

A Behavioral Biometrics User Authentication Study Using Motion Data from Android Smartphones

DPS Dissertation

by

Javid Maghsoudi

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

at

School of Computer Science and Information Systems

Pace University

May 2017

ProQuest Number: 10690910

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10690910

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

We hereby certify that this dissertation, submitted by **Javid Maghsoudi** has successfully satisfied all the requirements for the degree of the Doctor of Professional Studies in Computing”



Dr. Charles Tappert
Chairperson of Dissertation Committee

6/2/17
Date



Dr. Ronald Frank
Dissertation Committee Member

6/2/17
Date



Dr. Li-Chiou Chen
Dissertation Committee Member

06/02/17
Date

Seidenberg School of Computer Science and Information Systems
Pace University

Abstract

A Behavioral Biometrics User Authentication Study Using Motion Data from Android Smartphones

by
Javid Maghsoudi

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

May 2017

This is a study of the behavioral biometric of smartphone motion to determine the potential accuracy of authenticating users on smartphone devices. The study used the application Sensor Kinetics Pro and the Weka machine-learning library to analyze accelerometer and gyroscope data. The study conducted three experiments for the research. They were conducted in spring 2015, fall 2015, and spring 2016. The final experiment in spring 2016 used six Android-based smartphones to capture data from 60 participants and each participant performed 20 trials of two motions: bringing the phone up to eye level for review, and then bringing the phone to the ear, resulting in 1200 runs. The resulting sensor datasets were used for machine learning training and testing. The study used filtering data to remove noise, and then aggregated the data and used them as inputs to the Weka Machine Learning tool. The study used several machine classification algorithms: the Multilayer Perception (MLP), k-Nearest Neighbor (k-NN), Naïve Bayes (N-B), and Support Vector Machine (SVM) machine learning classification algorithms. The study reached authentication accuracies of up to 93% thus supporting the use of behavioral motion biometrics for user authentication. Preliminary studies with smaller numbers of participants in spring 2015 and in fall 2015 also produced 90%+ authentication accuracy.

Keywords - Behavioral Biometrics, User Authentication, Accelerometer, Gyroscope, Android devices, Weka, Motion Capture

Acknowledgements

I would like to dedicate this dissertation to my loving parents, Ali Asghar Maghsoudi and Marjan Alaei, for their endless love and support.

I also dedicate this work to my loving wife, Sorour, and my loving daughter Taneen. I could not have done this with your support and understanding. I love you both very much!

I also extend the dedication of this work to my brothers Houshang, Mohammad Ali, Mohammad Ebrahim, and my sisters Fakhri and Shamsi.

I would like to thank and acknowledge my advisor, Dr. Charles Tappert, whose advice, support, and constant encouragements made this possible for me. Working with Dr. Tappert allowed me to prepare a paper jointly and for me to present the paper at the European Intelligence and Security Informatics Conference (EISIC) at Uppsala University, Sweden, in 2016. I will always cherish this wonderful academic experience and opportunity...thanks very much, Dr. Tappert!

I have been privileged to have a wonderful and lifelong friend, Mehrdad Vatani. Thanks very much Mehrdad for all your encouragements, inspirations, and your belief in me!

I would like to acknowledge and say special thanks to Dr. Ronald Frank and Dr. Li-Chiou Chen for all the great courses I have taken with them and for accepting my invitation to be the members of my dissertation defense committee.

I would like also to say special thanks and extend my gratitude to Dr. Fred Grossman, Dr. Lixin Tao and the entire faculty and support staff at Seidenberg at Pace University for making this great DPS program possible.

I would also like to acknowledge and thank my cohorts in the class of 2016 and members of Team 1. We worked through it all together. I would like to extend special thanks and gratitude to Leigh Anne Clevenger and Hugh Eng for always being there to support me.

Finally, I would like to thank the Master's Degree students who worked with me on their capstone projects in 2015 and 2016. I am truly grateful to each of them for the work they did in support of this dissertation. My special thanks to Jonathan I. Lee for the research work in fall 2015 and spring 2016 in collecting the data and for the development of the Java program, and to Christopher Carlson for his work and for introducing the Sensor Kinetic Pro App for the spring 2015 research.

Contents

Abstract	iii
List of Figures	x
List of Tables	xiii
List of Equation(s)	xiv
Names and Acronyms Equivalence	xv
Chapter 1	16
Introduction	16
1.1 Overview	16
1.2 Current Authentication Methods/Technologies on Smartphones	17
1.3 Using Passwords to Authenticate Users	18
1.4 Using Security Tokens to Authenticate Users	19
1.5 Using Biometric Technology to Authenticate Users	21
1.6 Physical Biometric Technologies	22
1.6.1 Fingerprint	23
1.6.2 Iris Recognition	25
1.6.3 Facial Recognition	26
1.6.4 Voice Verification	28
1.7 Behavior Biometric Techniques	29
1.7.1 Signature Verification	29
1.7.2 Keystroke Dynamics	30
1.7.3 Mouse Biometrics	33
1.8 Sensors: The Technology behind the Biometrics	33
1.8.1 Sensor Type: Accelerometer	34
1.8.2 Sensor Type: Gyroscope	34
1.9 Machine Learning Algorithms	35

1.9.1	Supervised Learning	36
1.9.2	Unsupervised Learning	36
1.9.3	Reinforcement Learning	36
1.9.4	List of Common Machine Learning Algorithms	36
1.9.5	Support Vector Machine (SVM).....	37
1.9.6	Naïve Bayes (N-B).....	38
1.9.7	K- Nearest Neighbors (k-NN).....	39
1.9.8	Multilayer Perceptron (MLP)	40
1.10	Available Machine Learning Tools/Classifiers	41
1.10.1	Waikato Environment for Knowledge Analysis (Weka).....	41
1.10.2	Pace University Classifier.....	42
1.11	Problem Statement	43
1.12	Study's Relevance - General.....	45
1.12.1	Study's Relevance – So What?	45
1.12.2	Study's Relevance – Use Cases	47
1.12.3	Study's Relevance – Using Hand Motions as Behavioral Biometrics.....	47
1.13	Purpose of Study	48
1.14	Scope of this Study	48
1.15	Researches Conducted for this Study	49
1.16	Chapter Conclusion.....	49
1.16	Roadmap	49
Chapter 2	51
	Review of the Literature and Related Work	51
2.1	Introduction.....	51
2.2	Security Breaches in the News	51
2.3	Traditional Technologies Used to Protect Systems and Smartphones.....	53

2.4	User ID and Password Authentications	53
2.5	Identity Token Authentications.....	56
2.6	Biometrics Authentications.....	57
2.6.1	Using Smartphones to Authenticate Users Performing Daily Activities.....	58
2.6.2	Use of Multi-Sensor Authentication to Improve Smartphone Security.....	61
2.7	Role of Machine Learning Algorithms in Biometrics Authentication.....	62
2.8	Applications of Sensor-Based Smartphones.....	63
2.9	The Focus of This Study/Research: Using Motion Data to Authenticate Users.....	64
2.10	Chapter Summary and Conclusion	64
Chapter 3.....		65
Solution Methodology		65
3.1	Introduction.....	65
3.2	Chapter Organization	65
3.3	Tools Used to Support the Methodology and Research.....	66
3.3.1	The First Tool: Sensor Kinetics Pro Mobile App.....	66
3.3.2	The Second Tool: Weka Machine Learning	72
3.4	Data Gathering Protocol – General for the Three Researches Performed.....	77
3.4.1	Data Gathering Protocol in spring 2016	79
3.4.2	Data Gathering Protocol in fall 2015.....	79
3.5	Definition of Arbitrary Motions Using a Smartphone.....	81
3.6	Details on the Motions and Data Collection Steps	81
3.7	Complete Steps of Capture and Process of the Motion Data:.....	84
3.8	Data Storage.....	84
3.8	Feature Extractions, Data Processing & Machine Learning Algorithms.....	90
3.8.1	Manual Feature Extraction.....	90
3.8.2	Automated Feature Extractions.....	91

3.8.3	Manual Feature Extraction & Automated (spring/fall 2015 Research)	93
3.8.4	Automated Data Processing	96
3.8.5	The Java Program	96
Chapter 4		97
Experiments and Results		97
4.1	Introduction	97
4.2	Chapter Organization	97
4.3	Timeline of the Study and Related Researches	99
4.4	Research Study in spring 2015	99
4.4.1	Result of the spring 2015 Research	101
4.4.2	Conclusion of the Research of spring 2015	102
4.5	Research Study in fall 2015	102
4.5.1	Data Gathering Protocol for fall 2015 Research	103
4.5.2	Test Results for fall 2015 Research	104
4.5.3	Conclusion of the fall 2015 Research	106
4.6	Research Study in spring 2016	108
4.6.1	Data Gathering Protocol for Research of spring 2016	108
4.6.2	Discarding Some Data	112
4.6.3	Data Processing Methods and Machine Learning for Research of spring 2016	112
4.7.4	Summary of the High-Level Data Processing	115
4.7.5	Machine Learning Algorithms Used in the spring 2016 Research	116
4.7.6	Separate and Additional Experiments Conducted in spring 2016	116
4.7.7	Results for Complex Method Using Different Sample Sizes	120
4.7.8	The Final Test Results for Research of spring 2016	121
4.8	False positive Rates per Individual Test Participant	125
4.9	Chapter Conclusion	125

Chapter 5.....	127
Conclusions.....	127
5.1 Research Conclusion.....	127
5.2 Quick Summary of Conclusion.....	130
5.3 Future Development.....	131
Appendix A.....	132
Execution Results of Random Sets of Five, Ten and Twenty	132
Appendix B.....	142
Java Program for Automated Feature Extraction.....	142
References.....	149

List of Figures

Figure 1.1 Different User Authentication Types	18
Figure 1.2 Soft Token – Randomly Generated Number [by Entrust Company] [49].....	20
Figure 1.3 Types of Fingerprints [52].....	23
Figure 1.4 Block Diagrams: Enrollment, Verification, and Identification [51].....	24
Figure 1.5 Process: IRIS Identification Steps [115]	25
Figure 1.6 IRIS Scanners [54]	26
Figure 1.7 Facial Recognition [82].....	27
Figure 1.8 Voice Verification [96].....	28
Figure 1.9 Signature Verification [80].....	30
Figure 1.10 Keystroke Dynamics [60].....	32
Figure 1.11 Mouse Biometrics [63].....	33
Figure 1.12 Sensor Type: Accelerometer [111].....	34
Figure 1.13 Sensor Type: Gyroscope [99].....	35
Figure 1.14 Support Vector Machine (SVM) Algorithm [101].....	37
Figure 1.15 K-Nearest Neighbor (k-NN) Algorithm [101]	40
Figure 1.16 Multilayer Perceptron (MLP) Diagram/Algorithm [106].....	41
Figure 1.17 Waikato Environment for Knowledge Analysis (Weka).....	42
Figure 2.1 Biometric Experiment: Capturing Sensors' Data to Train/ Authenticate	62
Figure 2.2 SVM: Find the Largest Margin Separating Two Groups of Data [35]	63
Figure 3.1 Sensor Kinetics Pro App: Showing the X, Y and Z Graphs.....	67
Figure 3.2 Sensor Kinetics Pro: Starting the App.....	69
Figure 3.3 Sensor Kinetics Pro: Starting the Accelerometer or Gyroscope Sensor.....	70
Figure 3.4 Sensor Kinetics Pro: Save Captured Sensor's Data Locally	71
Figure 3.5 Sensor Kinetics Pro: Save captured Sensor's Data Locally	71
Figure 3.6 Sensor Kinetics Pro: Saving Captured Sensor's Data	72

Figure 3.7 Weka: GUI Chooser User Interface	73
Figure 3.8 Weka: Selecting Data Type as Input to Weka.....	74
Figure 3.9 Weka: Selecting the Algorithm Type	76
Figure 3.10 Sample Result of a Run Using Naïve Bayes (NB) Algorithm	77
Figure 3.11 Simple Method: Graph Motions - Manual Feature Extraction Highlighted	80
Figure 3.12 Arbitrary Motions in the fall 2015 Research.....	81
Figure 3.13 Test: Process Started Smartphone on a Table	81
Figure 3.14 Test: Sensor Kinetic Pro: App Home Page	82
Figure 3.15 Test: First Motion - Bring the Phone to View (First Long Motion).....	82
Figure 3.16 Test: Second Motion – Pause While Viewing.....	83
Figure 3.17 Test: The Second Long Motion: Move the Phone to the Ear Level.....	83
Figure 3.18 Process: Complete Capture and Evaluation of the Data from Motions.....	84
Figure 3.19 Data Storage: View of the Folders on Google Drive	85
Figure 3.20 Data Storage: Accelerometer and Gyroscope Data - Participant’s Trials	86
Figure 3.21 Data Storage: Google Drive Files Locally	87
Figure 3.22 Data Storage: Contents of a Directory with Data from the Sensors.....	88
Figure 3.23 Data Storage: Sample *.CSV Data from Gyroscope.....	88
Figure 3.24 Data Storage: The Aggregated Data with Average and Variance	89
Figure 3.25 Weka: A Sample Execution with Accuracy Rate.....	89
Figure 3.26 Multilayer Perceptron: Input, Output, and Hidden Layers [13]	92
Figure 3.27 Acceleration vs. Time.....	94
Figure 3.28 Data: Raw Motion Data.....	95
Figure 4.1 Research Timeline	99
Figure 4.2 Profile of Test Participants in spring 2016 Research	110
Figure 4.3 spring 2016: Summary of Research Data and Activities.....	110
Figure 4.4 Simple Division Method.....	113

Figure 4.5 Complex Processing of the Motion to Isolate Movement	114
Figure 4.6 Feature Extraction Summary	115
Figure 4.7 Accuracy Rate: Average vs. Variance.....	117
Figure 4.8 All Runs vs. the Additional Test Using Either Average or Variance.....	118
Figure 4.9 Using Simple Methods for Different Size Datasets	119
Figure 4.10 Complex Method: Accuracy Rates - k-NN Algorithm vs. Naive Bayes.....	120
Figure 4.11 Comparison of Results from Three Algorithms	122
Figure 4.12 Approximate Values of Improvements Using Different Parameters	124
Figure 4.13 True Positive and False Positive Rates per Individual Test Participant	125
Figure 5.1 Confusion Matrix.....	130

List of Tables

Table 2.1 Biometric Experiment Data: Walking, Jogging, and Stairs [32]	60
Table 2.2 Biometric Experiment: Accuracy Rate Using J48 & Neural Net Algorithm [33, 34]..	60
Table 2.3 Biometric Experiment with Multiple Sensors: Accuracy Rate [35].....	62
Table 3.1 The Data Type within a CSV File	75
Table 3.2 A CSV-formatted Accelerometer Data from the App. (3-axis).....	80
Table 3.3 Data extracted from Sensor in CSV-Formatted Form	93
Table 4.1 Correct Classified Instance - Bringing Phone to View.....	100
Table 4.2 Correct Classified Instance When Phone at Ear Level	101
Table 4.3 Correct Classified Instance - Hanging Up the Phone	101
Table 4.4 Fall 2015 Research - Accuracy Rate with Accelerometer - Manual Extraction.....	104
Table 4.5 fall 2015 Research - Accuracy Rate with Gyroscope – Manual Extraction	105
Table 4.6 fall 2015 Research: Accuracy Rate – Two Sensors – Manual Extraction	105
Table 4.7 fall 2015 Research: Accuracy Results (Automated Run)	106
Table 4.8 Data Captured from the Sensor Kinetics Pro Application.....	112
Table 4.9 Using Simple Methods Using Different Size Datasets.....	119
Table 4.10 Results for Complex Method Different Sample Sizes.....	120
Table 4.11 Comparison of Results from Four Algorithms	122
Table 4.12 Approximate Values of Improvements of Research in Spring 2016.....	123

List of Equation(s)

Equation 1.1 Naive Bayes (N-B) Formula [10]	39
---	----

Names and Acronyms Equivalence

The following names or acronyms are equivalent as shown in the body of the document:

- Naïve Bayes and Naive Bayes are the same.
- The Acronym for Naïve Bayes is shown either as N-B or as NB. They are the same.
- The Acronym for K-Nearest Neighbor is shown either as K-NN or as k-NN. They are the same.
- The words IRIS, Iris, and iris are the same.
- Multilayer Perceptron and multilayer perceptron are the same.
- Sensor Kinetics Pro and Sensor Kinetics Pro Mobile App are the same.
- Mobile App and Mobile Applications are the same.
- Mobile Apps and mobile apps are the same.
- WEKA and Weka are the same.
- Test subjects or, simply noted, as subjects are the same.
- Test subjects and test participants are the same.

Chapter 1

Introduction

This chapter introduces the use of biometrics to authenticate smartphone users. It will provide the information needed to support the study and set the context for the following chapters.

Below is the road map for this chapter. It will provide:

- Background information on the different types of technologies available on smartphones to authenticate users such as: passwords, Persona; identification Numbers (PINs), tokens, physiological biometrics, and the focus of this dissertation: behavioral biometrics;
- Background information on sensors and how some sensors can capture biometrics like motion;
- Background information on Machine Learning concepts, algorithms and tools;
- The problem statement, relevance and scope of this study;
- The conclusion to the chapter and the roadmap for the rest of the document and chapters.

1.1 Overview

This study starts by asking if behavioral biometrics, specifically human motions, are suitable for purposes of authenticating users of smartphones. User authentication via behavioral biometrics has been gaining momentum as more and more people use smartphones. This term,

“behavioral biometrics,” covers many gestures like touch, motion, orientation, mouse dynamics, handwriting, voice, hand geometry, and gait [40].

The challenges inherent in authentication make behavioral biometrics appealing for several reasons. For one, it will likely be harder for someone with malicious intent to successfully capture a natural motion compared to a password or even a fingerprint. Natural motions also provide the option of active authentication since motions such as holding the device, walking around with it, and holding it up to one’s ear are ongoing activities. This study focuses on two specific hand motions when one holds an Android-based phone: 1) to lift the phone to view at eye level, and 2) move the phone to the ear. These two motions are used daily worldwide by users and thus provide an opportunity to analyze unique biometrics characteristics of these individuals.

The outcome of the study is to support the use of behavioral biometrics in smartphones to authenticate users, especially when used in tandem with other authentication technologies such as passwords.

1.2 Current Authentication Methods/Technologies on Smartphones

There are several general categories of available authentication methods for smartphones specifically on Android-based smartphones. Users utilize the available user interface (UI) on the smart phones to communicate with the devices. Currently, smartphones use passwords, tokens, and biometrics to authenticate a user. These existing mobile authentication techniques can be broadly classified into three paradigms:

- Passwords: something **you know** such as an alphanumeric/graphical password, PIN, or piece of personal information (such as a mother's maiden name) [47];
- Tokens: something **you have** such as a card key, smart card, or token (like a SecurID card) [47]; and/or
- Biometrics: someone **you are** [47] involving both biological and behavioral characteristics

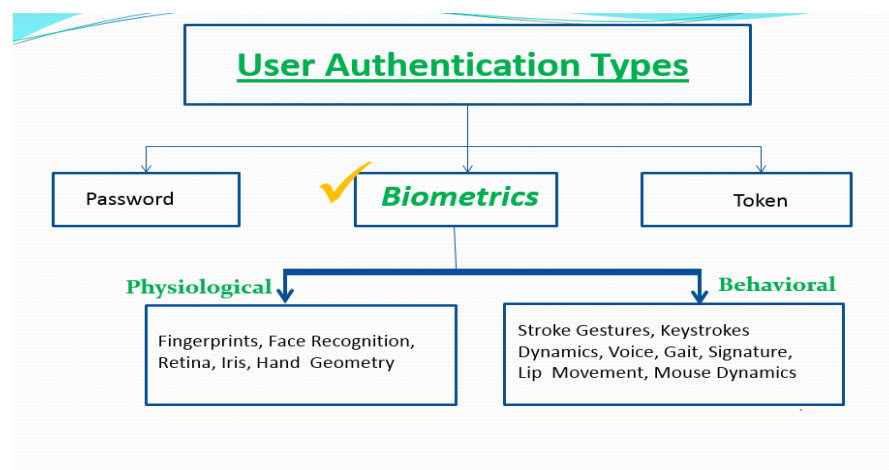


Figure 1.1 Different User Authentication Types

Figure 1.1 above shows the different user authentication types.

