifiers, Decision tree, KnearestNeighbors and Support Vector Machines an experiment on how the model's complexity would affect their accuracy in classification. The model complexity involved changing the parameters of the base classifier to find the optimum parameters that would give the best classification. The

next set of experiments was to find how the sample sizes used for training would also affect the classification accuracy. The last stage was to analyse which features from the data set gave the most information about the activity from the sensors chosen. The scores used for the experiments are the accuracy score and the confusion matrix. The results of each of the experiments were represented using graphs and tables that give a comparative view of the different performances of the algorithms.

5.1.1 Classifier model complexity

The focus of the first experiment was to observe how the complexity of the model from sk-learn can be modified using different parameters and how this affects the predictive power of the classifier. The goal of the test was to identify a single classifier from the three selected that has the best accuracy score after being modified. The classifiers are used with their base parameters at first and the data set is halved to give the training and test sets. The same dataset is used for all the classifiers in the same proportions. With the dataset and the classifier all kept standard, the data set with 1090 instances and split into half for training and testing the accuracy score function of sk-learn was used as the score. Table 2 below shows the results of the experiments.

Table 2: model accuracy

Model	Accuracy
Decision Tree	52.293%
KnearestNeighbors	71.192%
Support Vector Machine	61.467%

The table shows the accuracy as a percentage of the number of times the model was able to correctly classify the given test instance. The accuracy scores shown in the table

are not sufficient for the classification of the activities in this study. The next stage of tests used the same parameters for the classification but the classifiers used cross-validation for training and testing. Cross-validation is a prediction model validation technique. The cross-validation was used in the data set where 70% of the dataset was used for training the classifier then the remaining 30% was used to test the classifier. This was used to limit problems such as overfitting and allows for insight on how the classifier would perform with independent data. The data set was kept consistent with all the classifier and all other parameters were kept standard. The results of this experiment are shown in Figure 5-1 which shows the confusion matrix of each of the classifiers.

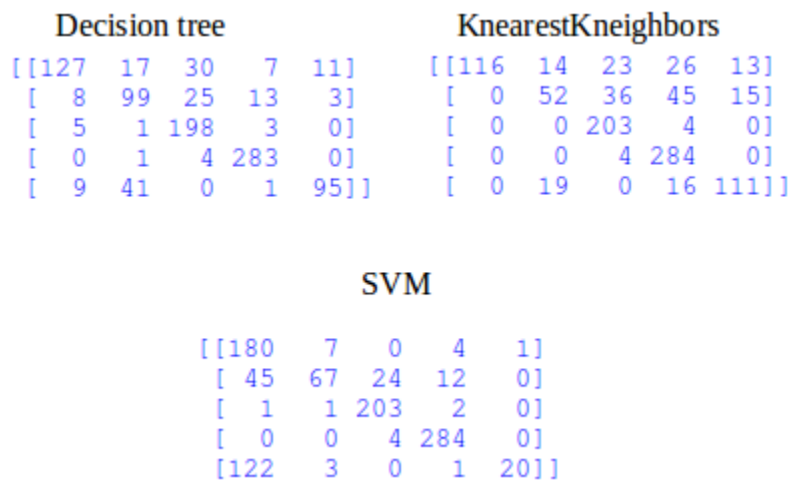


Figure 5-1 confusion matrix of each of the classifiers.

The confusion matrix shows the instances that were correctly classified and those that were misclassified. Table 3 shows the accuracy score of each of the classifiers using the cross-validation technique

Table 3: Accuracy scores of models with cross validation

Model	Accuracy
Decision Tree	81.753%
KnearestKneighbors	78.083%

Support Vector Machine	76.860%
------------------------	---------

The models were modified again by adding parameters to the classifiers that best suited the datasets. Each of the classifiers has different parameters. The Decision Tree uses the depth of the tree and this can be increased to a point where overfitting becomes a problem. The KnearestNeighbors algorithm uses the number of neighbours that it uses for the classification of an instance. The SVM uses the different kernels that are available to the classifier. Listing 18 shows the parameters used for each of the classifiers during this experiment. The decision tree used a depth of 9, KnearestNeighbors use the number of neighbours of 3 and SVM used the parameter gamma as 0.01 and c. C defines the hyper-plane margin used. A high c will tell the SVM to use a smaller hyper-plane margin while a smaller one uses a smaller hyper-plane margin.

Table 4: accuracy of models using different parameters for each classifier

Model	Accuracy
Decision Tree	92.354
KnearestNeighbors	90.825
Support Vector Machine	91.437

5.1.2 Classifier Sample size

The data set is composed of 1090 instances; in the first test for model complexity the first test used 50% of the data for training and 50% for testing. The tests that followed used 70% for training and 30% for testing. In this set of experiments, the goal is to observe the effects of the size of the training sample relative to the test set on the prediction power of the different classifiers. The classifiers will give the predictions of the activities from the test set using 70% of the dataset and have a decreasing size of the training set. The

intervals used for this experiment range from 10% to 70% in 10% intervals. The major reason behind using 10% is most major changes in the accuracy of the learning algorithms is dependent on the size of the training set. The algorithm gave low accuracy score below 10%. At 70% and above the scores had little to no variance. Therefore we chose to use this range for the experiments. The use of different samples can be used to find a reduced training sample that can still give good test results and this can reduce the time required for the training stage of the model. The results of this experiment are shown in Table 5 and Figure 5-2 shows a graphical comparison of the different classifiers.