1.3 Using Passwords to Authenticate Users

A password is a word or string of characters used for user authentication to prove one's identity to gain access to a resource [46]. Everyday users utilize their individual IDs and passwords, either created by them or given to them, to gain access to a system.

Password authentication has been widely used in many types of devices for years. Users are asked to create, and later, to enter a certain number of digits, characters, or combination thereof, to access a smartphone. While this method has been popular and widely accepted worldwide, it

carries certain risks. These risks are well documented. Passwords can be stolen, guessed, hacked and discovered.

A typical computer user has passwords for many purposes: logging into accounts, retrieving e-mail, accessing applications, databases, networks, websites, and even reading the morning newspaper online [46]. Passwords can be a collection of letters from the alphabets, numeric, alphanumeric, special characters, or combinations of all of them. Despite the name, there is no need for passwords to be actual words; indeed, passwords that are not actual words may be harder to guess: a desirable property of a good password. Some passwords are formed from multiple words or may more accurately be called a passphrase [46].

Most organizations establish a password policy that will require users to create their passwords based on certain specifications like a combination of uppercase and lowercase characters, numbers, and special characters like question marks or exclamation marks. These typically must be of some minimum length. Passwords are usually valid for only certain periods and may expire. Some governments have national authentication frameworks [1] that define requirements for user authentication to government services, including requirements for passwords [3].

1.4 Using Security Tokens to Authenticate Users

Security tokens have also been used to authenticate a user. A token is something that a user has [6, 47, 48]. Security tokens are used to prove one's identity electronically, as used when a customer attempts to access his/her bank account. The tokens could be equivalent to an electronic key to access a system—be it a bank or a corporation's back end system, etc. [49].

Users can utilize both a token and a password together for authentication purposes. Some tokens can store one's electronic signature, biometric data like fingerprint details, or passwords. Some tokens are tamper-resistant, while others are equipped with small keypads to enter a PIN (Personal Identification Number; e.g., a series of numbers) or a simple button to start generating a set of random numbers that can be displayed as shown below [49]:



Figure 1.2 Soft Token – Randomly Generated Number [by Entrust Company] [49]

Figure 1.2 [49], above, shows a randomly generated number on an equipment supported by the Entrust company.

Some others may use a Universal Serial Bus (USB) connector, Radio-Frequency Identification (RFID) functions or Bluetooth wireless interface to enable transfer of the generated key number sequence to a client system [48]. These numbers are generated at a server and they are then transmitted to a client.

As noted above, many use tokens in addition to a password to provide an additional layer of protection. This combination is known as a two-factor or multifactor authorization [50]. The security token's small design allows transport via keychain, pocket or purse [50].

The three main security token types are as follows:

- **Connected token:** These tokens require an actual physical connection to generate automated authentication data transfer. Some of the widely used connected security tokens include Universal Serial Buses (USBs) and smart cards [48, 50].
- **Disconnected tokens:** These tokens are very popular and are used for two-factor authentication. They may require a Personal Identification Number (PIN) before generating authentication data. It does not connect, physically or logically, to a host computer [48, 50].
- **Contactless tokens:** The uses of these types of tokens are few or are specific to special applications. They use Radio Frequency Identification (RFID) technology. These tokens are physically connected to the back-end computer for authentication purposes.

All tokens contain some secret information that is used to prove identity. There are, however, risks in using the above-mentioned tokens: they can be physically stolen (in the case of connected or disconnected tokens), or the transmitted information could be intercepted, like when RFID tokens are used.

1.5 Using Biometric Technology to Authenticate Users

Biometric technology deals with “what you are.” These technologies measure individuals’ unique personal characteristics to recognize or authenticate their identity. Biometric technologies can essentially be divided into two categories: physical and behavioral.

Common physical biometrics includes fingerprints, hand or palm geometry, retina, iris, or facial characteristics. Behavioral characters include signature, keystroke pattern, and gait. Of this class of biometrics, technologies for signature and voice are the most developed [47].

Physical biometrics relies on creating a profile of an individual based on a physical characteristic. The most common are: fingerprint, iris, facial, hand geometry, vein pattern, voice and retina. Behavioral biometrics is based on a measurable behavior used to recognize or verify the identity of a person. It focuses on behavioral patterns rather than physical attributes.

To employ biometric authentication, a biometric system must be created. A biometric system is essentially a pattern recognition system that operates by acquiring biometric data from an individual, extracting a feature set from the acquired data, and comparing this feature set against the template set in the database [51].

There are many commercial and government applications. For example, when the U.S. Federal government wants to keep track of who is coming into the country, biometrics may be the best way to do so. Unlike with documents, it is very hard for a traveler to present a forged copy of a fingerprint or iris. That is why the U.S. Department of Homeland Security plans to vastly expand the amount of biometric data it collects at the borders [54].

While this study will focus on behavioral biometrics, some discussion of physical biometrics is also presented in this chapter.

1.6 Physical Biometric Technologies

As noted earlier, among some of the well-known physical biometrics are fingerprints, iris shape, or facial characteristics [47].

1.6.1 Fingerprint

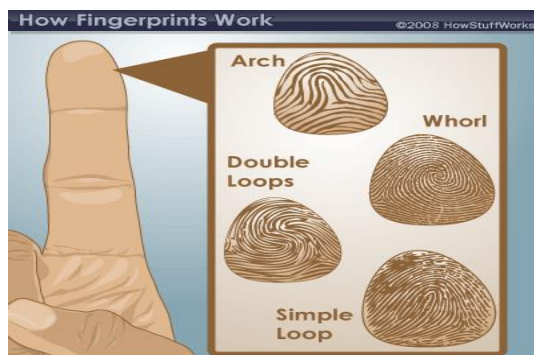


Figure 1.3 Types of Fingerprints [52]

Figure 1.3 above shows the different types of fingerprints [52].

Humans have used fingerprints for personal identification for many centuries and the matching accuracy using fingerprints has been very high [51, 53].

Fingerprints are the tiny ridges, whorls and valley patterns on the tip of each finger [52]. Fingerprints are now widely used in many devices including smartphones. Typically, a fingerprint scanner (e.g., used for passport, crimes, or driver licenses, etc.) takes an image of one's finger to determine ridges and valleys in the image. Systems, like smartphones, use this image to compare or match against what they have stored prior images (e.g., in a database). If there was a match between the image that was taken and what had been stored in the database the user would then be authenticated. Smartphones have been using this technology in recent years to authenticate users.

Fingerprint matching has been one of the most widely used biometric authentication methods for decades and continues to be popular today since every individual has a unique fingerprint. In a survey conducted by Ericsson in 2014, 52% of surveyed smartphone users

would prefer to use their fingerprints instead of passwords overall and even 50% would prefer to use their fingerprints to authorize payments online. In addition, 61% would prefer to use fingerprints to unlock their phones [63, 76].

Depending on the application context, a biometric system (in this example to capture a fingerprint) will have three modes [51]:

- In the enrollment phase, the system enrolls a person by capturing his/her fingerprint profile and stores it in a database.
- In the verification mode, the system validates a person's identity by comparing the captured biometric data with the user-specific biometric template(s) stored in the system database [51].
- In the identification mode, the system recognizes an individual by searching the templates of all the users in the database for a match [51].

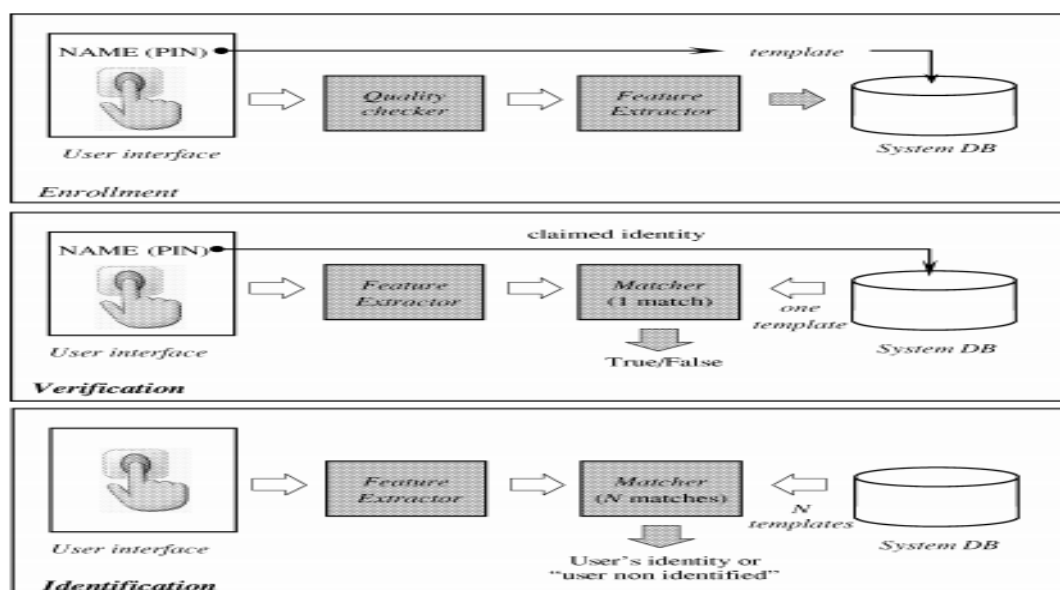


Figure 1.4 Block Diagrams: Enrollment, Verification, and Identification [51]

The Figure 1.4 above shows an example of Block diagrams, which contains Enrollment, Verification, and identification [51].

1.6.2 Iris Recognition

The iris is the annular region of the eye bounded by the pupil and the sclera (white of the eye) on either side. The visual texture of the iris is formed during fetal development, and it stabilizes during the first two years of life [51]. Each iris is distinctive and, like fingerprints, even the irises of identical twins are different. It is extremely difficult to, e.g., surgically, tamper the texture of the iris [51].

Iris recognition has been used in many airports (e.g., Washington Dulles or Amsterdam Schiphol, London Heathrow, and Dubai Airports) which provide recognition services to the immigration offices/departments of many countries. Scanning is widely used in airport technologies because everyone's iris is unique. An iris scanner uses biological characteristics that are stored in the database for security checks [54]. It takes a clear picture of a person's iris. Each passenger's iris is scanned during the airport security check and the information is verified with the existing database [54]. Figure 1.5 below shows a sample of steps involved in an IRIS recognition system [115].

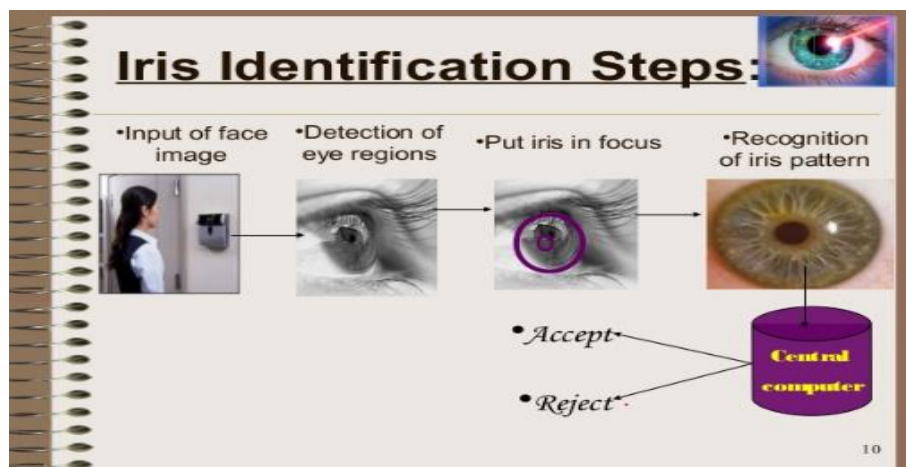


Figure 1.5 Process: IRIS Identification Steps [115]

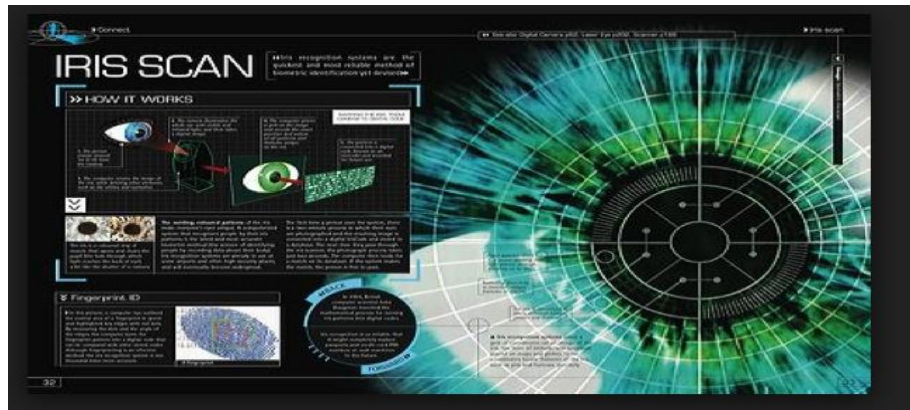


Figure 1.6 IRIS Scanners [54]

Figure 1.6 above shows an example of an IRIS scanner [54].

Iris scanning combines security and efficiency. Passengers can be processed accurately and quickly from immigration lines at airports. For example, the Schiphol airport in Amsterdam employs iris scan cards to speed up the passport and visa control procedures [51, 58].

The facial recognition at Schiphol airport process can be described as below:

- Passengers who are enrolled in this scheme receive an Iris card;
- These passengers insert their card at the gate and then look into a camera;
- The camera acquires the image of the traveler's eye and processes it to locate the iris and compute the Iris code [51, 55];
- The computed Iris code is compared with the data residing in the card to complete user verifications [51].

1.6.3 Facial Recognition

Facial Recognition is also a common sight at airports, particularly for travelers traveling the U.S. [63]. Facial recognition technology measures and matches the unique characteristics of each

face for identification or authentication. Often leveraging a digital or connected camera, facial recognition software can detect faces in images, quantify their features, and then match them against stored templates in a database [81].

Figure 1.7 below is an example of a facial recognition system [82].



Figure 1.7 Facial Recognition [82]

Biometric identification technology utilizing personal biometric features like facial recognition has been attracting a lot of attention from researchers and engineers in recent years [83-93]. Face recognition used to achieve identification by the analysis and comparison of facial visual features is the most natural and effective method to identify a person; it is safe, convenient, and fast [83].

Facial recognition analyzes the characteristics of a person's facial image input through a digital video camera. It measures the overall facial structure, including the distances between the eyes, nose, mouth, and jaw edges. These measurements are retained in a database and are used as a comparison when a user stands before the camera [94].

This biometric has been touted as a system for recognizing potential threats (whether terrorist, fraud artist, or known criminal) [94]. Every face has numerous, distinguishable landmarks, composed of the different peaks and valleys that make up facial features [94].

Each human face has approximately 80 nodal points. Some of these measured by the Facial Recognition Technology are distance between the eyes, width of the nose, depth of the eye sockets, the shape of the cheekbones, and the length of the jaw line. These nodal points are measured creating a numerical code, called a face-print, representing the face in the database [94].

1.6.4 Voice Verification

Voice biometrics uses the pitch, tone, and rhythm of speech. From a user's perspective, the implementation costs are reasonable since no special hardware is required—simply a telephone or microphone [63].



Figure 1.8 Voice Verification [96]

Figure 1.8 above is an example of Voice verification [96].

Voice biometrics works by digitizing a profile of a person's speech to produce a stored model voiceprint or template. Biometric technology reduces each spoken word to segments composed of several dominant frequencies called formants. Each segment has several tones that

can be captured in a digital format. The tones collectively identify the speaker's unique voiceprint. Voiceprints are stored in databases like those storing fingerprints or other biometric data [95].

1.7 Behavior Biometric Techniques

Behavioral biometrics is another very important biometric. The behavioral type includes learned movements such as signature, keyboard dynamics (typing) [15, 63, 64], and mouse biometrics.

1.7.1 Signature Verification

Signature is a behavioral biometric that encodes the flight movements of the signer for his or her chosen signature. Signatures have a widespread acceptance, and they have been commonly used by banks to validate signed transactions for years [56]. Signatures are among the most widely accepted attributes for personal authentication, both socially and legally [77, 78]. Many current mobile devices, including tablet computers, smartphones, and digital pens, allow the users to sign with a stylus or their fingers [77, 79].

Biometric systems to verify a signature or handwriting do not just look at how one shapes shape each letter; they analyze the act of writing. They examine the pressure one uses and the speed and rhythm with which one writes. They also record the sequence in which one forms letters, like whether one adds a dot or crosses a letter [80].

Handwriting



Figure 1.9 Signature Verification [80]

The Figure 1.9 above is an example of a signature verification system [80].

1.7.2 Keystroke Dynamics

It is generally understood that authentication serves two primary purposes. The first purpose is to identify correctly those users who are authorized to access a resource such as a web page or a database, and the second purpose to deny access to those who are not correctly identified [65]. It is also generally understood that passwords are not strong and can be stolen. The insufficient level of security provided by passwords is what has fueled the search for alternative forms of authentication [65, 66].

A need for improved authentication methods exists on a wide range of devices, from servers that store personal information to desktops, laptops, and mobile devices that are used to access information on demand [65]. The need for improved authentication forced researchers to consider different options, including keystroke dynamics as an alternative to authenticate users.

The research into keystroke dynamics as an authentication method relies on developing a technique that is robust, inexpensive, and has the potential to be transparent to the user [65]. It was researched as early as 1975 [65, 67] and has been the subject of several patents [65, 68, 69, 70]. Its feasibility has been examined on desktop computers [65, 71, 72, 73], web applications [65, 74], and mobile devices such as smartphones and PDAs [65, 75].

Keystroke dynamics is a behavioral biometrics measurement that identifies users based on the typing patterns of the individual, and thus it can be used to authenticate a user. Keystroke patterns are collected of how a user types on a standard keyboard or a mobile device. Since this behavior is unique to an individual and to his/her profile, authentication can be performed.

Identifying or authenticating people based on how they type is not a new idea, but thanks to advances in artificial intelligence, it can now be done with a very high level of accuracy, making it a viable replacement for other forms of biometrics [60]. Since many users type on keyboards to interact with their devices or smartphones, it is thus not intrusive. Since having a keyboard is primarily what is required, the cost is negligible.

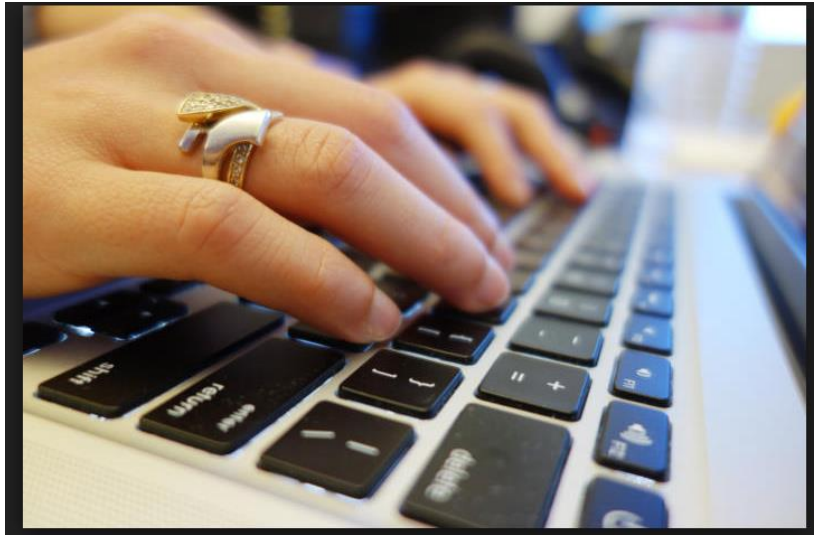


Figure 1.10 Keystroke Dynamics [60]

Figure 1.10 above is an example of a keystroke dynamics [60].

This biometric is based on three fundamental components:

- Key Press, which is the amount of time that a single keyboard button is pressed and released, measured as the time when the key is first pressed down until it is released;
- Key Flight, which is the timing between two different keyboard presses, and is measured as the time between when the first button is released and when the second button is pressed; and,
- Key Sequence, which is the time it takes to type from start to end a full sequence (e.g., word) or a part of a full sequence [56].

The keystroke dynamics biometrics can also be extended to touch screens, used mostly on smartphones, to authenticate users. In a study done at Pace University [57] using smartphones equipped with sensors like a gyroscope, they achieved an excellent very low EER (Equal Error Rate) of 4.3% which provided optimism that keystroke authentication can be achieved with high accuracy.

While in certain cases keystroke dynamics by itself may not always provide the highest expected accuracy, when implemented in conjunction with traditional schemes (e.g., password), keystroke dynamics allows for the design of more robust authentication systems than traditional password based alternatives alone [61, 62].

1.7.3 Mouse Biometrics

Mouse Biometrics is based on learning a unique pattern from user mouse movements, and like keyboard biometrics, it requires no special hardware for data collection. Mouse biometrics is based on collecting unique behavioral characteristics for a user based on his or her mouse actions, which consist of mouse movements and mouse clicks. The raw mouse data consists of mouse movement coordinates, movement angles, and the time to move the mouse from one location to the other, the timing of mouse clicks, and drag and drop movements.

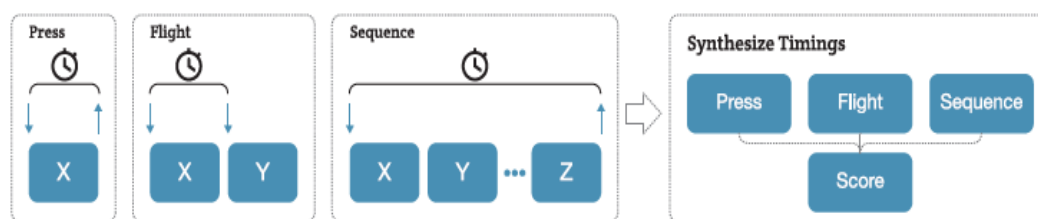


Figure 1.11 Mouse Biometrics [63]

Figure 1.11 above is an example of mouse biometric process [63].

1.8 Sensors: The Technology behind the Biometrics

Many of the recent smartphones come equipped with many sensors. Smartphones can use these sensors for biometric authentications. There are three categories of sensors: motions, environmental, and position [96]. For this study, motions sensors are considered. Of the several sensors in the motion category, accelerometer and gyroscope sensors are the ones that are relevant for this study.

1.8.1 Sensor Type: Accelerometer

An accelerometer is an electromechanical device used to measure acceleration forces. Such forces may be static, like the continuous force of gravity, or, as is the case with many mobile devices, dynamic to sense movement or vibrations [97]. Accelerometers are useful for sensing vibrations in systems or for orientation applications [14].

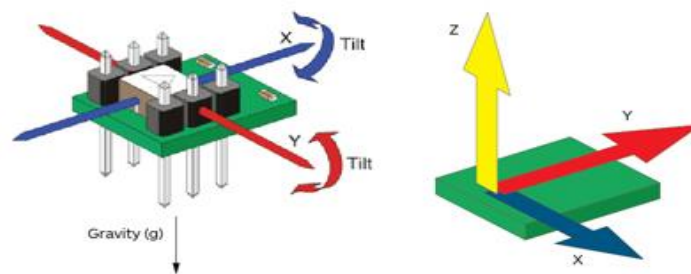


Figure 1.12 Sensor Type: Accelerometer [111]

Figure 1.12 is an example of an accelerometer works [111].

1.8.2 Sensor Type: Gyroscope

A gyroscope is a device that uses Earth's gravity to help determine orientation. Its design consists of a freely rotating disk called a rotor, mounted onto a spinning axis in the center of a larger and more stable wheel. As the axis turns, the rotor remains stationary to indicate the central gravitational pull, and thus which way is "down."



Figure 1.13 Sensor Type: Gyroscope [99]

Figure 1.13 above is an example of a gyroscope sensor.

1.9 Machine Learning Algorithms

Machine learning is one of the fastest growing areas of computer science with far-reaching applications in biometrics, medicine, transportation, academia, etc. Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed [100]. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from, and make predictions on, data. Such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is infeasible; example applications include spam filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), search engines and computer vision [100].

Machine learning tasks are typically classified into three broad categories:

1.9.1 Supervised Learning

This algorithm consists of a target/outcome variable (or dependent variable) which is to be predicted from a given set of predictors (independent variables). Using this set of variables, a function can be generated that maps inputs to desired outputs. The training process continues until the model achieves a desired level of accuracy on the training data. Examples of Supervised Learning are Regression, Decision Tree, Random Forest, k-NN, and Logistic Regression, etc. [101].

1.9.2 Unsupervised Learning

In this algorithm, one does not have a target or outcome variable to predict or to estimate. It is used for clustering population in different groups, which is widely used for segmenting customers in different groups for specific intervention. Apriori algorithm and K-means [101] are two examples of Unsupervised Learning.

1.9.3 Reinforcement Learning

Using this algorithm, the machine is trained to make specific decisions. The machine is exposed to an environment where it trains itself continually using trial and error. It learns from experience and tries to capture the best possible knowledge to make accurate business decisions. Markov Decision Process is an example of Reinforcement Learning [101].

1.9.4 List of Common Machine Learning Algorithms

Below is the list of commonly used machine learning algorithms. These algorithms can be applied to almost any data problem [101]:

1. Linear Regression
2. Logistic Regression
3. Decision Tree

4. Support Vector Machine (SVM)
5. Naïve Bayes (N-B)
6. K-Nearest Neighbors (k-NN)
7. K-Means
8. Random Forest
9. Multilayer Perceptron (MLP)

This study will use four of the above algorithms: Support Vector Machines, Naïve Bayes, K-Nearest Neighbors, and Multilayer Perceptron.

1.9.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a classification method. In this algorithm, one plots each data item as a point in an n -dimensional space (where n is the number of features) with the value of each feature being the value of a coordinate. For example, if there are only two features of an individual, like height and hair length, one must plot these two variables in a two-dimensional space where each point has two coordinates. These coordinates are known as **Support Vectors** [101].

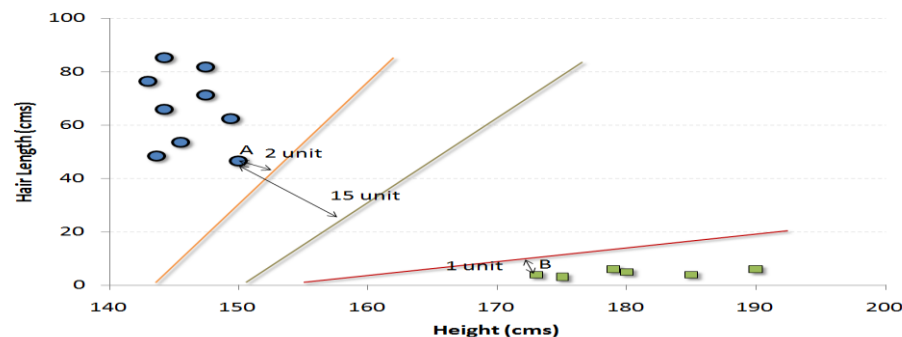


Figure 1.14 Support Vector Machine (SVM) Algorithm [101]

Figure 1.14 is an example of Support Vector Machine.

There is *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away. In the example shown above in Figure 1:14, the line that splits the data into two differently classified groups is the light *black* line, since the two closest points are the farthest apart from the line. This line is the classifier. Where the testing data lands on either side of the line determines what one can classify the new data as [101].

1.9.6 Naïve Bayes (N-B)

This is a classification technique based on Bayes' theorem of independence between predictors. In simple terms, a Naïve Bayes classifier assumes that the presence of a feature in a class is unrelated to the presence of any other feature. For example, a fruit may be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, a Naïve Bayes classifier would consider the properties that independently contribute to the probability that this fruit is an apple [101].

A Naïve Bayesian model is easy to build and particularly useful for very large data sets. Along with its simplicity, Naïve Bayes is known to outperform even highly sophisticated classification methods. Bayes' theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$.

Equation 1.1 Naive Bayes (N-B) Formula [10]

The diagram shows the Naive Bayes formula with labels for its components. The formula is $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$. Arrows point from the labels to the corresponding parts of the formula: 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Below the formula, the expanded form is given:

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Here, in the above equation 1.1,

- $P(c|x)$ is the posterior probability of class (target) given predictor (attribute),
- $P(c)$ is the prior probability of class,
- $P(x|c)$ is the likelihood that is the probability of predictor given class, and
- $P(x)$ is the prior probability of predictor.

1.9.7 K- Nearest Neighbors (k-NN)

K-Nearest Neighbors is an algorithm that stores all available cases and classifies new cases by a majority vote of its K neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function [101]. These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance [101,102]. The first three functions are used for continuous function and the fourth one (Hamming) is used for categorical variables. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor [101].

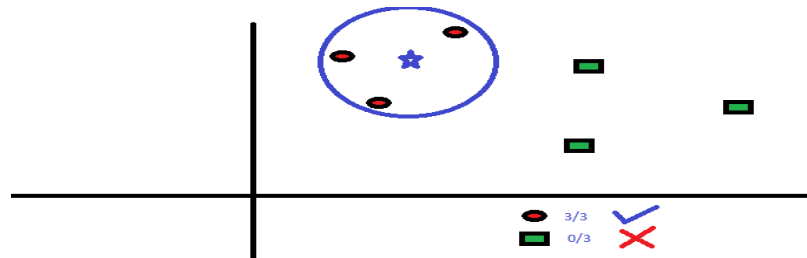


Figure 1.15 K-Nearest Neighbor (k-NN) Algorithm [101]

Figure 1.15 is an example of K-Nearest Neighbor (k-NN) [101].

k-NN can easily be mapped to real world lives. If one wants to learn about a person, of whom he or she has no information, he or she might like to find out about that person's close friends and the circles he moves in and out of to gain access to his or her information [101].

1.9.8 Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs [103]. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network [103, 104, 105].

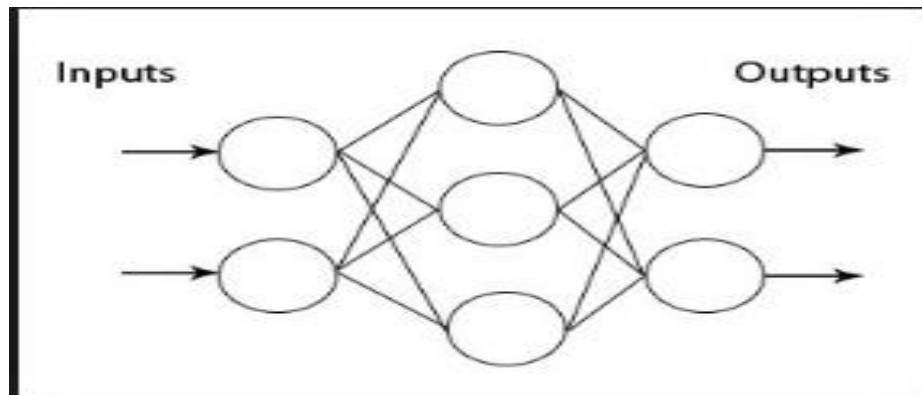


Figure 1.16 Multilayer Perceptron (MLP) Diagram/Algorithm [106]

Figure 1:16 is an example of Multilayer Perceptron (MLP) diagram [106].

1.10 Available Machine Learning Tools/Classifiers

There are many classifiers available on the market. Waikato Environment for Knowledge Analysis (Weka) and Pace University Dichotomy Classifier are two examples.

1.10.1 Waikato Environment for Knowledge Analysis (Weka)

Weka, a product of the University of Waikato, New Zealand, collects a set of Java machine learning algorithms engineered specifically for data mining. This GNU GPLv3-licensed collection has a package system to extend its functionality, with both official and unofficial packages available [107]. Weka (pronounced to rhyme with Mecca) is a workbench that contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions [108].

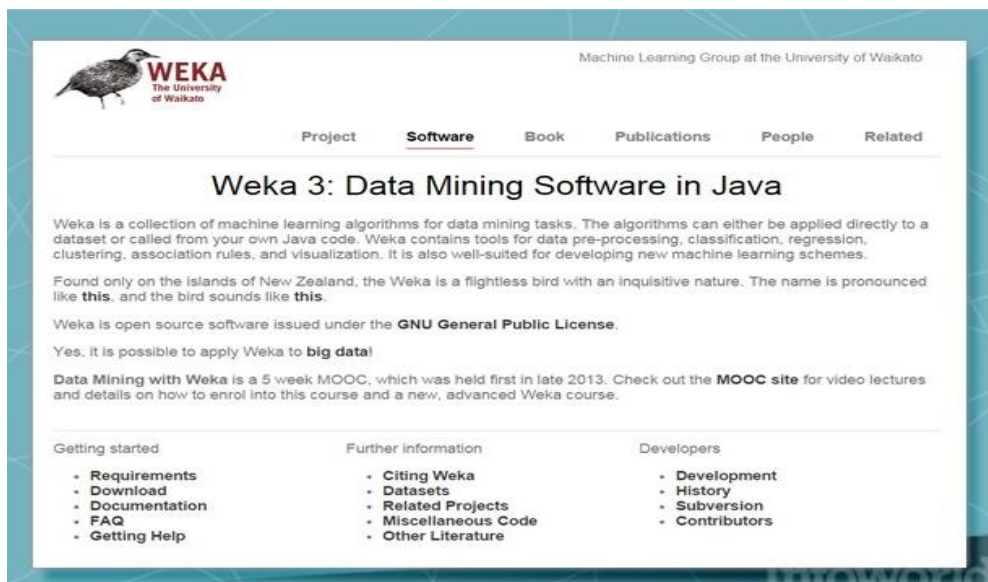


Figure 1.17 Waikato Environment for Knowledge Analysis (Weka)

Figure 1.17 is an example Weka (Waikato Environment for Knowledge Analysis).

1.10.2 Pace University Classifier

The Pace University Classifier utilizes a vector-difference authentication model, which transforms a multi-class problem into a two-class problem, resulting in an “authenticated” class and a “not authenticated” class.” A user’s feature sample is converted into a feature difference space by calculating the vector differences between pairs of samples of the same person, as well as calculating pairs of samples of different people [109, 110].

When authentication is attempted, the sample of a user’s profile is converted into a feature 9 vector, and that vector is then compared to a previously entered training feature data set. K-Nearest Neighbor is used to compare the differences between the offered feature vector with the previously captured set. What the difference vector returns when comparing the sample with the

training data determines whether the user is authenticated as the person they claim to be [109, 110].

1.11 Problem Statement

With the proliferation of smartphones worldwide, authenticating and protecting one's identity has become ever more important and critical. Smartphones are no longer simply used for personal use. More and more companies and organizations are requiring employees work with smartphones to access the organizations' systems and data—sometimes very sensitive data. The traditional ways of providing protections by using passwords or tokens to authenticate users are no longer adequate.

Passwords and tokens can be stolen. Hackers are becoming ever more sophisticated in gaining access to one's smartphone. They can compromise users' identity to gain access to a smartphone and then subsequently gain access to an individual's, sometimes, vast personal information. The hackers could go further; they could use the credentials they stole in the compromised smartphone to then penetrate an organization's system and gain access to its sometimes very important data for malicious reasons. While organizations are constantly working on new ways to protect their assets, hackers are not sitting idle, either. They too innovate and employ many new and different approaches to crack users' passwords.

Finding new ways to authenticate users is becoming very crucial. Using behavioral biometrics is the natural next step in the authentication technology where personal identity can be used to authenticate a user, through fingerprints, iris, hand geometry, motions, or gaits since they are unique to an individual.

Additional new studies and focuses are needed to expand the authentications field. The field of behavioral biometrics is vast and many different studies and research could be performed. One immediate and important area should be to focus on a very specific and targeted area within behavioral biometrics where a large number of individuals in public use and interact with smartphones daily.

One of these areas is hand motions. Smartphones are used daily around the world to conduct phone conversations. Smartphone users lift their phones to bring it into view (e.g., to see who is calling) and then move it to the ear to conduct a conversation. These two motions are normal and are widely used by the public worldwide every day. It is intriguing to see if these specific motions — very simple and yet widely used — could be unique to individuals and thus provide opportunities to authenticate them based on their individual behavioral biometrics signatures. Using these specific, targeted hand motions for behavioral biometrics is an interesting area of a study and research.

The contribution of this research is to leverage on that specific focus of behavioral biometrics to provide another solution. In this case, the study will investigate if hand motions alone are viable behavioral biometrics enough to be used as the sole authentication method if accuracy rate is very high like 98% and above. Or, if the accuracy is less than 98% use hand motion authentication in combination with another method like a password to reach high accuracy rate to authenticate smartphone users.

This study will capture users' hand motion data from some Android-based smartphones and will implement methods to use those data for "training" under some well-known Machine Learning algorithms like Naïve Bays (N-B), Support Vector Machines (SVM), K-Nearest Neighbor (k-NN), or Multilayer Perceptron (MLP).

1.12 Study's Relevance - General

Using behavioral biometrics has become very prominent recently in authenticating users, especially in smartphones. Passwords and security tokens have been the primary means of authenticating users for a long time, but they have shown to have inherent weaknesses. Hackers could steal the tokens or guess the passwords that could ultimately compromise a smartphone users' identity.

In the case of identity tokens, there are other considerations as well. There are usually some initial investments to acquire equipment, and there are future recurring costs for maintenance and licenses. There are also potential costs for training and retraining. These costs add up, and could in the end become very expensive for large organizations. Because of the drawbacks noted earlier — like inherent weaknesses that passwords and tokens have in protecting one's security — and expensive investments and recurring costs, many organizations and individuals have opted to search for other options. Biometric authentication provides the next solution that seems viable and economical.

1.12.1 Study's Relevance – So What?

So, the question is ... so what? Why is it so important to use behavioral biometrics? The following points provide the reasons for the relevance of this study:

- The biometrics of a person is something a person has that cannot be stolen. Using behavioral biometrics allows smartphones to take advantage of an individual's personal characteristics (e.g., fingerprint, gestures, voice, movements, motions, iris, etc.) to authenticate that individual when he or she attempts to log in to a smartphone.
- Behavioral biometrics is difficult to capture. It may be more difficult for someone with malicious intent to capture a person's natural motion as compared to a fingerprint or a password. Further, natural motions are committed continually in holding or walking around with a device or when holding it up to the user's ear; therefore there is an opportunity for a continuous form of authentication [40].
- Behavioral biometrics is mostly non-intrusive. Using behavioral biometrics for user authentication has huge potential even though historically it has been comparatively less established than the use of physiological biometrics or passwords and tokens [39]. Behavioral biometrics is mostly non-intrusive and includes touch gestures, motion, and orientation (that one normally applies to mobile devices), as well as other characteristics such as eye tracking, voice, mouse dynamics, handwriting, grip, or gait/stride [40] which users worldwide are performing daily. Being non-intrusive will make behavioral biometrics very appealing to users.
- Theft and security issues are increasing for smartphone users. As noted above and to emphasize again, several previous studies have investigated the significant increase for the use of non-physiological biometrics for

authentication in recent years [41, 42, 43, 44]. The reasons for the increase in interest in this field are obvious and numerous: smartphone proliferation [45], vast expansion of smartphone theft, cyber security issues, and desire for password/PIN alternatives [42].

- The technology is already here. Today's smartphones are highly sophisticated. They are equipped with many sensors like accelerometer and gyroscope, and they come packaged with solid and inexpensive software apps. The technologies come built in for the smartphones and biometric authentication can be easily.

1.12.2 Study's Relevance – Use Cases

There are many use cases for the use of behavioral biometrics. One use case is to authenticate users when they use their hand motions to lift their smart phones:

- to login to their bank accounts,
- to login to their social media sites,
- to open doors to secure areas,
- to make reservations at restaurants and hotels,
- to log in to many e-commerce sites like Amazon.com, or
- to gain access to information stored locally on the smartphone.