Table 5: changing training set size

	Decision Tree	KnearestNeighbors	SVM
10%	81.753	78.084	76.860
20%	89.106	83.830	78.670
30%	89.253	83.879	84.665
40%	88.685	85.474	84.404
50%	90.092	88.073	87.156
60%	90.596	88.532	87.156
70%	91.743	88.685	86.850

This set of tests the test samples were the only varying parameters. The model complexity was kept consistent in that each of the classifiers was kept with their optimum characteristics.

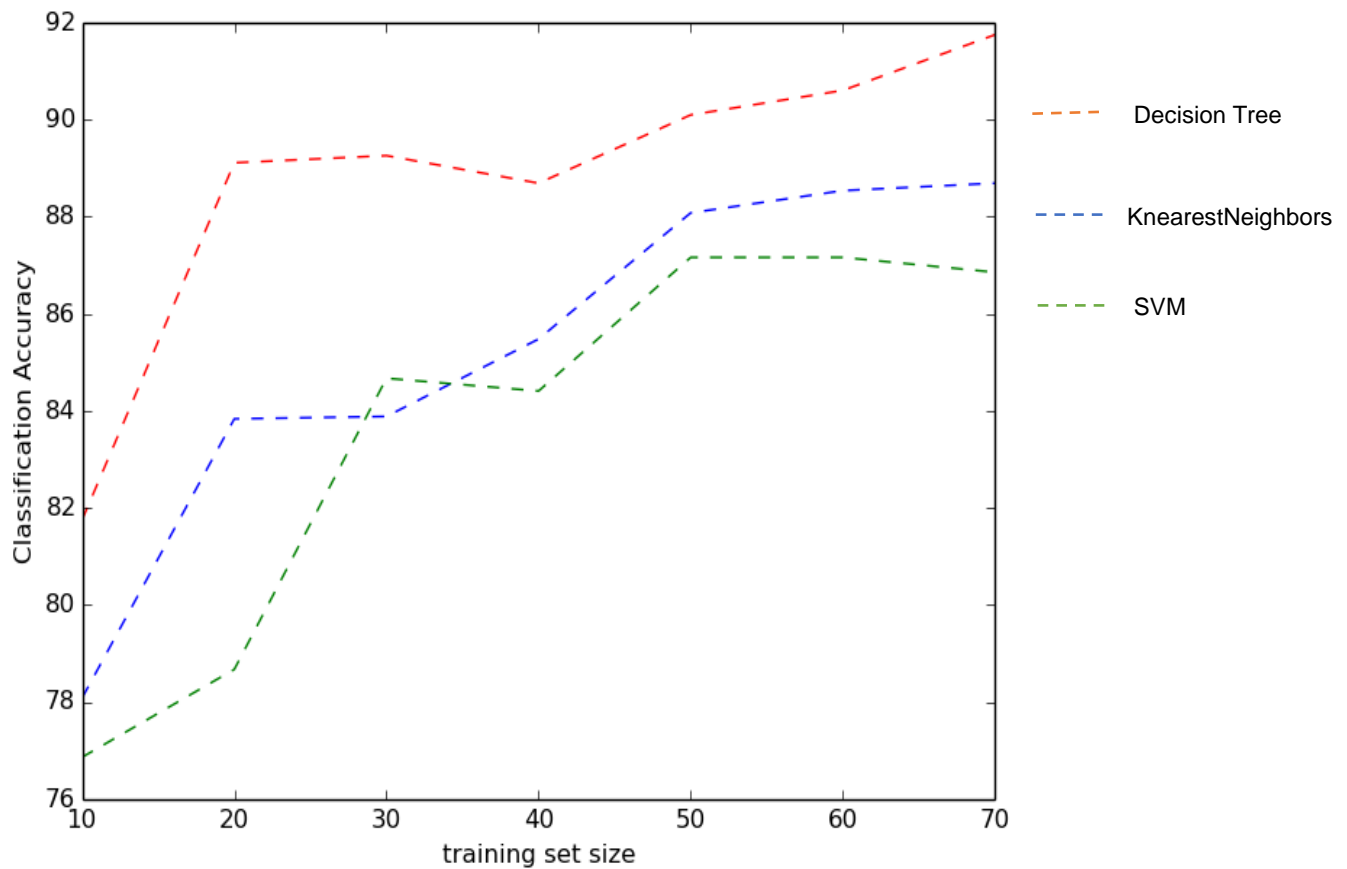


Figure 5-2 Graph of performance of classifiers

5.1.3 Classifier Dimensionality of features

All features within the dataset were used in all the previous tests. These allowed for the classification of the activities. The goal of the next set of tests was to find how the classifier performs with certain features. Some of the features may not be relevant to the activity being performed. The tests involved using different features and how well the classifier predicted the activity. The model complexity and the training are kept consistent for the first set of tests then the test size is varied at a later stage. The training set was at 70% of the dataset and the test set was the remaining 30%. The results of the test using accelerometer-based features only are shown below followed by that of using the light and proximity-based features. Table 6, 7 and 8 show how the dimensionality reduction affects the accuracy of the classification.

Table 6: Accuracy scores using accelerometer only compared to all features

	Accuracy (accelerometer)	Accuracy (all features)
Decision Tree	87.155%	92.354%
Knearesneighbors	89.602%	90.825%
SVM	88.685%	91.437%

Table 7: Accuracy scores using light and proximity only compared to all features

	Accuracy(light and proximity)	Accuracy (all features)
Decision Tree	72.171%	92.354%
Knearesneighbors	71.559%	90.825%
SVM	74.617%	91.437%

Table 8: Accuracy scores using accelerometer X and Y axis only compared to all axis

	Accuracy (X-axis and Y-axis)	Accuracy (all features)
Decision Tree	86.850%	87.155%
Knearesneighbors	85.626%	89.602%
SVM	83.792%	88.685%

The features selected for classification can have different effects on the accuracy of the classifier. Table 6 shows how the accuracy of the classifier decreases as the light and proximity are not used in classification in comparison to using all the features. Table 7

shows when we use only light and the proximity and the results are lower than those of using the accelerometer. The main reason for this is that the most affected sensor by physical activities is the accelerometer. Table 8 then goes to observe if using just two of the axis of the accelerometer would give much difference assuming X and Y are the most affected by the forces of gravity during movement. Dimensionality reduction can be separated into two forms, feature extraction and selection (Teng et al. 2014). Feature selection is used here by removing the features that are irrelevant. This can be seen in Table 6 as the results are not greatly affected by the removal of some of the features. Knowledge of the domain was used for the feature selection as the researcher noticed from the results how the accelerometer was the sensor most affected by physical activities.

5.2 Pattern Analysis Testing

The algorithm was tested with a dataset that had some variations in it to show a change in behavior over a short period. The dataset represents the values sent to the server from different devices. The X, Y and Z axis were used for the purpose of classification. These features were chosen based on the results collected from the dimensionality tests done on the data sets.

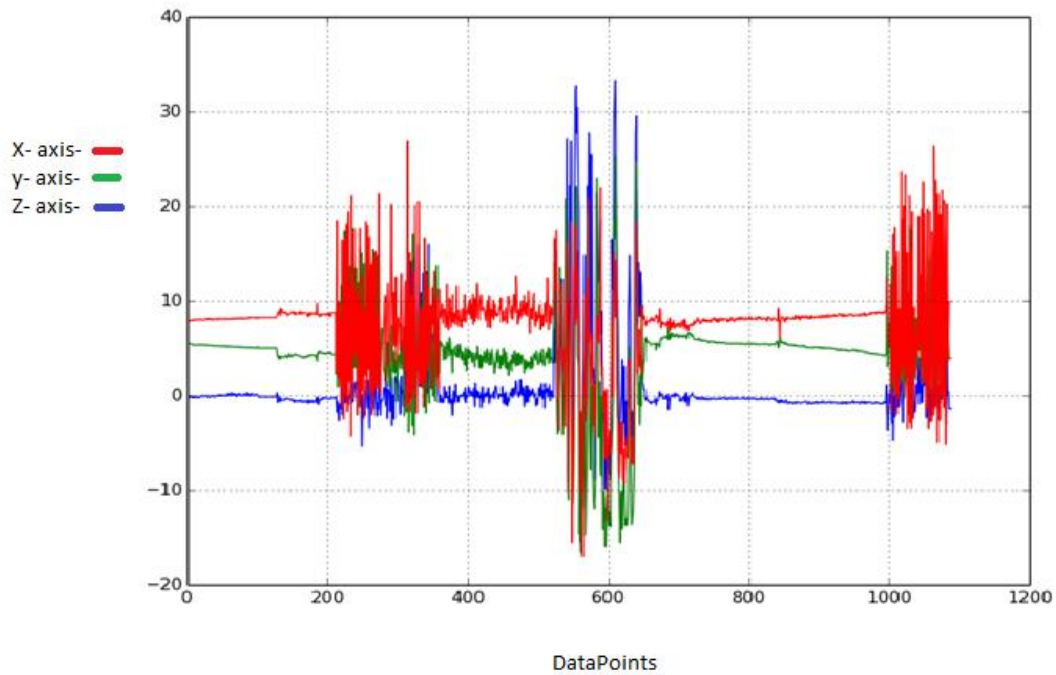


Figure 0-3 Graph of all points against the X, Y and Z axis of the accelerometer

The algorithm can then make use of the dataset to find the different patterns that may be similar and give its prediction on how they would have turned out. Individuals may carry out different activities and so a lower accuracy score was chosen. An accuracy score of 60% was used for this purpose.