1.12.3 Study's Relevance – Using Hand Motions as Behavioral Biometrics

This study will focus on the movements of the hand when an individual lifts a phone to view and moves the phone to his or her ear to conduct a conversation. These two movements are widely used by the public. The study is relevant since it focuses on human hand motions that can be tracked to authenticate users.

Smartphones, through their embedded sensors, can capture one's movements to create a signature of that behavior. The raw data captured via the motions can then be used as training by some Machine Learning algorithms like Naïve Bayes or Support Vector Machine (SVM). The results could then be used to authenticate users.

1.13 Purpose of Study

The ultimate purpose of this study is to research whether two of the more popular hand motions can be used as behavioral biometrics to identify and authenticate users of Android-based smartphones. This study will answer the following questions:

- How accurately can users be identified when hand movements are used to bring smartphones to eye level and to ear level?
- What are the accuracy levels in these authentications?
- Which feature extraction sets provide the best results for providing authentication?
- What sensors are most relevant to capture the motions data for this study?
- What machine learning algorithms provide the highest level of accuracies?
- Will the population size matter for these algorithms to learn from the data?
- Which parameters within the Machine Learning (e.g., 10-fold) provide the best results?

1.14 Scope of this Study

This study will focus on the movements of the hand when an individual lifts a smartphone to view and takes the phone to his other ear to conduct a conversation as the basis for identifying

one's individual characteristics to authentic a user. These phones will be Android-based smartphones equipped with accelerometer and gyroscope sensors.

This study will consolidate the results of three research studies conducted in three separate semesters on the same topic. Every research study expanded on the prior semester's effort:

- by increasing the number of participating test subjects,
- by adding new and different Android-based phones,
- by applying additional feature extractions, and
- by employing additional Machine Learning algorithms.

1.15 Researches Conducted for this Study

There were a total of three researches conducted for this study. The three research studies were in spring 2015, fall 2015, and spring 2016. This dissertation consolidates all three research studies.

1.16 Chapter Conclusion

There are huge opportunities in the field of biometrics authentications to provide authentications and identifications, and there are many new areas yet to be researched. This chapter has shown different areas and types of biometric authentications, and it provided information on what different technologies are required (e.g., smartphones, software apps, sensors, Machine Learning algorithms, etc.) to conduct research and studies in the field of behavioral biometrics.

1.16 Roadmap

There are four more chapters in this dissertation document:

- Chapter 2, entitled “Review of the Literature and Related Work,” will review the available research and literature to provide additional background to support this study.
- Chapter 3, entitled “Solution Methodology,” will describe the methods utilized to capture and analyze data from smartphones.
- Chapter 4, entitled “Experiments and Results,” will provide the details of the research conducted for this study.
- Chapter 5, entitled “Conclusion,” will describe the conclusion of the study, and it will show the study accomplish to support the use of behavioral biometrics to authenticate users. It will also describe future work in this area.

Chapter 2

Review of the Literature and Related Work

2.1 Introduction

This chapter provides a literature search and evaluations to provide background and context for this research. This literature review will cover the well-publicized vulnerability cases in the news, the traditional models that have been used to protect systems (e.g., user IDs/password and tokens), and the weaknesses and limitations of the current models. The review will then cover new areas of research, specifically the area of biometrics authentications in smartphones. The chapter will also provide literature reviews on recent technologies used to support biometric authentication like Machine Learning (ML). The chapter will conclude with the literature review on a specific area of biometrics authentication, i.e., movements. While most of the literature reviews will be relevant for the majority of today's systems and computers, the focus of this study and literature review will be on biometric authentication and finding context for its use in smartphones, specifically in Android-based smartphones.

2.2 Security Breaches in the News

The New York Times reported on March 15, 2017 that the Justice Department charged two Russian intelligence officers with directing a sweeping criminal conspiracy that stole data from 500 million Yahoo accounts in 2014 [1].

The Washington Post reported on July 9, 2015 that in 2014, there were two major breaches of U.S. government databases holding personnel records and security-clearance files, which

exposed sensitive information of about at least 22.1 million people, including not only federal employees and contractors but also their families and friends [2].

Wired.com wrote on February 16, 2017 that Android-based phone hacks could unlock millions of cars. In the era of the connected car, automakers and third-party developers competed to turn smartphones into vehicular remote controls, allowing drivers to locate, lock, and unlock their cars with a screen tap. However, phones can be hacked, and when these phones are hacked, those car-connected features can fall into the hands of hackers, too [3].

The BBC reported on September 2, 2014 that Apple has confirmed that some celebrities' iCloud accounts were broken into, but said they had found no evidence that this was caused by a breach of its security systems. Instead, the firm suggested the perpetrators carried out their thefts by deducing victims' login credentials. The statement was followed with another story about the online publication of intimate pictures of about 20 personalities [4]. The report said that there had been speculations that the images were obtained due to vulnerability in software allowing users to locate missing iPhones that allowed unlimited password guesses [4].

CNN reported on May 9, 2016 that LinkedIn was hacked in 2012, and what initially seemed to be a theft of 6.5 million passwords was actually a breach of 117 million passwords [5]. The worst part is that, because people tend to reuse their passwords, hackers are likely to gain access to 117 million people's email and bank accounts [5].

These specific news stories, and many similar ones that are reported regularly, reflect the significant security vulnerabilities that exist with many of the systems the users interact with daily. Many malicious actors and hackers worldwide could access other people's information

stored in local computers, government agencies' databases, Cloud-provided services, social networking sites, smartphones, and many, many other sources.

2.3 Traditional Technologies Used to Protect Systems and Smartphones

Users have had several ways to interact with computer systems and smartphones over the years. Users have used user ids and passwords, token IDs, and other similar techniques or a combination of these techniques within the last several decades to access systems. These technologies while widely used have shown to be still vulnerable to hacking for malicious intent and to steal users' identity.

2.4 User ID and Password Authentications

The use of user ID and passwords has been one of the primary ways to gain access to systems and smartphones. Organizations provided user IDs and users created passwords to gain access to the systems. Users of Smartphones used passwords to login to their devices. These combinations of user ID and passwords, or just a password, have not proved to provide full protections to individual users for their personal use or to employees for their business use. These devices have been compromised when hackers find ways to find the users' credentials to gain access to users' personal information or gain access to organizations' systems.

Authenticating a user is the first step to allow a user to interact with a smartphone or with a system in general. These users usually use a password. The term "password" may be used for passwords, passphrases and PINs [6, 15]. Passwords are typically what only a user knows [15]. Using a single password creates vulnerability since passwords can be guessed or cracked.

Organizations and personal data will be compromised when hackers find ways to gain access to users' credentials (user IDs or passwords or a combination of these two.) For

organizations, their intellectual property (IP), marketing information, corporate strategy, or many of their other internal secrets could be compromised which could ultimately bring significant financial or reputational loss to the organizations. *US News and World Report* published a report from the Center for Strategic and International Studies and funded by cybersecurity firm McAfee reported that hackers are costing consumers and companies between \$375 and \$575 billion annually, a number only expected to grow as online information stealing expands with increased Internet use [7].

Vulnerabilities caused by these credentials in the form of weak passwords, for example, have caused a significant increase in cybercrimes. The following points were noted in a paper by Moshe Zviran, called “Password Security: An Empirical Study” [8]:

- In the fall of 1978, Stanley Rifkin obtained the electronic transfer code for the Security Pacific Bank in Los Angeles. Posing as a branch manager, he used the code to transfer \$13 million from Security Pacific to his Swiss bank account [9].
- A group of German hackers penetrated dozens of military, government, and commercial computer systems by cracking the passwords of legitimate users and system administrators. They were looking for military information that could be sold to the then Soviet Union [10, 11].
- In April 1994, an English teenager penetrated Pentagon computers and set off a massive security alert when his internet probing nearly provoked an act of war with North Korea. He entered the Department of Defense systems through the Air Force's Rome, New York laboratory using the default password “guest” [12].

- Early in 1998, a trio of Israeli teenagers hacked into the information systems of the Knesset, Israel's parliament. By guessing user passwords, they accessed 150 accounts. They left the data and system unharmed but sent the system administrator an e-mail message describing the system's security loopholes [13].
- Two California teenagers cracked the passwords of several Pentagon computers to enter data on the Department of Defense payroll data and personnel files [14].

Passwords do not always provide the required protection since there are ways for passwords to be cracked and the users' information is exposed and compromised. In recent times, another layer of protection has been used. It is called the advanced encryption standard (AES) that has recently been adopted as the standard encryption algorithm for the U.S. government [19].

Lawrence O'GORMAN explains AES:

“In physical terms, this algorithm is like a very strong bank vault, practically impossible to break into. For AES, the user chooses a private key to perform encryption and decryption. For the vault, there is a combination. The maximum AES key length is 256 bit. If an attacker were to try to guess the key, it would require on average over 10 guesses to do so, too time-consuming even by computers in the near future. However, a 256-b key is too long for most humans to remember, so in practice this key is stored in a computer file protected by a more memorable password. Therein lies the problem, because humans often choose a password that is not only memorable to them, but also easily guessable by a person or computer [20 – 26]. Using the bank vault analogy, this is like storing the vault combination on a piece of paper in a hidden place close to the vault. Now, all an attacker should do is to find the piece of paper and use the combination to open the vault. The strongest vault can be attacked by

exploiting a human mistake, just as the strongest encryption algorithm can be attacked by exploiting a weak password. Because user authentication deals with humans complete with humans' limitations and foibles, and because it often is the front-end protection of otherwise strongly secure systems, it is variously called the Achilles' heel, the weak link, and the last yard of secure systems." [15]

Passwords are currently the most common method of authentication. However, they suffer from several weaknesses. Passwords are vulnerable to attacks because they are easily guessed. They suffer from social engineering attacks, like phishing, pretexting, etc. The usability issue is also a serious factor, since users do not like to have to enter, and reenter, passwords or Pins [35]. Because of these limitations, many organizations have looked at other options like using security or identity tokens.

2.5 Identity Token Authentications

Tokens are typically known as "what you have" [15]. An identity token, security token, access token, or simply token, is a physical device or software that performs or aids authentication. This can be a secure storage device containing passwords, such as a bankcard, remote garage door opener, or smart card [15]. This can also be an active device that yields one-time passcodes and time-synchronous passcodes ... changing in synchrony with a master at the host [16].

There are certain advantages and disadvantages to security tokens. Security tokens, whether in the form of hardware or software, have the advantages of offering a second authentication factor. However, they can be stolen, they can incur additional cost to both users or organizations,

or require special training [17, 18]. Since the token is at risk of being lost or stolen, many combine it with another protection like a password.

As noted in [15], a token can provide three major advantages when combined with a password.

One is that it can store or generate multiple passcodes. This change the task of remembering multiple and changing passwords to one of remembering only the single password needed to access the token like a Single Sign On (SSO) device).

A second advantage is that it provides compromise detection; since its absence is observable and loss of a password is not.

The third advantage is that it provides added protection against denial-of-service attacks. For an account with only a password, an attacker can enter incorrect passwords for that user until the account locks out; whereas if combined with a token, the attacker cannot just enter incorrect passwords because they must steal the token first, which is presumably a more difficult task and one requiring physical presence.

There are some disadvantages as well; the two main disadvantages of a token are inconvenience and cost [15]. While identity tokens prove to be a very good way to protect users' identity, research still needs to be taken to the next level where tokens cannot be stolen or hacked.

This opens the door for biometrics authentication — identifying individuals based on their unique characteristics.

2.6 Biometrics Authentications

Biometrics is what a person is, and it cannot be stolen or hacked. Biometrics authentication has been used by itself, or it has been combined with another method like a password, to provide

a strong authentication option. As noted in [15], a biometric is a feature measured from the human body that is distinguishing enough for user authentication. Biometrics includes fingerprints, eye (iris and retina), face, hand, voice, and signature, as well as other more subtle obscure or futuristic biometrics [27, 28] such as gait and smell. A biometric purports to inextricably link the authenticator to its owner, something passwords and tokens cannot do, since they can be lent or stolen [15].

Since human biometrics is unique to everyone, finding ways to use one's biometrics to authenticate will be a huge leap forward for smartphone users. While systems may not always reach 100% accuracy in identifying individuals, unless the someday authenticate through DNA, this field is, increasingly becoming sophisticated with the application and use of new tools, new and multiple sensors, new Machine Learning tools and algorithms, etc. While there are many devices a user can interface with on biometrics authentications, this study will focus only on literature on the use of Android-based mobile phones.

2.6.1 Using Smartphones to Authenticate Users Performing Daily Activities

Many researchers have performed research on a specific area of biometrics authentication. Biometrics authentication has been used to interact with personal computers (PCs), smartphones, and other devices. For example, the researchers in [29-31] used the data extracted from an accelerometer to authenticate users or to identify user traits by mining smartphones. In [30], Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore conducted research using accelerometer data to authenticate users. They chose Android-based cell phones as the platform for their project because the Android operating system was free, open-source, and easy to program, and was expected to quickly become a dominant entry in the cell phone marketplace. The project used several types of Android phones, including the Nexus One, HTC Hero, and

Motorola Backflip. These Android phones contain tri-axial accelerometers that measure acceleration in three spatial dimensions. As noted before, the researchers in this project used accelerometer data to identify or authenticate cell phone users. The data is generated while the users perform normal daily activities such as walking and climbing stairs with a cell phone in their pocket [30].

The researchers in this study used certain data collection protocols, applied feature extractions, and worked with thirty-six volunteers who performed certain tasks. These subjects were asked to walk, jog, climb up stairs, and climb down stairs for specific periods while they carried the Android phone in their front pants leg pocket. The researchers generated informative features based on the 600 raw accelerometer readings and generated forty-three features. In general, they used statistical models like Average, Average Acceleration Value (for each axis), Standard Deviation, Standard Deviation (for each axis), Average Absolute Difference, and Average Absolute Difference, between the value of each of the 200 readings within the 10-second interval and the mean value over those 200 values (for each axis).

The researchers assigned each volunteer a unique ID and worked with them to perform, as noted earlier, a certain number of activities like walking, jogging, walking up the stairs and down the stairs, and totaled these number of activities per user.as shown in Table 2.1

They captured six different datasets per user. Table 2.1 below shows the numbers for walking, jogging, and going up and down Stairs, and the number of examples per user and per activity [32].

Table 2.1 Biometric Experiment Data: Walking, Jogging, and Stairs [32]

ID	Walk	Jog	Up	Down	Total
1	74	15	13	25	127
2	48	15	30	20	113
3	62	58	25	23	168
4	65	57	25	22	169
5	65	54	25	25	169
6	62	54	16	19	151
7	61	55	13	11	140
8	57	54	12	13	136
9	31	59	27	23	140
10	62	52	20	12	146
-	-	-	-	-	-
30	35	31	28	19	113
31	64	55	17	16	152
32	34	32	0	0	66
33	64	0	0	0	64
34	59	59	0	0	118
35	55	46	19	12	132
36	87	81	23	16	207
Sum	2081	1625	632	528	4866
%	42.8	33.4	13.0	10.8	100

The researchers used Weka data mining suite [32] to induce models for personal identification. They used decision trees (J48) [33] and neural networks (NN) [34]. The researchers found a high level of accuracy overall and correctly identified with very high accuracy the most frequent users. Table 2.2 below shows their accuracy rate:

Table 2.2 Biometric Experiment: Accuracy Rate Using J48 & Neural Net Algorithm [33, 34]

	Aggregate	Walk	Jog	Up	Down	Aggregate (Oracle)
J48	36/36	36/36	31/32	31/31	28/31	36/36
Neural Net	36/36	36/36	32/32	28.5/31	25/31	36/36

This specific study showed that Android-based smartphones could be used to positively identify and authenticate users.

2.6.2 Use of Multi-Sensor Authentication to Improve Smartphone Security

In this section, the review will describe a research study performed by Wei-Han Lee and Ruby B. Lee from Princeton Architecture Lab for Multimedia and Security (PALMS) of the Department of Electrical Engineering at Princeton University [35].

Their report notes that the widespread use of smartphones gives rise to new security and privacy concerns. Smartphone thefts account for the largest percentage of thefts in recent crime statistics. Using a victim's smartphone, an attacker can launch impersonation attacks, which threaten the security of the victim and other users in the network [35]. Lee & Lee developed a threat model including the attacker taking over the phone after the user has logged on with his or her password or PIN [35]. The intent of the research was to design a mechanism for smartphones to better authenticate the current user, continuously and implicitly, and raise alerts when necessary.

The authors used Android-based mobile phones like the Nexus 5. The intent of the research was to use three sensors in a smartphone: accelerometer, orientation sensor, and magnetometer. The system leveraged data collected by the three sensors and then trained a user's profile using the SVM [36] machine learning technique. The system continuously authenticated the current user without interrupting user-smartphone interactions [35].

The authors used the model shown in figure 2.1 to retrieve data from the sensor by sensing, constructing a feature extraction, re-sampling the data, constructing the training, and finally authenticating [35].

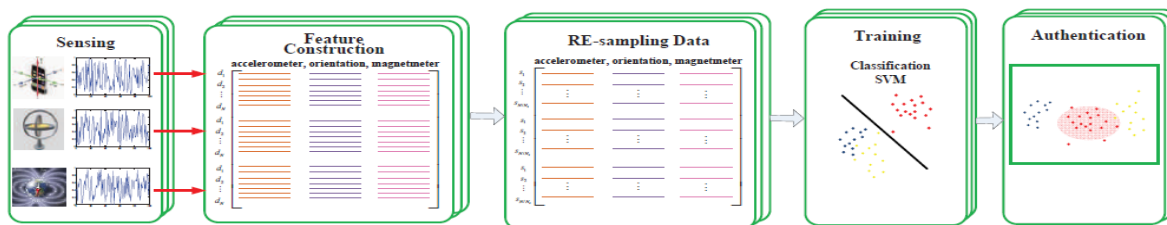


Figure 2.1 Biometric Experiment: Capturing Sensors' Data to Train/ Authenticate

In this method, the researchers first constructed a vector at each time by using sensors' data. They used nine values from the accelerometer, magnetometer and orientation sensors in a smartphone. Then they re-sampled the data collected from the sensors. Finally, they trained the re-sampled data with the SVM technique to get a user's profile. Based on the user's profile, they could do the implicit authentication [35]. Table 2.3 shows the results. The research provided an accuracy of 92.3%.

Table 2.3 Biometric Experiment with Multiple Sensors: Accuracy Rate [35]

Devices	Sensors	Method	Accuracy	Detecting time	Script
Nexus 5 Android	orientation, magnetometer, accelerometer	SVM	90.23%	train:6.07s test:20s	No

The authors [35] showed that multiple sensors could be combined to authenticate smartphone users.

2.7 Role of Machine Learning Algorithms in Biometrics Authentication

Biometric authentications on Smartphones require the use of sensors, even multiple sensors simultaneously, to capture data. These sensors produce huge amount of data. As was described in the prior sections, Machine Learning (ML) algorithms can be employed to process and train.

There are many different types of algorithms. The literature review showed that many of

these studies use algorithms like K-Nearest Neighbor (k-NN), Support Vector Machine (SVM), J48 Decision Tree, Neural network (NN), Multilayer Perceptron (MLP), and Naive Bayes (N-B). They each have their own advantages and disadvantages.