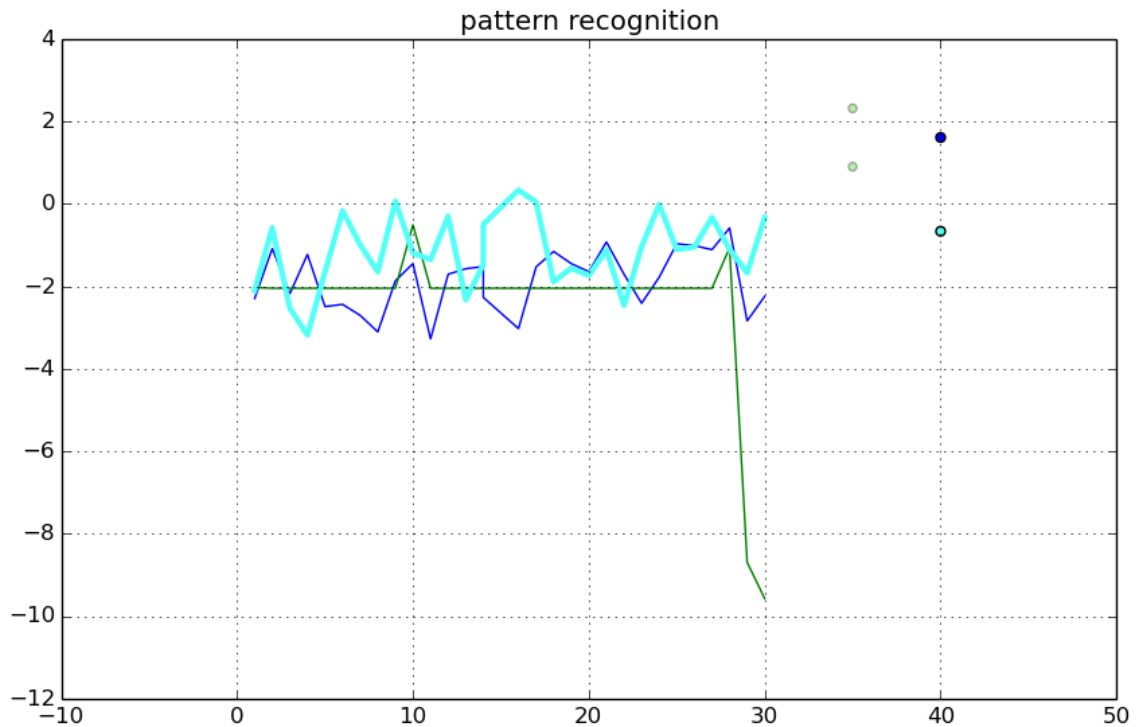


Figure 0-4 patterns in data set and the similar patterns

The algorithm finds the average of the X, Y and Z axis and uses this to give the one line in the graph. The cyan line represents the pattern being looked at, the other lines are similar patterns seen in the past within a similar threshold. The cyan dot represents where the actual pattern went while the other dots represent the predictions of the algorithm from the lines in the past. The dark blue dot shows the average of the predictions. In some cases, more lines were similar to each other and this gave a figure as follows:

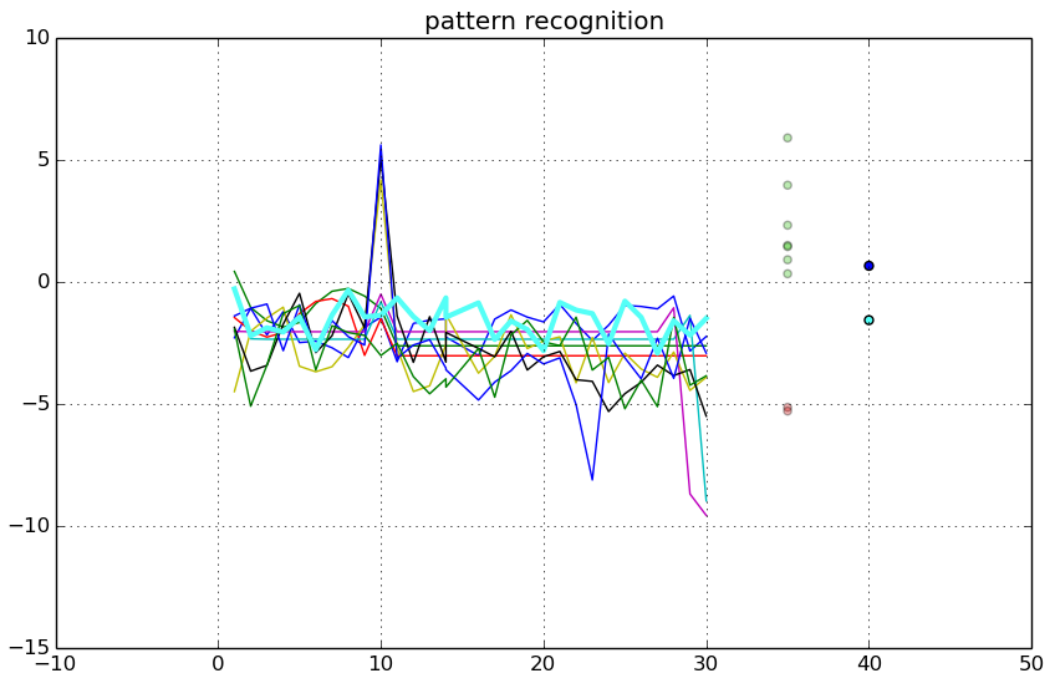


Figure 0-5 multiple similar patterns

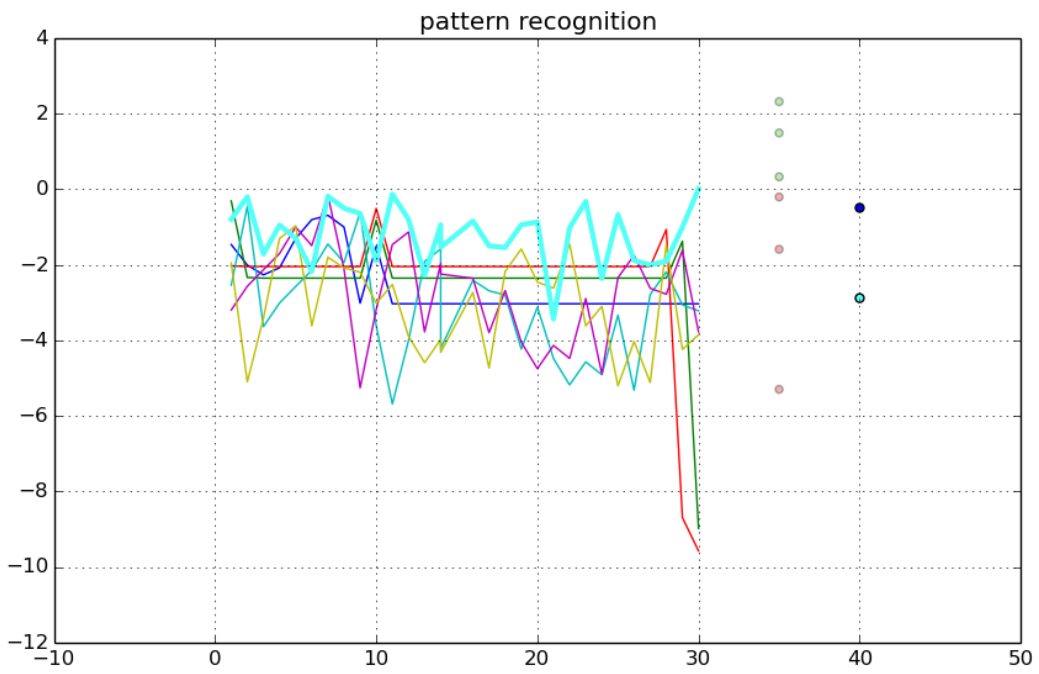


Figure 0-6 multiple predictions showing where the pattern could go

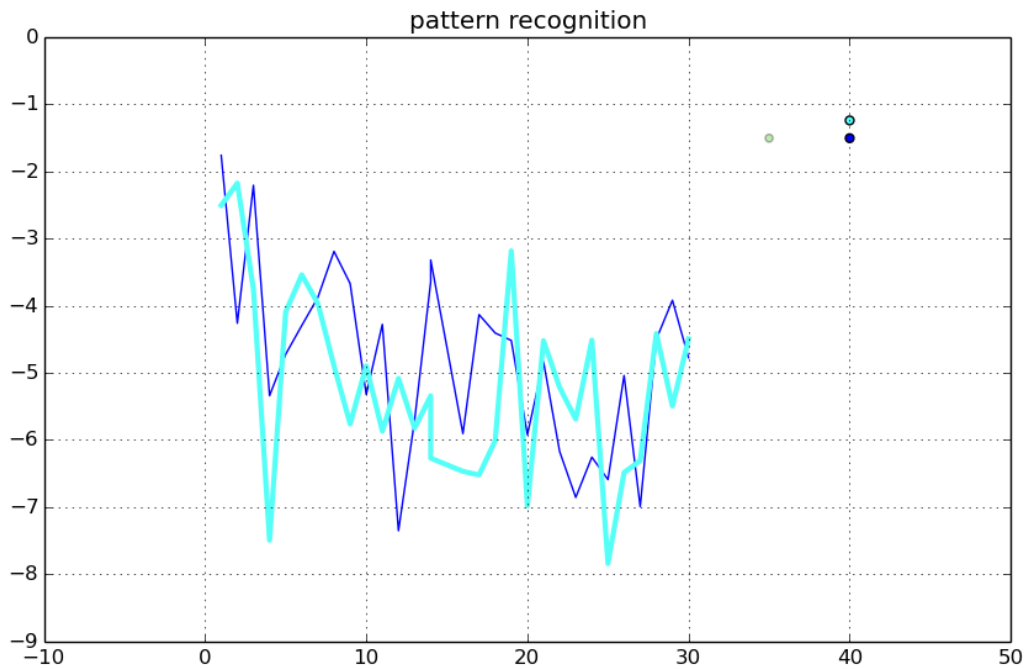


Figure 0-7 pattern recognition using 70% accuracy

Figure 5-7 shows the accuracy increased to 70% and this yields very few lines because of the variations in how individuals may do certain activities.

5.3 Discussion

This section discusses the results that have been presented in the previous sections of the chapter. The results given in this chapter for the classifiers were due to the effects of the changes applied to the different parameters. Each parameter was tested while the other parameters were unchanged in order to observe its effects in a controlled environment. We observed some results of the classifier with the model complexity, sample size and features used were systematically changed. In Figure 5.2 the results for tests done on the pattern analysis algorithm are also presented. These show the different patterns that can be found in the dataset. This section discusses the results and how they can be used to show any emergent behaviours.

5.3.1 Model Complexity

The discussion begins by explaining the changes in the performances of the classifiers due to the model complexity. Each of the algorithms begins to perform better as the complexity of the model increases. The first change was the use of cross-validation and this improved the classifier performances as a whole as shown in Table 3. Using cross-validation brings an element of ambiguity, the classifier will give different scores with changes in the dataset used in training and the average of these scores gives the better of its scores across the entire dataset (Krogh & Vedelsby 1995). The generalisation error and all other factors are kept constant. The next stage was to change the parameters of the classifiers. With decision trees, the depth of the tree can increase or compromise the performance. The depth can keep increasing but the risk of overfitting becomes eminent. In knearestneighbors algorithms the number of neighbours affects the classification ability. The algorithm uses the closest related neighbour to the data point but may in some cases need to use a majority vote to select the most appropriate class. Too many neighbours may make the voting decision difficult. In SVM the kernel used may be best suited for the dataset that may be in question.

5.3.2 Sample Size

This section deals with the analysis of how the training set size affects the classification of the given classifiers in finding the activity of the user. Table 5 shows how the accuracy scores of the algorithms change the size of the training set also increases. In Figure 5-2 it is visualized and can be observed that the classification accuracy increases as the number of training instances increases. Figure 5-2 shows as the classification accuracy increases as the training set size increases from 10% to 70%. The decision tree generally shows the best results in its performance as the training set increases. The larger training sets sizes decrease the bias for particular features within the data set. The SVM does not show much improvement as it mainly uses the two points on each side of the hyper-plane and hence the sample size does not directly affect it. The knearestneighbors algorithm is affected by the training set size because k nearest data points to the test instance and

uses the most common labels. The more labels it can collect for the test set the better the algorithm can be.