Some of the researchers like Wei-Han Lee and Ruby B. Lee [35] preferred using SVM. It allowed them to separate two groups of data. Below is an illustration of SVM [35]:

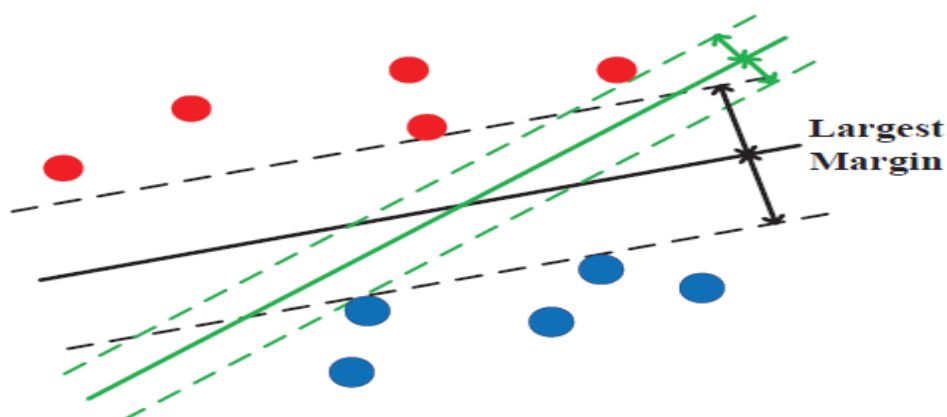


Figure 2.2 SVM: Find the Largest Margin Separating Two Groups of Data [35]

Figure 2.2 above is an example of SVM, which shows how to find the largest margin separating two groups of data [35].

2.8 Applications of Sensor-Based Smartphones

Users use smartphones for many applications. These applications can be in banking, medicine, education, military, and many others. Nationwide Bank in England has been exploring the ways in which smartphone users interact with their phones, from how fast they type to how they hold their devices, as a new way to verify and authenticate the consumers who use their mobile banking services [38].

2.9 The Focus of This Study/Research: Using Motion Data to Authenticate Users

This research will fill a gap of authentication for Android-based users when they hold a phone and move the phone with their hands to view and to the ear. The study of this specific application on motion is in line with other research and studies when phones are carried with users during walking, jogging, or other motions.

2.10 Chapter Summary and Conclusion

In this chapter, the different related work and literature were reviewed on the available authentication methods, like using passwords, identity tokens, and biometrics. The chapter presented in detail some of the research where researchers used Android-based phones to authenticate users. In all, this review found that there is significant literature on the methods to authenticate smartphone users, specifically in biometrics. It is interesting to note that much of the more recent literature has been on biometric authentication. This might be due partly on the findings that Machine Learning tools could now be used to process a huge amount of data to train to authenticate users. Smartphones are becoming ever more sophisticated with more and expansive sets of sensors that ultimately will help expand the field of biometric authentication.

Chapter 3

Solution Methodology

3.1 Introduction

This chapter describes the methods for this study. It will detail the processes used to capture data from mobile phones, the types of phones and the mobile apps used to capture the movements of the phones. It will show how the extracted data was analyzed by test participants, the organization of data, trials, and data formats (e.g., in Comma Separated Values - CSV format), and by sensors (accelerometer and gyroscope). Data preparation for running the experiments is also discussed, as well as data storage (e.g., Google Drive). Manual and automated feature extraction (via a Java program) is presented. Finally, the Weka Machine Learning tool is described for setting up and running the different algorithms: e.g., K-Nearest Neighbor (k-NN), Multilayer Perception (MLP), Support Vector Machine (SVM), and Naive Bayes algorithms. The specific attributes of the results will be presented to validate the problem statement in this research. For example, the time it took to execute the algorithms, per data sets, percentage of accuracies after the executions, etc. The chapter also has sections on tools, data gathering protocol, and data processing.

3.2 Chapter Organization

This chapter will have the following sections:

- Tools section: In this section, the two tools that this study used will be discussed. The first tool was a mobile app called “Sensor Kinetics Pro” that was used to capture the data from Android-based smartphones. The second tool was Weka, a Machine

Learning classification tool. Weka is a collection of Machine Learning algorithms for data mining tasks.

- Data Gathering Protocol section: this section will show how the data was collected, what types of phones and models were used, and what arbitrary motions/movements were selected for this study.
- Data Storage Section: this section will show examples of files created, the type of data storage used, and the types of contents within the folders in this data storage. It will also show the type of raw data contained in the files created from sensors, the calculated resulting files that contain aggregated data in the form of Averages and Variances derived from raw sensor data, and finally, the process to run these calculated files under Weka.
- Feature Extraction section: this section will show both the manual and automated feature extraction methods and Machine Learning algorithms used. This study used manual feature extractions in the research during spring 2015 and fall 2015s, and used automated feature extraction process via a Java program in the spring 2016 research.

3.3 Tools Used to Support the Methodology and Research

To support the methodology, there were two specific tools that the study used to capture and analyze the data.

3.3.1 The First Tool: Sensor Kinetics Pro Mobile App

The first step in the research was to capture the sensors' raw data. To capture sensors' data, there were several options available. One option was to write a programming language (e.g.,

Java, C++, etc.) with an Application Programming Interface (API) to capture the data from an Android-based phone.

Another option was to use an already available Android-based mobile app from the Google Play Store. This study found a mobile app and used this app to conduct the three separate research studies in spring 2015, fall 2015, and spring 2016. The study selected Sensor Kinetics Pro mobile app that had been developed by INNOVENTIONS® Inc. to run on Android-based phones.

This app detects the sensors that the phone possesses and allows them to be tested. It allows a user to apply filters to the data stream, save the raw sensor data to the user's phone, view the data in a tabular format right on the user's phone, and export it to a computer [116, 117]. Sensor Kinetics Pro gives a user a detailed and comprehensive look at the combined operations of all the sensors [116]. Figure 3.1 below shows the X, Y, and Z graphs on the Sensor Kinetics Pro App.



Figure 3.1 Sensor Kinetics Pro App: Showing the X, Y and Z Graphs

The following paragraphs show a quick review of the product and how one works with it.

The app supports several different groups and types of sensors. The three-dimensional sensors include the accelerometer, gyroscope and magnetometer. The derived 3-D sensors are the gravity sensor, linear acceleration sensor and a rotation sensor. The last group of sensors is the scalar sensor group that consists of the ambient temperature sensor, proximity sensor, light sensor, pressure sensor and relative humidity sensor. Sensor Kinetics Pro allows the user to monitor each sensor individually as a chart, save, and share the data. Sensor Kinetics Pro also allows the user to run multiple sensors at once to enable the observation of data captured from one motion from multiple sensors captured [13].

This study used the accelerometer and gyroscope sensors since they were the most common sensors found in smartphones and their combination covered the desired range of motion. After the data gets collected, the app permits the user to save the data. It saves the motion data points as Comma Separated Values (CSV) files. They have a “.csv” at the end of the file name, and they can be viewed as a graph. Included on the .CSV file are numerical values for time and the location of the phone on the x, y and z axes. This *.csv file can then be used with Weka to process the data [13].

A user can begin by downloading the app and opening it. The opening screen contains buttons for the accelerometer, gyroscope, and any other sensors the phone has. This screen displays the rate at which the sensor data is collected as well as the real-time data. A user starts the process by selecting the first button: “Multi-Sensor Recorder - new”.

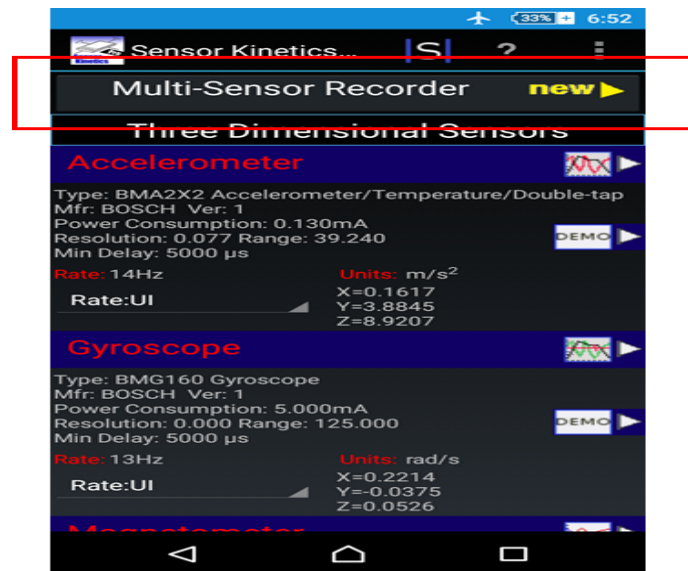


Figure 3.2 Sensor Kinetics Pro: Starting the App

Figure 3.2 above shows how to start the App to start capturing the sensors' data.

The sensors are broken down into sections. The next screen shows the sensors that are available. By tapping the blue dot next to the sensor, one can toggle that sensor's recording to either on or off. At the bottom of the screen is a Max Rate button. By tapping this button, one can choose the max rate at which the sensors are recorded. By selecting Sensor, each sensor will record at its max rate. If the sensors record at different rates, this option can allow one to set a rate. By selecting the rate of the slowest sensor, all sensors can be recorded at the same rate for easier data comparison. This screen also has a start/stop button and a clear button. To record the motions: press start, perform the desired motion, and press stop.

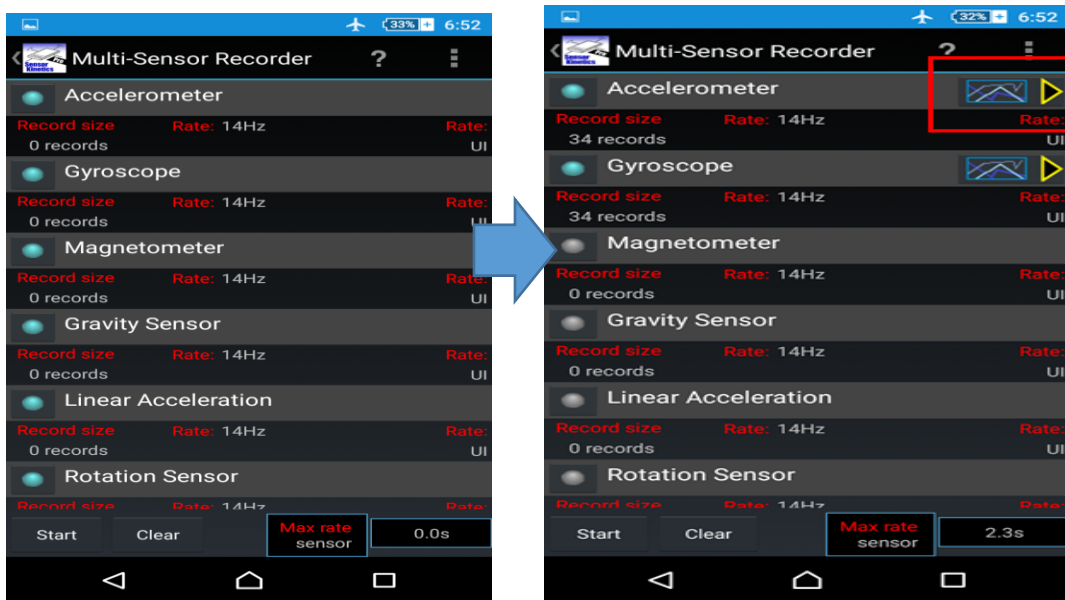


Figure 3.3 Sensor Kinetics Pro: Starting the Accelerometer or Gyroscope Sensor

Figure 3.3 shows how to start either Accelerometer or Gyroscope on the App.

Each sensor's recording data needs to be saved separately. Pressing the Chart button next to accelerometer will display a chart of the data that was just acquired. This option may be selected by pressing the button in the top right corner to open a menu where one has the option to save the data. Pressing the Files & Sharing option allows the user to save the file locally on the smartphone and then share it with others via e-mail.

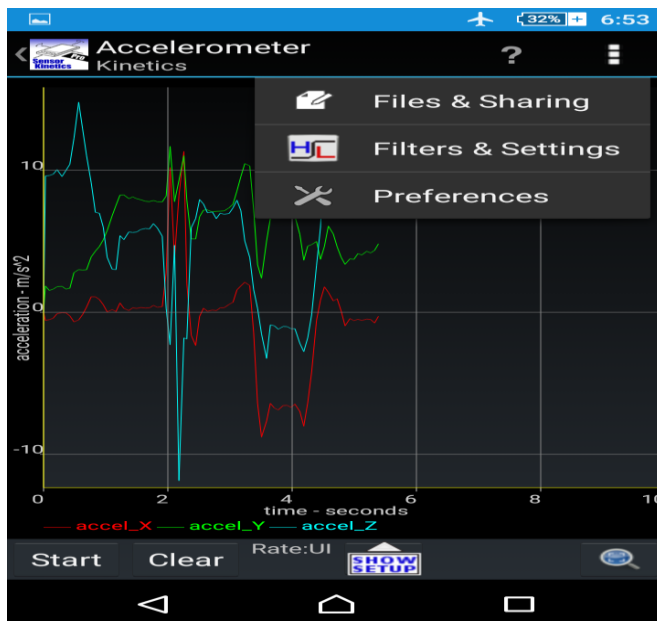


Figure 3.4 Sensor Kinetics Pro: Save Captured Sensor's Data Locally

Figure 3.4 shows how to save the captured sensors' data to a local file or share to send via e-mail. This will bring up a list of all saved files for that sensor. To save the data, a user presses the disk button on the top right, names the file, and presses OK.

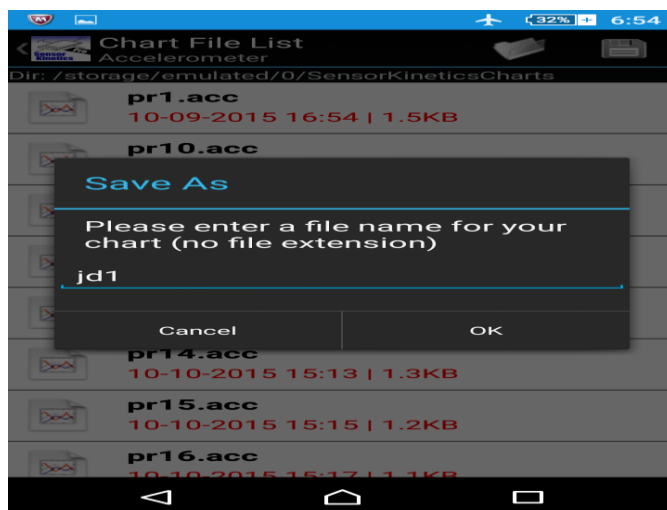


Figure 3.5 Sensor Kinetics Pro: Save captured Sensor's Data Locally

Figure 3.5 above shows how to save the sensor's data locally

Once the file is saved, clicking the file brings up an Options screen. To transmit the file, press the Share button. Press the Email button to display the phone share options. This includes email as well as shared drives such as Google Drive or Dropbox.

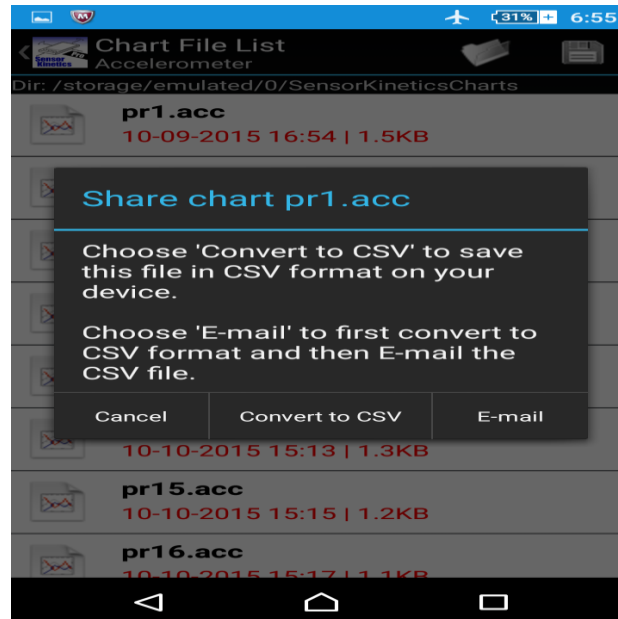


Figure 3.6 Sensor Kinetics Pro: Saving Captured Sensor's Data

Figure 3.6 above shows the selection to save the sensor's data locally to a CSV-formatted file to be sent to an address via an e-mail.

3.3.2 The Second Tool: Weka Machine Learning

This study uses Weka's numerous Machine Learning algorithms to build models for identifying users for authentication. For the purposes of this study, Weka's algorithms for K-Nearest Neighbors, Multilayer Perceptron, and Naïve Bayes, Support Vector Machine (SVM) were used. The following paragraphs show a quick review of the product.

Weka is data mining software that provides a collection of Machine Learning algorithms that can be found at <http://www.cs.waikato.ac.nz/~ml/weka/> [118]. Weka provides a graphical user interface (GUI) or Java code that can be directly implemented in a software program. The paragraphs below show the basics of how to use the Weka Explorer GUI.

Weka is a library with a collection of Machine Learning algorithms. It is an open source software package developed at the University of Waikato, New Zealand, and it is available under the GNU General Public License. The most recent version of Weka is fully Java-based which makes it easily accessible to most users. Weka is mainly used for data preprocessing, clustering, classification, regression, visualization, and feature selection [16].

After installing and opening the Weka GUI, the first screen to display is the Weka GUI Chooser. This screen displays four applications that can be run. This study used the “Explorer” option from this window. Figure 3.7 below shows Weka’s Graphic User Interface.

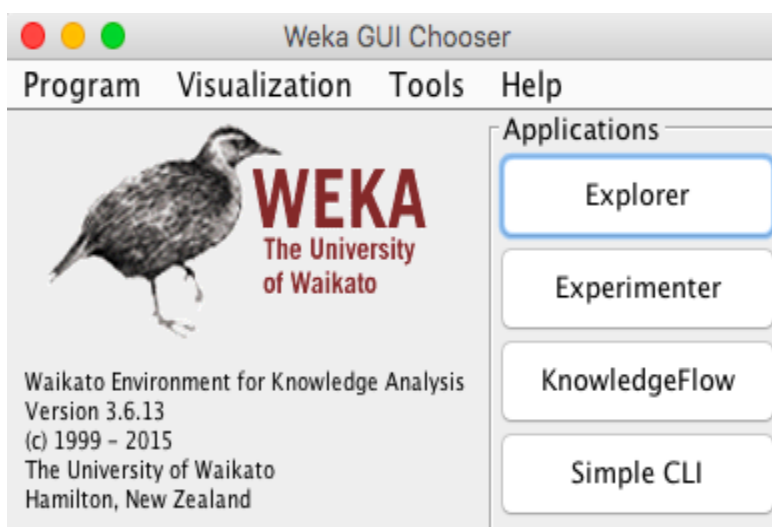


Figure 3.7 Weka: GUI Chooser User Interface

After selecting the Explorer option, the Weka Explorer window will be displayed with the Preprocessor tab selected. This tab is where the data file that will be used to train the classifier will be selected. Click the Open File... button and an Explorer window will open.

There is a drop-down menu where the type of file can be chosen. .ARFF and .CSV files can both be used in Weka. One can choose by selecting the appropriate file type and by clicking the “Choose” button. This study used the .CSV.

Once the file is chosen, there are options on this screen like adding filters and attributes to remove from the uploaded data. In addition, an analysis and visualization for each attribute is displayed.

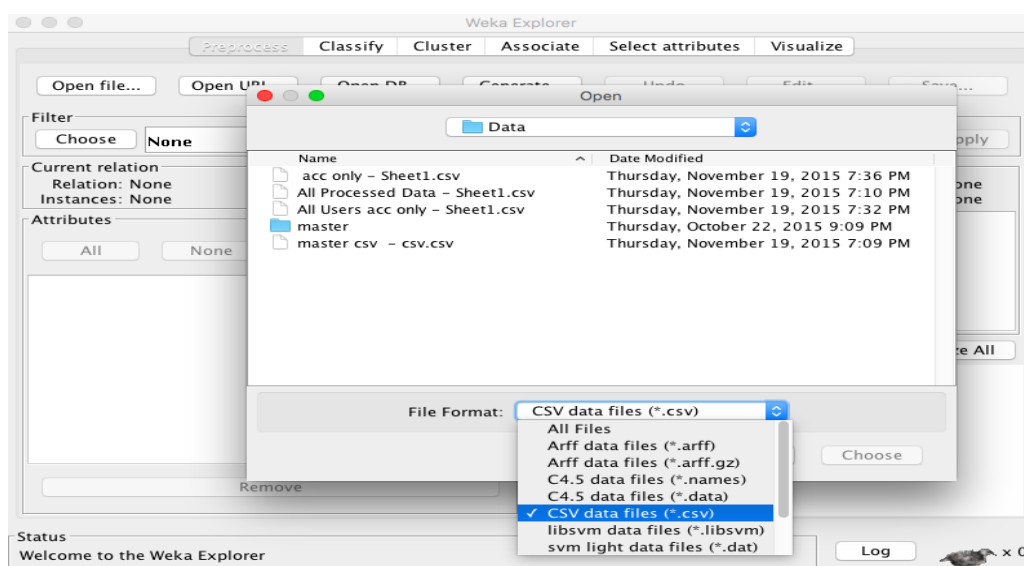


Figure 3.8 Weka: Selecting Data Type as Input to Weka

Figure 3.8 above shows how to select the file type to use as input to Weka.