5.3.3 Pattern Analysis

This section discusses the results given in Figure 5.2 for the pattern analysis. The aims of the research are to find the emergent behaviors from users' context data. As mentioned in the literature review pattern analysis involves two major tasks. The first is to find the characteristics of the pattern being observed. The results show how the system can be used to find patterns that are similar in the past from the context data. The algorithm uses the characteristics of the pattern to find these similar patterns. The algorithm gives the predicted outcomes in the pattern at the next data point as shown in Figures 5-5, 5-6 and 5-7. Each prediction is based on some pattern that the algorithm has observed in the data before the current pattern being observed.

In the case that algorithm's prediction and the actual outcome are very different such as in the lowest positioned predictions in Figure 5-5 and Figure 5-6 the actual outcome can be regarded as a new behaviour by a certain user or group of users. If the predicted outcomes are close to the actual point where the actual pattern went then we can assume that there is conformity in the behaviors of the users. We can take this into perspective and say if patterns for a group are almost 60% similar for five days then if a new pattern is observed on the sixth day then this can be regarded as a new behaviour. If the patterns are similar, then it can be said there is an observation of conformity. The second major task of pattern analysis is the classification of the pattern being observed. The activities being performed can be found by using the classification algorithm.

5.4 Conclusion

The chapter outlined the classification performance of the Decision Tree, KnearestNeighbors and SVM. Each of the experiments was discussed and the results

associated with the experiments also shown. The experiments done on the pattern analysis algorithm are also given in this chapter. The chapter concludes by discussing the results from the experiments and explaining the different representations given. The closing chapter of the dissertation gives the findings of the research with reference to the objectives given in the introduction chapter.

Chapter 6: CONCLUSION AND FUTURE WORK

The purpose of this research has been to investigate if context information can be aggregated and from this if emergent behaviour may be found. The experiments performed in the previous chapter and the discussion aimed at showing that is indeed possible. This chapter concludes the research work by showing how each of the findings in the research map to the research questions introduced in chapter 1. This chapter concludes by giving improvements and key interest areas that can be followed for future studies.

6.1 Mapping of Findings to Research Questions

1. What context-aware computing models currently available?

The research introduced context-aware computing in the literature review (chapter 2). The different context-awareness models are discussed in this chapter including the frameworks. Context-aware computing has been a major move in technology and software development. It has gained even more ground with the improvement of devices especially mobile devices. Models have been developed to better utilize the context information, such as the use of widgets. Frameworks have also been developed to allow more software to be developed that is context-aware. The problem still remains that little of this context is used in its aggregated form.

2. What context can be found using sensors on a device. Can an application using this context be developed?

The growth in the functionalities of mobile devices has allowed for different sensors to be embedded in them. This is discussed in the literature review (chapter 2) as context-aware computing is introduced. The research investigates in detail the different sensors that can be found in the mobile devices, specifically on the Android platform. The research makes use of the Android platform to develop an application for collecting sensor data; this is documented in the design and Implementation (chapter 4). The mobile application is developed to use the following sensors:

- Accelerometer
- Proximity Sensor
- Light Sensor

These above-mentioned sensors are used as context information that will be used to classify the activity being performed by the user. The application is also used to collect training data for the classifier.

3. Can the context information from different devices be collected and aggregated using a central service?

The implementation of the application for collecting sensor information allows a device's sensor information to be collected. The next stage was the implementation of a server to aggregate the context information from different devices. The details of the implementation of the server are documented in the design and implementation (chapter 4). The server allows devices to connect to it and send their context information. The context information is then aggregated into one database. The server and the database base function for the purpose of context aggregation as a central service for all connected devices.

4. How can emergent behaviours and patterns be mapped to certain context?

The context information that was collected and stored in the database was used as a dataset to analyse the observable patterns within the dataset. Section 5.2 in Testing and Evaluation (chapter 5) shows the patterns that were found in the dataset. These patterns were used to attempt to find different behaviours within the dataset. The experiments performed show that in certain cases some of the predicted outcomes were not far off from the expected results and hence this can be regarded as a common behaviour among the different devices used in the research. The ability of the pattern analysis algorithm to correctly identify the patterns predicted outcome from the past shows that either the same behaviour was exhibited by the same devices in the future or that the behaviour is common among many devices. Any changes in this behaviour can be observed as a large difference in the outcome and the predicted outcome. The classification of activities can

then be used to find the activities being done at that point thus mapping the patterns to the activities.

6.2 Recommendations and Future Work

The scope of the research was limited to the set of research questions and objectives mentioned in chapter 1. Areas of possible research include exploring the effects of secondary sensors, such as step counter and step sensor on the classification of different activities. Another possible research area includes the use of complex profiles to deduce observations that are not evidently observable such as emotional states of users. This can be done using techniques like sentiment analysis. Furthermore, the study into application areas of this research may be useful as this allows the observation of abnormal behaviours among different individuals which could mean a number of things. Such as:

- Early disaster identification from the movement of users.
- Use in clinics or hospitals to observe patient behaviour.
- User status monitoring in other applications to better their services.
- Personal mobility information.

The application and the system are intended to be further developed addressing major concerns such as:

- Data Usage
 - The periodic probing of sensor data by the server may solve this issue by setting up intervals for sending sensor values instead of continuous probing. This would mean the device and server only need to connect at particular intervals.
- Privacy Issues
 - This refers to the fact that most users are not inclined to having their information used without a level of anonymity. From applications that have

been developed by other developers, it may be possible to achieve this anonymity

- Battery Life
 - The periodic probing of sensor data by the server may solve this issue by setting up intervals for sending sensor values instead of continuous probing.

6.3 Limitations

For the research to simulate the server aggregating the context of multiple devices two devices were used and tested multiple times. Each instance of the tests was taken as a new device. This was due to a lack of devices and users to run the application over long durations of time.

6.4 Conclusion

The aim of this research was to aggregate the context from different devices and observe the patterns and behaviours from this context. Firstly, the research embarked on finding if the activities being performed by users could be classified using machine learning algorithms. The aggregated context was then analysed and some patterns were found within the data. This context was used to find conformity in the behaviour of the users or lack of it using the prediction abilities of the pattern analysis algorithm. The research can then give a greater understanding of the behaviours or patterns that are common in certain environments.

REFERENCES

Abbas, A., Zhang, L. & Khan, S.U., 2015. A Survey on Context-aware Recommender Systems Based on Computational Intelligence Techniques. *Computing*, pp.667–690. Available at: <http://link.springer.com/article/10.1007/s00607-015-0448-7>.

Abraham, A., Guo, H. & Liu, H., 2006. Swarm intelligence: Foundations, perspectives and applications. *Studies in Computational Intelligence*, 26, pp.3–25.

Almuallim, H., Kendeda, S. & Akiba, Y., 2002. Development and Applications of Decision Trees. In *Expert Systems The Technology of Knowledge Management and Decision Making for the 21st Century Six-Volume Set*, 1, pp.53–77.

Arlot, S. & Celisse, A., 2010. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4, pp.40–79. Available at: <http://eprints.pascal-network.org/archive/00006812/>.

Ayodele, T., 2010. Types of machine learning algorithms. ... */Show/Title/Types-of-Machinelearning-Algorithms*, pp.19–49. Available at: <http://cdn.intechweb.org/pdfs/10694.pdf>.

Baer, D.M., Wolf, M.M. & Risley, T.R., 1968. Some current dimensions of applied behavior analysis. *Journal of applied behavior analysis*, 1(1), pp.91–97.

Baldauf, M., Dustdar, S. & Rosenberg, F., 2007. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4), p.263.

Beach, A. et al., 2010. Fusing Mobile, Sensor, and Social Data To Fully Enable Context-Aware Computing. *Workshop on Mobile Computing Systems & Applications*, pp.60–65. Available at: <http://dl.acm.org/citation.cfm?id=1734583.1734599>.

Ben-david, S. & Shalev-Shwartz, S., 2014. *Understanding-Machine-Learning-Theory-Algorithms*, Available at: <http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf>.

Berkhin, P., 2002. Clustering survey Bherkin. *Technical Report, Accrue Software*, pp.1–56.

Bonabeau, E., Dorigo, M. & Theraulaz, G., 1999. Swarm intelligence. , pp.1–11. Available at: <http://www.bydbest.com/ebooks/E-books/Harvard Business Review/Harvard Business Review/2001/may 2001/swarm Intelligence.pdf>.

Chuang, L.-Y. et al., 2009. Chaotic Genetic Algorithm for Gene Selection and Classification Problems. *Omics : a journal of integrative biology*. pp 407-420 Available at: <http://www.ncbi.nlm.nih.gov/pubmed/19594377>.

Cortes, C. & Vapnik, V., 1995. Support-Vector Networks. *Machine Learning*, 20(3), pp.273–297.

Costa, A., Guizzardi, R.S.S. & Guizzardi, G., 2007. COReS: Context-aware, Ontology-based Recommender system for Service recommendation. pp 2-15 Available at: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:COReS+:+Context-aware+,+Ontology-based+Recommender+system+for+Service+recommendation#0>.

Davidson, R.M., 2004. Chapter Three : Research Methodology. *City University of Hong Kong,lecture*, pp.1–20. Available at: <http://www.is.cityu.edu.hk/staff/isrobert/phd/ch3.pdf>.

Dey, A.K., Abowd, G.D. & Salber, D., 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, 16(2), pp.97–166.

Duda, R.O., Hart, P.E. & Stork, D.G., 2001. Pattern Classification. *New York: John Wiley, Section*, p.680.

Dutt, V., Chaudhry, V. & Khan, I., 2012. Pattern Recognition: an Overview. *American Journal of Intelligent Systems*, 6(6), pp.57–61.

Eberhart, R.C., Shi, Y. & Kennedy, J., 2001a. Swarm intelligence. , pp.1–31.

Eberhart, R.C., Shi, Y. & Kennedy, J., 2001b. Swarm intelligence. , (March).

Erik G., 2014. Introduction to Supervised Learning. , pp.1–5. Available at: <http://people.cs.umass.edu/~elm/Teaching/Docs/supervised2014a.pdf>.

Fletcher, T., 2009. Support Vector Machines Explained. *Online*. <http://sutikno.blog.undip.ac.id/files/2011/11/SVM-Explained.pdf>. [Accessed 06 06 2013], pp.1–19. Available at: <http://sutikno.blog.undip.ac.id/files/2011/11/SVM-Explained.pdf>.