The file uploaded should have one set of data per line. Each set of data should have all the attributes associated with that set and one class. Multiple sets of data can have the same class, and the class is used to determine the accuracy of the classification. Table 3.1 below shows eight

attributes with the classes of HW and NB. The classifier will then know which sets of data belong to HW and which belong to NB. This table was a snapshot of a file of the research conducted in fall 2015 for this study. The classes NB and HW are the initials of the two test participants. In the table, it shows 25 trials performed by the test participant with the initials HW. When the classifier was trained, the test data is classified as HW or NB based on the training data classes. The classification would be compared to the actual class provided to determine if the classification was correct.

Table 3.1 The Data Type within a CSV Format File


Q	R	S	T	U	V	W	X	Y
Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7	Attribute 8	CLASS
4.81044965	0.540020559	1.201333333	-0.547133333	1.47883	0.005752333	0.020424076	0.056063333	
0.888014887	0.23272642	1.074142857	0.601987143	2.478514286	0.001933143	0.355075561	0.375277217	hw
0.076584126	0.24169637	0.823	-0.46704	2.229426667	0.0013896	0.082406141	0.14884418	hw
1.906734689	0.555759089	0.692166667	-0.206598333	1.441718333	0.001544567	0.155773494	0.083549613	hw
0.39277306	0.059827429	1.436	0.309066667	2.628971667	0.00134	0.377792819	0.188859434	hw
1.864360472	0.51405156	0.982333333	0.470871667	1.446758333	0.001344267	0.139778233	0.129951511	hw
0.232300914	0.492244444	1.015833333	-0.468785	2.69162	0.001060567	0.081689256	0.072654092	hw
0.293654045	0.520812109	1.002285714	-0.012902857	1.994674286	0.001747905	0.150624812	0.304153385	hw
0.45141221	0.406276031	1.265333333	0.842404444	2.681158889	0.003011	0.130438051	0.149336417	hw
0.231872104	2.165636725	1.258625	0.93444875	1.8294925	0.002621696	1.31327574	0.782085257	hw
0.231872104	2.165636725	1.258625	0.93444875	1.8294925	0.002621696	1.31327574	0.782085257	hw
0.590088458	0.687989676	1.409888889	0.597943333	1.952391111	0.002836361	0.651635586	0.689219014	hw
1.070014436	1.100123696	0.932714286	-0.171418571	2.096861429	0.001933905	0.221262095	0.252175813	hw
0.33302736	0.509726935	1.049285714	-0.53863	2.25676	0.001729238	0.149304704	0.047699245	hw
0.679713187	0.599874352	1.263166667	0.363333333	2.42222	0.001338967	0.150317068	0.242656269	hw
2.123443829	2.224923136	1.074833333	0.390528333	2.273908333	0.001322167	0.197689683	0.145729644	hw
0.500649638	0.779730696	1.749333333	1.078234444	2.758867778	0.003126	0.078167634	0.2011431	hw
0.124781553	0.284103906	0.931714286	0.603882857	1.270725714	0.001938905	0.140638822	0.173489752	hw
0.073055414	0.187828401	1.019333333	1.389358333	1.039411667	0.001598667	0.191958554	1.793577323	hw
0.427942707	0.379724482	1.609375	-0.114865	3.06904125	0.002390839	0.759448385	0.235920747	hw
0.255334085	0.264525096	1.204428571	0.791078571	1.620831429	0.001754286	0.728045794	0.880283111	hw
0.267599879	0.128093863	0.897833333	0.483521667	1.344455	0.001434167	0.235414068	0.137134112	hw
0.636864133	0.766255176	0.831166667	0.696291667	1.649285	0.001649767	0.060928509	0.174392166	hw
0.544438351	0.957929512	0.900166667	1.50522	1.900563333	0.001345367	0.783527739	1.556395625	hw
0.315509591	0.342990524	0.9555	1.815338333	1.67589	0.0013955	0.125239444	0.108168612	hw
0.054510013	1.928170901	1.4334	-0.035516	8.683503	0.003890267	0.109948627	0.564865379	nb
0.078317679	0.228697649	0.680666667	-0.760686667	9.051627778	0.0032045	0.079085148	1.350898951	nb

Table 3.1 above shows the data types with a CSV-formatted file to be used as input to Weka.

Next, select the Classify tab. This tab classifier and test options are selected. The results of the test are also displayed in this tab.

First, click the Choose button under Classifier. This will display a list of the classifiers algorithms loaded into Weka. Naïve Bayes is listed under bayes, K-Nearest Neighbor is listed as IBk under Lazy, and Multilayer Perceptron is under Functions. Once the classifier is chosen, one can click Close.

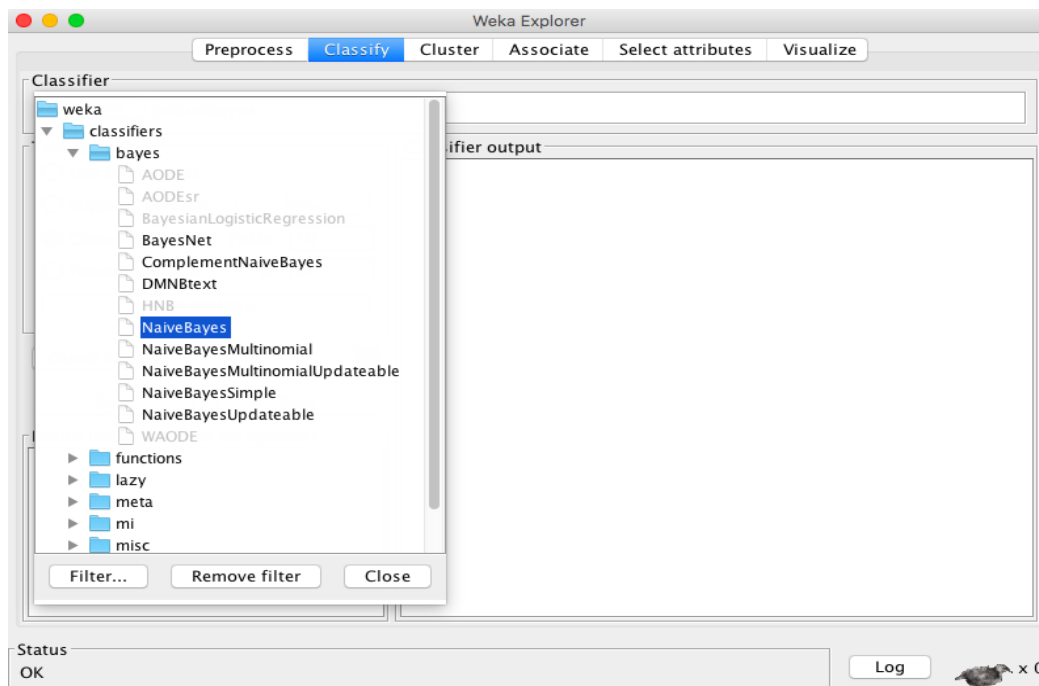


Figure 3.9 Weka: Selecting the Algorithm Type

Figure 3.9 above shows the different available algorithms in Weka.

After the classifier is chosen, one can choose one of the four test options. The Use training set option will use the uploaded file to train the classifier and then classify that same file. The Supplied test set gives the ability to upload another file with separate test data sets.

The Cross-validation option provides the ability to select the number of folds. The data was divided into that number of equal sections. Each section was used one time each as test data while the remaining sections were used as training data. Finally, Percentage provides the ability to select the percent of the data to be used for training while the rest of the data was tested. Once

the test option is chosen, click the Start button to begin the training and testing. The Results list will display the time of test and the classifier used. If multiple tests are run, this provides a test history and an ability to look at previous test results. In addition, the Classifier output frame will display a summary of the data and the classifications with the accuracy data. The confusion matrix displays how the data sets were classified in comparison to their actual class. The actual class with the key to the class type is displayed down the right side while the class that the set was classified as is displayed along the top. Figure 3.10 below shows the results of a run when Naïve Bayes was used in Weka.

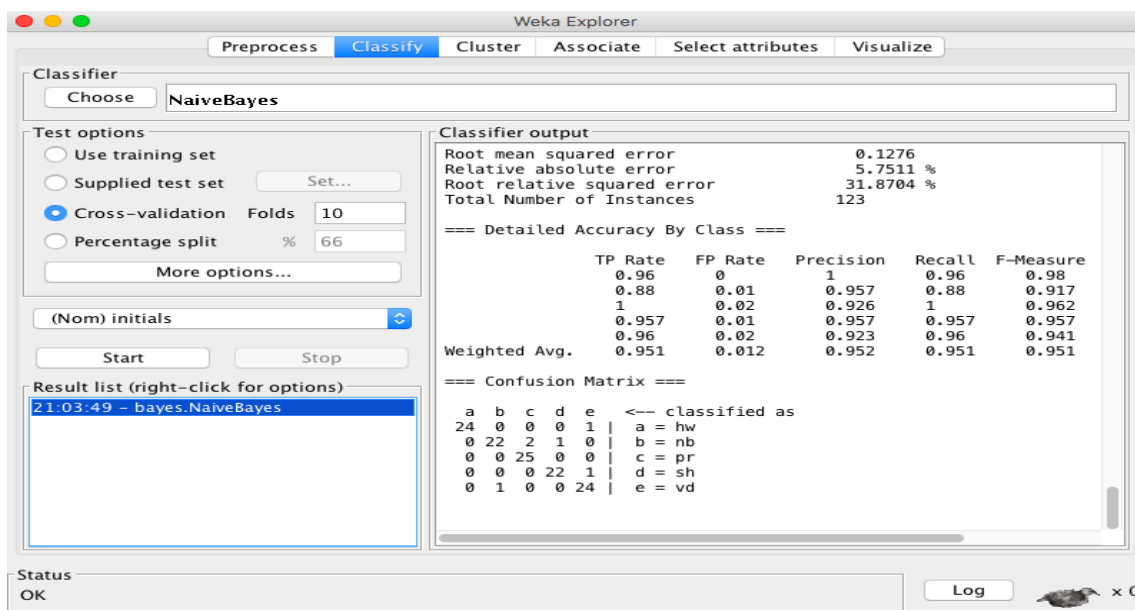


Figure 3.10 Sample Result of a Run Using Naïve Bayes (NB) Algorithm

3.4 Data Gathering Protocol – General for the Three Researches Performed

Behavioral biometrics needs a larger set of training data for the phone to learn users' unique use patterns, in comparison to physiological biometrics capture, such as fingerprinting and iris

scanning. Unlike physiological biometrics capture, behavioral biometrics requires a larger set of training data for the phone to learn the users' unique use patterns.

In the research conducted in spring 2016, there were six test coordinators and each had an Android-based smartphone. Each test coordinator researched with his/her test participants using the phone that he/she had. The list below shows the phones and models used for the spring 2016 research:

1. LG Nexus 5
2. Motorola Moto G (3rd Gen)
3. HTC One m8
4. Samsung Galaxy Note 4 (2 testers)
5. Samsung Galaxy Note 5

Every test coordinator gathered data from 10 participants, who performed 20 trials each, for a total of 200 runs per tester or 1200 runs in total. It is worth noting that the "Motorola Moto G" did not have a gyroscope sensor; therefore, it only recorded data from its accelerometer sensor. The tester coordinators facilitated by running the "Sensor Kinetics Pro" application, by preparing the Android phone for capture, by observing while the test participants motioned the phone in accordance with the test protocol, and by preparing the device for the next round of capture. While various motions were considered for this study, the one chosen was deemed best because they comes naturally to most people and did not feel artificial. The motions were to lift the phone to view, pause, and then move the phone to ear level. No time limit was set for the test participants. The only time enforcement involved was the 2-3 second pauses in between motions.

3.4.1 Data Gathering Protocol in spring 2016

The trial procedure was straightforward. The participant sat at or stood by a table or desk with the phone face up on the surface of the table/desk. The participant pressed the Start button while the phone was still lying flat, and then lifted it up to view. After a 2-3 second pause the participant raised the phone with his or her hand of choice to his or her ear. The test participant then pressed the Stop button at the ear after another 2-3 second pause to finish that trial.

Operating a 10-fold cross validation test (a parameter on the Weka Machine Learning tool), 90% of those trial runs worked as the training data, while the other 10% were used as the testing data which the algorithm was required to perform against. This procedure then repeated itself with a different tenth serving as the testing data.

The tester then stopped the motion capture of the device, and prepared for the next trial run until all the trials were completed. Table 3.2 below shows a sample image of the data captured.

3.4.2 Data Gathering Protocol in fall 2015

The test coordinators assisted by running the “Sensor Kinetics Pro” application, prepping the device for capture, standing back while the test participant motioned the phone in accordance with the test protocol, stepping in to stop the motion capture when the motion was completed, and preparing the device for the next round of capture.

Upon positive cue from the test coordinator, the standing participant performed the following actions in fall 2015:

1. He/she raised the phone from waist height to view the identity of the caller.
2. He/she reviewed the screen for a moment, gently shook the phone with a simple flick of the wrist. This helped to signify that one motion had ended and another began.

3. Finally, he/she raised the phone the rest of the way to their ear.
4. A fourth motion was also captured, as the device was removed downwards from the ear to end the motion capture.

The test coordinator then stopped the motion capture of the device, and prepped for another trial run as needed. Sample images of these trials are represented in Table 3.2 and Figure 3.11 below.

Table 3.2 A CSV-formatted Accelerometer Data from the App. (3-axis)

time	X_value	Y_value	Z_value
0.000	0.00000	0.00000	0.00000
0.060	0.20000	0.64000	10.28000
0.130	-0.05000	0.64000	9.94000
0.199	0.06000	0.62000	10.18000
0.259	0.05000	0.66000	10.18000
0.329	0.06000	0.63000	10.14000
0.400	0.03000	0.68000	10.14000
0.464	0.03000	0.64000	10.12000
0.529	0.03000	0.64000	10.11000
0.590	0.01000	0.67000	10.13000

Figure 3.11 below shows a simple graph of the motions with manual feature extraction highlighted.

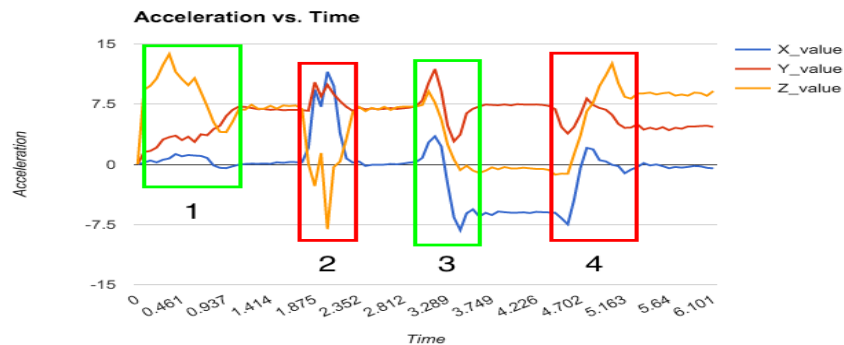


Figure 3.11 Simple Method: Graph Motions - Manual Feature Extraction Highlighted

3.5 Definition of Arbitrary Motions Using a Smartphone

In all the research conducted in spring 2015 [112], fall 2016 [113], and spring 2016 [113], users used arbitrary motions to lift or pick up a phone, bring it to a view position, and move it to ear level. The paper will detail the methods later, but in general the motions and moves are shown below in Figure 3.12.



Figure 3.12 Arbitrary Motions in the fall 2015 Research

3.6 Details on the Motions and Data Collection Steps

While there were small variations between the three researches conducted, in general, the following steps were followed to collect data:

- First, the phone was laid flat on a table (or kept at waist level); and then the test coordinator to test participant started the “Sensor Kinetic Pro” App to begin capturing the data from the motion.



Figure 3.13 Test: Process Started Smartphone on a Table

Figure 3.13 above shows the starting point. A phone is placed flat on a table. Figure 3.14 shows the homepage of Sensor Kinetic Pro app.

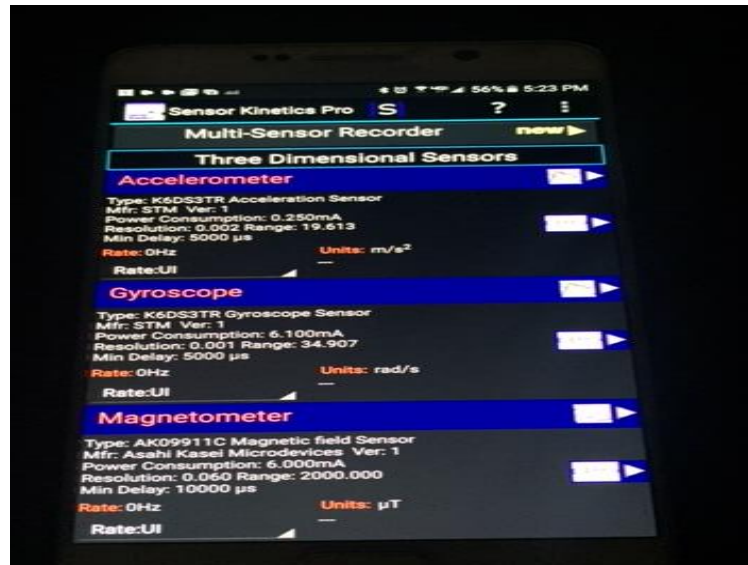


Figure 3.14 Test: Sensor Kinetic Pro: App Home Page

- Second, the test participant starts the motions. The first part of the motion is raising the phone from the starting position to about halfway between the starting position and the participant's face.



Figure 3.15 Test: First Motion - Bring the Phone to View (First Long Motion)

- Third, the next part of the motion is moving the phone to view and wait for couple of seconds in that position as shown in graph 3.16.



Figure 3.16 Test: Second Motion – Pause While Viewing

- Fourth, the next step, as shown in graph 3.17, is to have the participant raise the phone all the way to their ear and wait for couple of seconds.



Figure 3.17 Test: The Second Long Motion: Move the Phone to the Ear Level

- Finally, while the phone is at ear position, stop the recording, save the data, clear the chart, and prepare for the next trial.

In all, there were four motions: two small and two large motions. The first motion was small, just to pick up the phone from the table; the next was a big motion of moving the phone to the face/view position; the third move was another big motion of moving the phone to the ear, and the final position was to move a little to press the Stop button to stop recording.

3.7 Complete Steps of Capture and Process of the Motion Data:

The complete process of the work from the start of the app to the final execution of the data for this experiment is shown below in Figure 3.18:



Figure 3.18 Process: Complete Capture and Evaluation of the Data from Motions

3.8 Data Storage

All the trial data were captured on Android-based phones, and after converted to “.CSV” formatted file, were uploaded onto to a group Google Drive folder for the spring 2016 research. The folder had the name of each test coordinator. Each test coordinator was responsible for ten test participants and thus there were ten folders per test coordinator, numbered accordingly. The folders 0-9 belonged to one test coordinator, 10-19 to another test coordinator, and so on. There were six test coordinators and each had ten participants for 60 test participants in total. As noted,

the data from each participant was saved in a folder belonging to a test coordinator. Figure 3.19 below shows the screen capture of the Google Drive.