Gopalakrishnan, K. et al., 2012. A Fresh Graduate's Guide to Software Development Tools and Technologies. pp 5-9 Available at: http://www.academia.edu/28761386/A_Fresh_Graduates_Guide_to_Software_Development_Tools_and_Technologies_Chapter_Mobile_Platform_CHAPTER_AUTHORS

Gsma Intelligence, 2014. Mobile platform wars. *Insurance & Technology*, 35(February), pp.1–26. Available at: http://cm7ly9cu9w.search.serialssolutions.com/?ctx_ver=Z39.88-2004&ctx_enc=info:ofi/enc:UTF-8&rft_id=info:sid/ProQ&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.genre=article&rft.jtitle=Insurance+%26+Technology&rft.atitle=The+Mobile+Platform+Wars&rft.au=.

Hong, J., Suh, E. & Kim, S.-J., 2009. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4), pp.8509–8522. Available at: <http://dx.doi.org/10.1016/j.eswa.2008.10.071>.

Jain, A.K., 2010. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8), pp.651–666.

Joachims, T., 2002. Introduction to Support Vector Machines. *Distribution*, pp.1–15. Available at: http://videlectures.net/epsrws08_campbell_isvm/.

Karypis, G. & Han, E.S., 1999. C HAMELEON : A Hierarchical Clustering Algorithm Using Dynamic Modeling. *COMPUTER*. 32(8), pp 68 - 75

- Kotsiantis, S.B., Zaharakis, I.D. & Pintelas, P.E., 2006. Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review*, 26(3), pp.159–190.
- Kpalma, K. & Ronsin, J., 2007. An Overview of Advances of Pattern Recognition Systems in Computer Vision. *Vision Systems: Segmentation and Pattern Recognition*, (June), pp.169–194. Available at: [\url{http://hal.archives-ouvertes.fr/hal-00143842/en/}](http://hal.archives-ouvertes.fr/hal-00143842/en/).
- Krogh, A. & Vedelsby, J., 1995. Neural Network Ensembles, Cross Validation, and Active Learning. *Advances in Neural Information Processing Systems 7*, pp.231–238.
- Lettner, M., Tschernuth, M. & Mayrhofer, R., 2012. Mobile platform architecture review: Android, iPhone, Qt. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp.544–551.
- Lim, T.S., Loh, W.Y. & Shih, Y.S., 2000. Comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3), pp.203–228.
- Mäntyjärvi, J., 2003. Sensor-based context recognition for mobile applications. *Vtt Publications*.
- Mining, D. & Discovery, K., 1997. BIRCH: A New Data Clustering Algorithm and Its Applications. , 182, pp.141–182.
- Mitchell, T.M., 1997. *Machine Learning*, Available at: <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0070428077>.
- Nadeem, T., 2012. App Development for Smart Devices Lec # 5: Android Sensors Objective Working in Background Sensor Manager.
- N. Nillson, N.J., 2005. *Introduction to Machine Learning*, Available at: <http://www.ncbi.nlm.nih.gov/pubmed/21172442>.
- Noh, Y., 2015. 지도학습 (Supervised Learning). , 21.

Okediran, O.O., Arulogun, O.T. & Ganiyu, R. a, 2014. Mobile Operating System and Application Development Platforms : A Survey. *Journal of Advancement in Engineering and Technology*, 1(4), pp.1–7.

de Oliveira, L.B.R. & Loureiro, A.A.F., 2011. CodeDroid: A Framework to Develop Context-Aware Applications. *CENTRIC: Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services*, (c), pp.72–77.

Quinlan, J.R., 1993. J. Ross Quinlan_C4.5_ Programs for Machine Learning.pdf. *Morgan Kaufmann*, 5(3), p.302. Available at: <http://www.springerlink.com/index/10.1007/BF00993309>.

Raghavendra, R.G.B. & Vasantha, S., 2016. Comparative Study on Mobile Platforms. *International Journal of Innovative Research in Science, Engineering and Technology*, pp.5601–5607.

Renner, T., 2011. Mobile OS-Features, Concepts and Challenges for Enterprise Environments, pp.1–9. Available at: http://www.snet.tu-berlin.de/fileadmin/fg220/courses/SS11/snet-project/mobile-os-features_renner.pdf.

Saunders, B.M. & Tosey, P., 2013. The Layers of the Research Onion. *Rapport*, 14(4), pp.58–59.

Schmidt, A., Beigl, M. & Gellersen, H.W., 1999. There is more to context than location. *Computers and Graphics (Pergamon)*, 23(6), pp.893–901.

Singh, K.P., Basant, N. & Gupta, S., 2011. Support vector machines in water quality management. *Analytica Chimica Acta*, 703(2), pp.152–162.

Tan, P.-N., Steinbach, M. & Kumar, V., 2006. Classification : Basic Concepts , Decision Trees , and. *Introduction to Data Mining*, 67(17), pp.145–205. Available at: <http://www-users.cs.umn.edu/~kumar/dmbook/index.php>.

Tang, J., Alelyani, S. & Liu, H., 2014. Feature Selection for Classification: A Review. *Data Classification: Algorithms and Applications*, pp.37–64.

Tropeano, G., 2006. Self Modifying Code. *CodeBreakers Magazine*, 1(2), pp 1-6.

Weis, M. et al., 2009. Comparison of different classification algorithms for weed detection from images based on shape parameters. *Image analysis for agricultural products and processes*, 69(2005), pp.53–64.

Weiser, M., 1991. The Computer for the Twenty-First Century. *Scientific American*, 265(3), pp.94–104. Available at: <https://ctools.umich.edu/access/content/group/01ef2b16-f564-4682-8045-14ce56611c49/Readings/Weiser91-ComputerFor21st-SciAm.pdf>.

Winograd, T., 2001. Architectures for Context. *Human-Computer Interaction*, 16(2), pp.401–419.