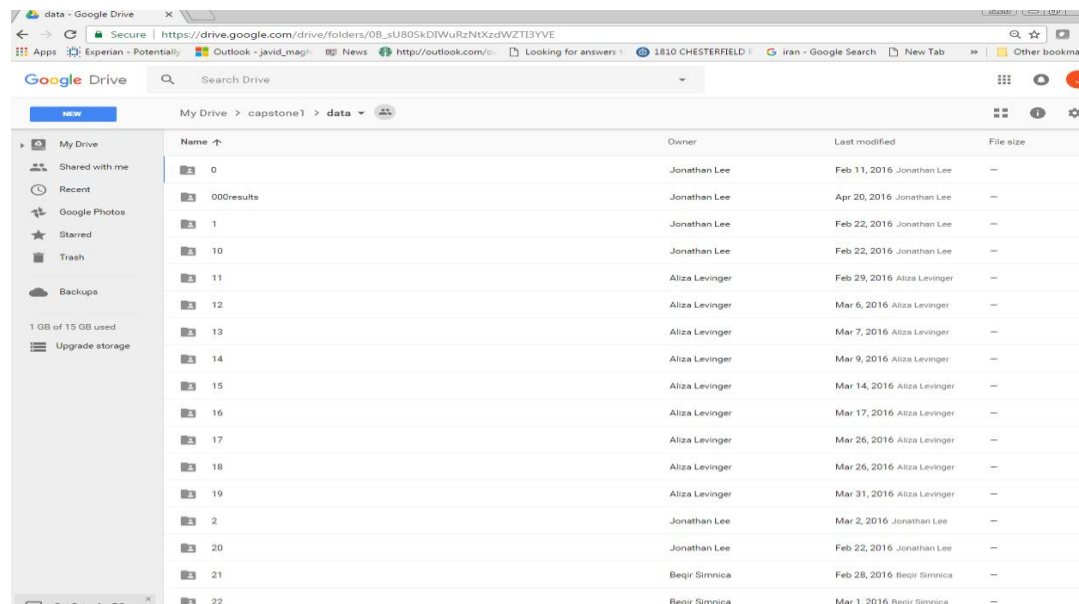


Figure 3.19 Data Storage: View of the Folders on Google Drive

The folders were numbered 0 to 62. There were some invalid tests, which was why there were three extra folders. As noted before, there were ten folders for a given test coordinator to store his or her participants' data, and as mentioned earlier, every folder contained the trial data from each participant for that test coordinator. Below are the files of each trial for folder one. File names started with letters "chart1" through "chart 20" since each participant tested 20 times. It contained data for both accelerometer and gyroscope, if the phone was equipped with both sensors; the file names would be like "chart xx_acc.csv from SensorKineticsPro" if the data was from accelerometer sensor and "chart xx_gyr.csv from SensorKineticsPro" if the data was from the gyroscope sensor of that phone. If the phone was only equipped with one sensor (only

accelerometer), the name would then had the name with ‘*_acc*’ to show the data was from the accelerometer sensor.

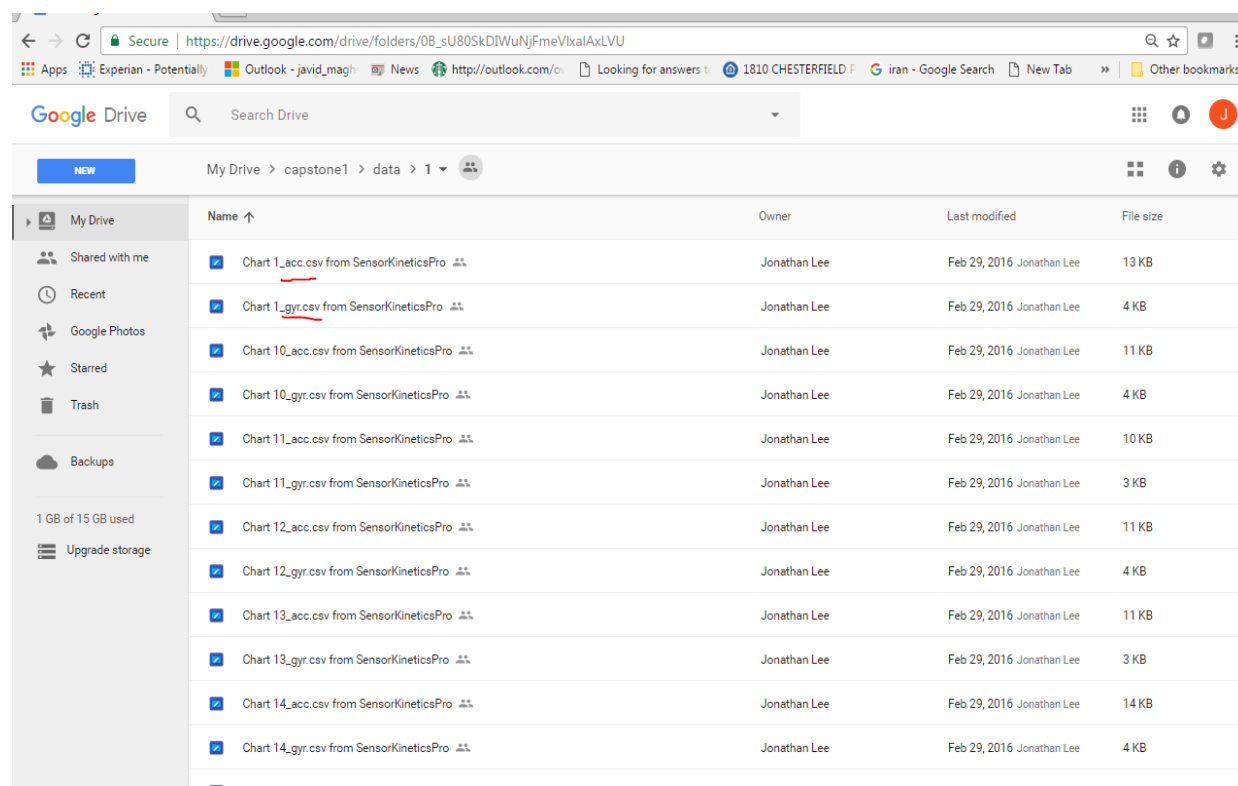


Figure 3.20 Data Storage: Accelerometer and Gyroscope Data - Participant’s Trials

Figure 3.20 above shows the files created from participants’ different trials. The file types were of accelerometer or gyroscope types.

These folders also were synchronized locally on a user's drive as shown below on a Windows explorer.

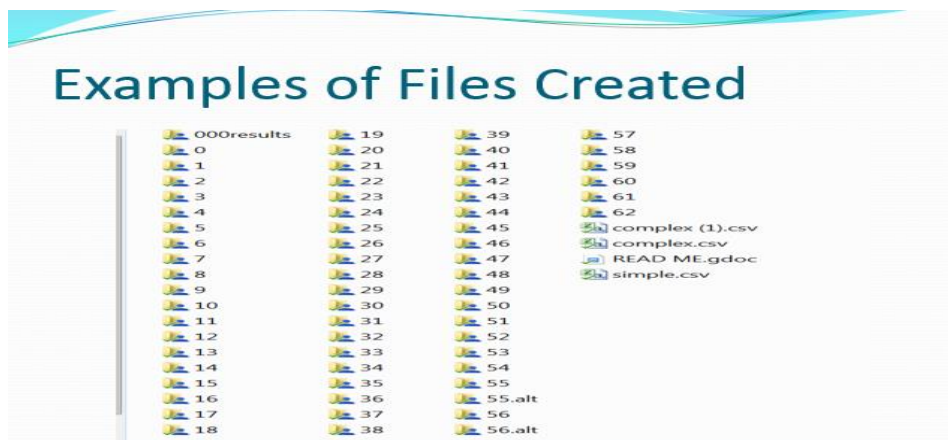


Figure 3.21 Data Storage: Google Drive Files Locally

Figure 3.21 above shows files at local level synchronized with the files on Google Drive.

Below is an example of data collections and contents of a directory. For every trial, as noted earlier, when applicable for both sensors, there was a file name with “chart_nn_sensor_type_format_type” from the source. For example, “chart_1_acc_csv_from SensorKineticPro” meant that it was a participant's first trial, sensor type was accelerometer and data type format was of “.CSV” type, and the data was captured from the “Sensor Kinetic Pro” App. With the same definition, “char 15_gyr_csv_from Kinetic Sensor Pro” meant this file was created on participant's fifteenth (15th) trial, was of sensor type gyroscope, data type was of “.CSV” format and the data was from “Sensor Kinetic Pro” app.

Example of Data Collections and Contents of a Directory

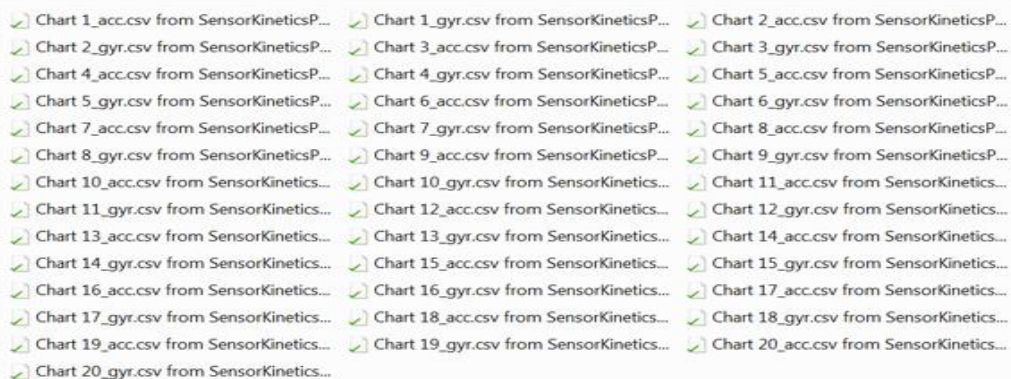


Figure 3.22 Data Storage: Contents of a Directory with Data from the Sensors

Figure 3.22 shows the contents of each file. As shown above, each participant tried 20 times.

To repeat again, for each trial, one file per sensor (two when both were available) were created.

These files, in .CSV formats, were saved locally on a smartphone and were transferred to the team's Google Drive. Figure 3.23 below shows a sample of .CSV-formatted data for a gyroscope:

Sample of *.CSV Data for Gyroscope

time	X_value	Y_value	Z_value
0	0	0	0
0.03	-0.00098	0.01067	0.00087
0.093	-0.00018	0.00172	0.00015
0.16	0.00174	-0.00048	0.00067
0.223	0.00134	-0.00131	-0.00044
0.288	0.00023	-0.00058	-0.00064
0.353	0.00201	0.00053	0.0013
0.419	0.00122	-0.00137	0.0005
0.481	0.0022	-0.0015	-0.00061
0.611	-0.05931	0.24814	0.00958
0.676	-0.01077	0.02528	0.25119
0.739	-0.1053	1.16769	0.22839
0.805	0.20262	5.67775	-1.53311
0.869	-0.42592	0.42848	0.26974
0.935	-1.16797	-0.76234	0.08032
0.997	1.15111	-5.07042	0.03064
1.063	1.63174	-5.43347	0.45956

Figure 3.23 Data Storage: Sample *.CSV Data from Gyroscope

The contents of each file from each participant were then put together and averages and variances were taken. Figure 3.24 below is a screenshot that shows the data summarized of all participants for both sensors (when applicable) plus the statistical methods of AVG (Average) and VAR (Variance) that were calculated and all were collectively kept in a .CSV-formatted file.

The screenshot shows a CSV file with columns labeled A through GL. The data is organized into two main sections. The first section contains columns A through J, and the second section contains columns GE through GL. Each row represents a subject, with the subject ID listed in the 'subject_number' column. The data values are numerical, representing averages and variances for different sensors.

Figure 3.24 Data Storage: The Aggregated Data with Average and Variance

This file was subsequently used as an input to the Weka Machine Learning tool. Below is an example of how Weka Machine Learning (ML) tool displays the results:

The screenshot displays the Weka Explorer interface. The 'Classifier' tab is selected, and the 'NaiveBayes' classifier is chosen. The 'Test options' section shows 'Cross-validation' with 'Folds 10' and 'Percentage split % 66'. The 'Classifier output' section shows the following results:

```

Scheme:weka.classifiers.lazy.NB1
Relation: complete (3)
Instances: 1152
Attributes: 193
[Test model:10-fold cross-validation]
Time taken to build model: 0.01 seconds

==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances 1035 89.843%
Incorrectly Classified Instances 117 10.157%
Kappa statistic 0.8967
Mean absolute error 0.0094
Root mean squared error 0.0962
Relative absolute error 10.3305 %
Root relative squared error 45.4543 %
Total Number of Instances 1152
    
```

The accuracy rate of 89.843% is highlighted with a red circle.

Figure 3.25 Weka: A Sample Execution with Accuracy Rate

Figure 3.25 above shows the output of a run under Weka that shows the accuracy rate.

3.8 Feature Extractions, Data Processing & Machine Learning Algorithms

As noted before, three separate research studies were conducted and three separate papers were produced accordingly in three semesters. All were the foundations of this dissertation. In the first two research studies done in spring 2015 and fall 2016, the research used manual feature extractions. However, the feature extraction was automated in the final research in spring 2016.

The following paragraphs below describe the manual feature extraction, the fully automated data processing, and the algorithms used.

3.8.1 Manual Feature Extraction

For the manual feature extraction, the following steps were performed:

1. Using a graph of the accelerometer data, the motions were visually inspected to isolate the times a given motion began and ended;
2. With the relevant “timestamps” marking the “beginning” and “end” of a given motion, the motions were broken into quartiles;
3. Depending on whether the accelerometer and gyroscope captured data at the same sampling rate (the Nexus 5 did not), the time indices for the gyroscope were adjusted to match the timestamps identified for the accelerometer. More accurately, the row numbers were used, with the assumption that the sampling occurred at relatively constant rates;

4. The quartiles per motion were then averaged ($\{x, y, z\}$ (separately)) and had a variance taken ($\{x, y, z\}$ (separately)), so that each motion had 24 attributes for the accelerometer, and 24 attributes for the gyroscope, if gyroscope sensor was present. Each trial consisted of two major motions, and thus consisted of 96 total data points (again, 48 for the “Motorola G 3rd Gen” phone, which lacked a gyroscope sensor).

The use of quartiles was intended to recapture individual idiosyncrasies, while making the data amenable to the Weka’s numerous classification algorithms. Dividing the data into smaller partitions would possibly have created samples with too few data points, resulting in meaningless variances.

3.8.2 Automated Feature Extractions

A separate processing of the data was performed that was entirely automated using a Java program. This Java program accepted the “.CSV”-formatted files with user motion data, and then partitioned the data stream for the entire motion into eight segments. Eight segments were selected to maintain parity with the two sets of quartiles from the manual feature extraction. As above, the data points in the eighths were averaged and had their variances calculated in the X, Y and Z-axes separately.

For both the manual data extraction and the automated version, the information was aggregated into .CSV files that were plugged into Weka, and each 96-data-point row was post-pended with an identifier ... the test participant’s initials.

Once the data was appropriately captured and pre-processed, the following Machine Learning algorithms were applied using Weka: First, the Multilayer Perceptron was used. Second, the K-Nearest Neighbors (k-NN) algorithm was employed because it was a benchmark

in addition to being fast and simple to run. Third, the Naïve Bayes (N-B) was selected because it was a well-known and widely used algorithm. These three algorithms were used in spring 2015 and fall 2015 research studies. Support Vector Machine (SVM), also a widely used algorithm, was the additional algorithm used in the final research study in spring 2016.

A Multilayer Perceptron (MLP) is a feedforward artificial neural network. Patterns are presented via the input layer that communicates with hidden layers where the actual processing is done via the system of weighted connections. The hidden layers then link to an output layer.

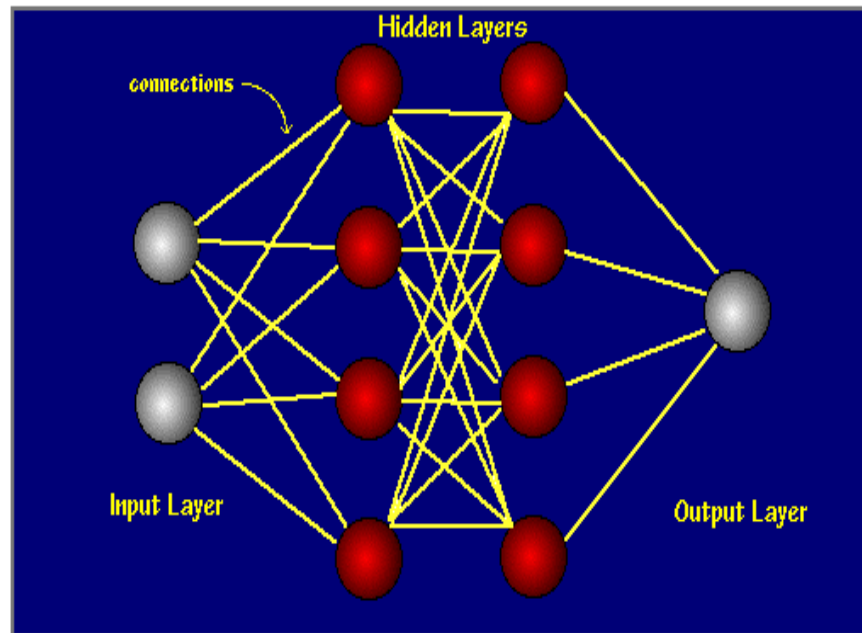


Figure 3.26 Multilayer Perceptron: Input, Output, and Hidden Layers [13]

Figure 3.26 above shows Multilayer Perceptron with input, output, and hidden layers.

The network is trained using backpropagation—a supervised process that occurs with each cycle or “epoch” (i.e. each time the network is presented with a new input pattern). When a neural network is initially presented with a pattern, it makes a random “guess” as to its

classification. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights.

The K-Nearest-Neighbor (k-NN) algorithm assigns an unknown object to the class most common among the unknown K's nearest neighbors in feature space. A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data.

The Naïve Bayes classifier is a simplification of Bayes' decision method that assumes the features are independent of each other. Naïve Bayes algorithms are quite successful at resolving real world classification problems.

3.8.3 Manual Feature Extraction & Automated (spring/fall 2015 Research)

Upon downloading the .CSV files from "Sensor Kinetics Pro", the data may appear as follows in Table 3.3:

Table 3.3 Data extracted from Sensor in CSV-Formatted Form

time	X_value	Y_value	Z_value
0.000	0.00000	0.00000	0.00000
0.060	0.20000	0.64000	10.28000
0.130	-0.05000	0.64000	9.94000
0.199	0.06000	0.62000	10.18000
0.259	0.05000	0.66000	10.18000
0.329	0.06000	0.63000	10.14000
0.400	0.03000	0.68000	10.14000
0.464	0.03000	0.64000	10.12000
0.529	0.03000	0.64000	10.11000
0.590	0.01000	0.67000	10.13000

For each trial run, a *.CSV file is generated separately for the accelerometer and gyroscope.

Graphed, this should appear as follows (showing the accelerometer data):

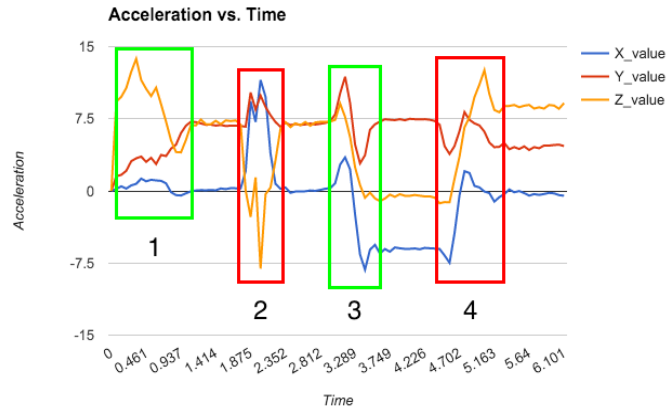


Figure 3.27 Acceleration vs. Time

Upon visually inspecting the data, the beginning and end-points of a motion of interest must be identified. In the case of this study, motions 1 (raising the phone to visual level) and 3 (raising the phone to the ear) are specified. For the purposes of the experiment, long pauses are enforced between the motions to show peaks and valleys; however, in life such long pauses would not be present. A specially filled out excel spreadsheet is shown below in Figure 3.28, which shows the calculations. This work was done during the research of spring 2015 and fall 2015.

D	E	F	G	H	I	J	K	L	M	N
0.10484	12.04347								0.891	-0.22633
-0.3405	9.49886								0.955	0.03215
0.48311	7.16609								1.022	0.02681
0.80446	6.73048								1.083	-0.09709
1.08772	9.82259								1.148	-0.07359
0.09511	12.98848	<< acc data	A. Samples			G. Samples		gyr data>>	1.212	-0.35985
0.30479	12.66475	=COUNT(B2:B999)				=COUNT(M2:M999)			1.276	1.002
0.02629	9.83687	Peak1			Valley1				1.34	0.87383
0.48311	6.78285	Peak2			Valley2				1.406	1.42284
0.89491	5.6617		53	=ROUND(G25*\$J\$22/\$G\$22)		167		=ROUND(I25*\$J\$22/\$G\$22)	1.469	1.1227
0.74495	7.63503		93	=ROUND(G26*\$J\$22/\$G\$22)		202		=ROUND(I26*\$J\$22/\$G\$22)	1.533	0.69865
0.65688	10.90089		=G25	=H25	=I25	=J25			1.598	0.87917
0.29744	13.53835		=ROUND((G30+G25)/2)	=ROUND((H30+H25)/2)	=ROUND((I30+I25)/2)	=ROUND((J30+J25)/2)			1.663	0.28316
0.27602	11.31032		=G28+1	=H28+1	=I28+1	=J28+1			1.727	0.35472
0.23317	7.84212		=ROUND((G25+G26)/2)	=ROUND((H25+H26)/2)	=ROUND((I25+I26)/2)	=ROUND((J25+J26)/2)			1.79	-0.49016
0.09987	7.90401		=G30+1	=H30+1	=I30+1	=J30+1			1.855	0.08769
0.14986	8.70381		=ROUND((G26+G30)/2)	=ROUND((H26+H30)/2)	=ROUND((I26+I30)/2)	=ROUND((J26+J30)/2)			1.92	-0.21886
0.40932	9.51076		=G32+1	=H32+1	=I32+1	=J32+1			1.983	0.15605
0.5688	11.73878		=G28	=H28	=I28	=J28			2.048	0.20091
0.60451	11.28889								2.113	0.0407
0.15462	9.6274		acc-x-avg-1-1	=AVERAGE(INDIRECT("r"&\$G\$27.1);INDIRECT("r"&\$G\$28.1))	gyr-x-avg-1-1	=AVERAGE(INDIRECT("r"&\$H\$27.1);INDIRECT("r"&\$H\$28.1))			2.177	0.25005
0.18318	8.23012		acc-y-avg-1-1	=AVERAGE(INDIRECT("c"&\$G\$27.1);INDIRECT("c"&\$G\$28.1))	gyr-y-avg-1-1	=AVERAGE(INDIRECT("c"&\$H\$27.1);INDIRECT("c"&\$H\$28.1))			2.24	0.05031
0.157	7.87544		acc-z-avg-1-1	=AVERAGE(INDIRECT("r"&\$G\$27.1);INDIRECT("r"&\$G\$28.1))	gyr-z-avg-1-1	=AVERAGE(INDIRECT("r"&\$H\$27.1);INDIRECT("r"&\$H\$28.1))			2.305	0.03535
0.49739	8.90138		acc-x-var-1-1	=VAR(INDIRECT("r"&\$G\$27.1);INDIRECT("r"&\$G\$28.1))	gyr-x-var-1-1	=VAR(INDIRECT("r"&\$H\$27.1);INDIRECT("r"&\$H\$28.1))			2.37	0.30879
0.52357	11.3484		acc-y-var-1-1	=VAR(INDIRECT("c"&\$G\$27.1);INDIRECT("c"&\$G\$28.1))	gyr-y-var-1-1	=VAR(INDIRECT("c"&\$H\$27.1);INDIRECT("c"&\$H\$28.1))			2.435	-0.09923
0.06676	11.48409		acc-z-var-1-1	=VAR(INDIRECT("r"&\$G\$27.1);INDIRECT("r"&\$G\$28.1))	gyr-z-var-1-1	=VAR(INDIRECT("r"&\$H\$27.1);INDIRECT("r"&\$H\$28.1))			2.498	0.03642
0.38335	9.70119		acc-x-avg-1-2	=AVERAGE(INDIRECT("r"&\$G\$29.1);INDIRECT("r"&\$G\$30.1))	gyr-x-avg-1-2	=AVERAGE(INDIRECT("r"&\$H\$29.1);INDIRECT("r"&\$H\$30.1))			2.562	-0.05009
0.05009	8.68477		acc-y-avg-1-2	=AVERAGE(INDIRECT("c"&\$G\$29.1);INDIRECT("c"&\$G\$30.1))	gyr-y-avg-1-2	=AVERAGE(INDIRECT("c"&\$H\$29.1);INDIRECT("c"&\$H\$30.1))			2.627	1.42604
0.34505	8.89186		acc-z-avg-1-2	=AVERAGE(INDIRECT("r"&\$G\$29.1);INDIRECT("r"&\$G\$30.1))	gyr-z-avg-1-2	=AVERAGE(INDIRECT("r"&\$H\$29.1);INDIRECT("r"&\$H\$30.1))			2.692	0.10158
0.52357	9.38936		acc-x-var-1-2	=VAR(INDIRECT("r"&\$G\$29.1);INDIRECT("r"&\$G\$30.1))	gyr-x-var-1-2	=VAR(INDIRECT("r"&\$H\$29.1);INDIRECT("r"&\$H\$30.1))			2.754	-0.76146
0.54738	9.67024		acc-y-var-1-2	=VAR(INDIRECT("c"&\$G\$29.1);INDIRECT("c"&\$G\$30.1))	gyr-y-var-1-2	=VAR(INDIRECT("c"&\$H\$29.1);INDIRECT("c"&\$H\$30.1))			2.819	-4.46782
0.1927	9.96065		acc-z-var-1-2	=VAR(INDIRECT("r"&\$G\$29.1);INDIRECT("r"&\$G\$30.1))	gyr-z-var-1-2	=VAR(INDIRECT("r"&\$H\$29.1);INDIRECT("r"&\$H\$30.1))			2.884	0.53737
-0.231	10.24629		acc-x-avg-1-3	=AVERAGE(INDIRECT("r"&\$G\$31.1);INDIRECT("r"&\$G\$32.1))	gyr-x-avg-1-3	=AVERAGE(INDIRECT("r"&\$H\$31.1);INDIRECT("r"&\$H\$32.1))			2.948	0.81294
0.38811	10.29152		acc-y-avg-1-3	=AVERAGE(INDIRECT("c"&\$G\$31.1);INDIRECT("c"&\$G\$32.1))	gyr-y-avg-1-3	=AVERAGE(INDIRECT("c"&\$H\$31.1);INDIRECT("c"&\$H\$32.1))			3.012	-0.52968
0.12865	9.3727		acc-z-avg-1-3	=AVERAGE(INDIRECT("r"&\$G\$31.1);INDIRECT("r"&\$G\$32.1))	gyr-z-avg-1-3	=AVERAGE(INDIRECT("r"&\$H\$31.1);INDIRECT("r"&\$H\$32.1))			3.08	-0.1035
0.0118	8.91566		acc-x-var-1-3	=VAR(INDIRECT("r"&\$G\$31.1);INDIRECT("r"&\$G\$32.1))	gyr-x-var-1-3	=VAR(INDIRECT("r"&\$H\$31.1);INDIRECT("r"&\$H\$32.1))			3.15	-0.64931
0.0594	9.74403		acc-y-var-1-3	=VAR(INDIRECT("c"&\$G\$31.1);INDIRECT("c"&\$G\$32.1))	gyr-y-var-1-3	=VAR(INDIRECT("c"&\$H\$31.1);INDIRECT("c"&\$H\$32.1))			3.21	0.48289
0.12865	11.07704		acc-z-var-1-3	=VAR(INDIRECT("r"&\$G\$31.1);INDIRECT("r"&\$G\$32.1))	gyr-z-var-1-3	=VAR(INDIRECT("r"&\$H\$31.1);INDIRECT("r"&\$H\$32.1))			3.27	-0.18895
0.21434	10.58907		acc-x-avg-1-4	=AVERAGE(INDIRECT("r"&\$G\$33.1);INDIRECT("r"&\$G\$34.1))	gyr-x-avg-1-4	=AVERAGE(INDIRECT("r"&\$H\$33.1);INDIRECT("r"&\$H\$34.1))			3.335	0.66768
0.12843	9.21559		acc-y-avg-1-4	=AVERAGE(INDIRECT("c"&\$G\$33.1);INDIRECT("c"&\$G\$34.1))	gyr-y-avg-1-4	=AVERAGE(INDIRECT("c"&\$H\$33.1);INDIRECT("c"&\$H\$34.1))			3.404	2.52406

Figure 3.28 Data: Raw Motion Data

Figure 3.28 above shows the beginning and end-points of the motions outlined in the four boxes as saved in an Excel worksheet.

The spreadsheet does the rest. The motions are quartered: First, it finds the midpoint between the start and end-point (rounding down for the end of the second quarter's end, and rounding up for the third quarter's beginning). Likewise, it finds the quarter cut points to distinguish the first quarter from the second, and the second from the fourth.

Roughly speaking, it could be designed as:

start

$$\begin{aligned} &(\text{start}+\text{end})/4 \\ &(\text{start}+\text{end})/2 \\ &3*(\text{start}+\text{end})/4 \end{aligned}$$

end

To adjust for differing sample rates between accelerometer and gyroscope, all the cut-points distinguishing the quarters are simply multiplied by

“gyrometer_samples/accelerometer_samples”. Then, the average and variance were calculated for each quarter, in each axis, for each motion, and for each sensor. In total, this should generate 96 data points for each trial run. The spreadsheet pulls this all to the front page (which is already pre-labelled). A person’s individual 25 runs (for spring and fall 2015 researches) also had their initials appended as the 97th data point.

3.8.4 Automated Data Processing

During the spring 2016 research, the study used an automated script in Java (shown in Appendix B) that performed the steps of breaking the motion into eighths and generated the averages and variances in the X, Y and Z-axes. After the Java program was run, two .CSV-formatted files were generated in the same directory and could be combined as needed, for example, with each phone, or altogether. One is called “Simple” and the other called “Complex”; this will be explained later in chapter 4.

3.8.5 The Java Program

This research study used a Java program to read the sensors’ data and to create the appropriate output—it is shown in Appendix B.

3.10 Chapter Conclusion

This chapter described the three different research studies conducted to support the study, and it showed how this study required different and integrated steps, tools, and methods to lay the foundations and infrastructure to get the required data and results.

Chapter 4

Experiments and Results

4.1 Introduction

To conduct the study and prepare for this dissertation, three separate research studies were conducted over three different semesters. This chapter will first provide a quick review of the first research study conducted (in spring 2015) and the second research study conducted (in fall 2015) to show the history and the supporting work that were done. This chapter, for most part, will focus on the research conducted in spring 2016 that was, by far, the largest and the most extensive among the three research studies performed.

4.2 Chapter Organization

This chapter will have the following sections:

- **Timeline of the research:** In this section, the overall research timelines and number of research studies performed will be described.
- **The research study in spring 2015:** In this section, the detail of the research conducted in spring 2015 will be described. This research was mainly conducted to establish a feasible study.
- **The research study in fall 2015:** In this section, the details of the research conducted in fall 2015 will be described. This research was to build upon the first research study done in spring 2015. In this study, the number of participants was expanded to 20 from 10, the number of phones used was increased to three from one, and the number of employed Machine Learning

algorithms was increased to three from one. The research in fall 2015 [113] was expanded overall to build on the proof of concept established in the prior research in spring 2015 [112].

- The research study in spring 2016 [113]: In this section, the details of the research conducted in spring 2016 will be described. This study built upon the prior two studies to increase the confidence about the results of the research by making it more expansive. The results of the research in spring 2016 were used to write a paper and to present in an international conference on biometrics in August of 2016. The conference was the Intelligence and Security Informatics Conference (EISIC) [114] held at Uppsala University in Sweden. In this study, the number of participants was increased to 60, the total phones used were increased to six (two participants used the same type and model), and four Machine Learning algorithms were employed. In spring 2016, the research conducted additional experiments such as different sample sizes (e.g., five random sets of five, ten, and twenty datasets) to measure if accuracy rates would change when Machine Learning algorithms processed different sample sizes. The research also experimented with the effect of using different statistical methods. While the main part of the study had utilized both the Average and Variance statistical methods, the additional experiments examined the accuracy rate if the machine learning algorithms had to process only one of the statistical methods—either the Average or just the Variance. This would allow the research to optimize and select one statistical method over the other in the future. In these expanded experiments, the study used Naïve-Bayes (N-B)

and K-Nearest Neighbor (k-NN) algorithms to measure the effectiveness of using either the Average or Variance methods. The study benefited immensely through expanding the research in spring 2016 and conducting these additional experiments. The study returned valuable information.

4.3 Timeline of the Study and Related Researches

Below shows a quick timeline and specific profiles:

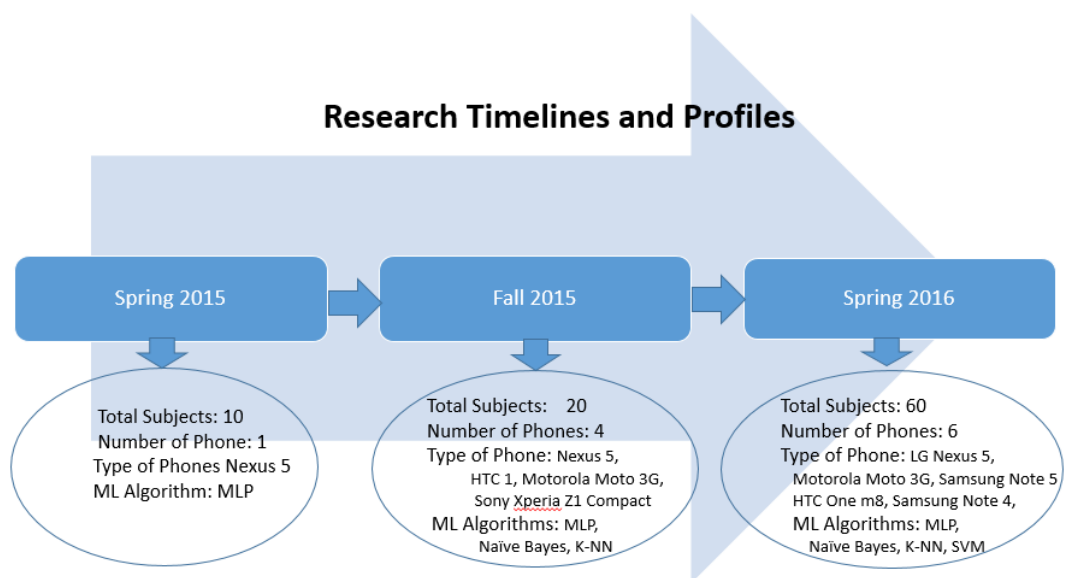


Figure 4.1 Research Timeline

The figure above shows the timeline for the research that spanned three semesters.

4.4 Research Study in spring 2015

The initial research began in spring of 2015 to establish the proof of concept. The study used the same mobile app as the other two to capture data from the motion and the movement of the phones. The app was an Android-based application called “Sensor Kinetics Pro” produced by INNOVENTIONS, Inc., and the phone was “Nexus 5”, an Android-based phone. The phone was

equipped with both the gyroscope and accelerometer sensors. During this experiment, there were ten volunteers (five male, five female) all in the age group of 19-25.

Each participant simulated a phone conversation, and he/she was asked to repeat the process five times. To have a controlled environment, the smartphone was placed face up flat on the table. Each participant would pick up the phone with either hand depending on his/her preference, hold the phone to his or her ear for five seconds and return the phone to the table. The test used the app's built-in accelerometer and gyroscope sensors to record the exact positioning and rotation of each axis of the device. The app would collect data during these movements. The most important moves were picking up the phone to look and then move the phone to the ear.

The results were all saved as .CSV-formatted files and later were input into the Weka Machine Learning (ML) tool to be examined under its Multilayers Perceptron (MLP) algorithm. The collected data were classified as phases such as "Answering the Call," "During the Call," or "Hanging up the Call." As noted above, the research used the Multilayer Perceptron (MLP) algorithm and the following results were gathered:

- Results of experiment of "Answering the Call" phase: During this experiment, the testers picked up the phone from the table to view or answer the call. The accuracy rate was at 78% as table 4.1 below shows:

Table 4.1 Correct Classified Instance - Bringing Phone to View

Metric	Value	%
Correct Classified Instances	39	<u>78</u>
Incorrect Classified Instances	11	22
Kappa Statistic	0.8	
Total Number of Instances	50	

- Results of experiment of “During the Call” phase: During this experiment, the device was on the participant’s ear for the duration of the phone call. The accuracy rate was at 88% as shown in Table 4.2 below:

Table 4.2 Correct Classified Instance When Phone at Ear Level

Metric	Value	%
Correct Classified Instances	44	<u>88</u>
Incorrect Classified Instances	6	12
Kappa Statistic	0.85	
Total Number of Instances	50	

- Results of experiment of “Hanging up the Call” phase: During this final phase of the experiment, the phone was on its descent back into the stop position. The accuracy rate was at 74% as shown in table 4.3 below:

Table 4.3 Correct Classified Instance - Hanging Up the Phone

Metric	Value	%
Correct Classified Instances	37	<u>74</u>
Incorrect Classified Instances	11	22
Kappa Statistic	0.8	
Total Number of Instances	50	

4.4.1 Result of the spring 2015 Research

The research was successful. It showed that there were accuracies ranged between 74% and 88% using MLP algorithm for authenticating users depending on the motion of the phone. However, the main conclusion from the research was that sensor data can be used to authenticate Android-based mobile phone users and measure accuracy rates when both accelerometer and gyroscope sensors were used.

Proof of concept was accomplished with this project in spring 2015, and it proved that motions or movement of a phone could be used as a method of biometric authentication. With this baseline and the proof of concept established, the study was then expanded by having two additional studies in the following two semesters.

4.4.2 Conclusion of the Research of spring 2015

When looking at the results, it was clear that the accelerometer combined with the gyroscope was very accurate when the device was in a stationary position as seen in the “Talking On The Phone” experiment. The tests showed an accuracy rate of 88%. The conclusion was to follow up with the experiments by increasing the number of participants and by increasing the test cases to receive more data to provide confidence in the accuracy of the experiment.

4.5 Research Study in fall 2015

Upon having the proof of concept from the first study, the research continued in fall 2015 by conducting a study with a completely different, but larger, number of participants. There were a total of 20 participants who volunteered for the tests which was an increase of ten from the research that had been done in spring 2015. The research was successful in proving the hypothesis that one’s movements and motions measured from an Android-based phone could be used to authenticate a user. The study continued to use Android-based phones, the same mobile app that was used before (Sensor Kinetics Pro) and Weka Machine Learning library tool, to analyze the data generated by the built-in accelerometer and gyroscope sensors on those phones. This research incorporated increased training data, improved the data filtering, and used two additional Machine Learning classification algorithms compared to the prior semester’s study. With these improvements, this study supported the use of behavioral motion biometrics for

authentication. It achieved authentication accuracies of over 98% with several classification methods on three of the four devices tested.

4.5.1 Data Gathering Protocol for fall 2015 Research

There were four testers, each with one Android-based phone, for a total of four phones. Each teste coordinator found five individuals (as test participant) on whom to perform the tests. Each test participant performed the motion 25 times. Using a 10-fold cross-validation test, 90% of those served as the training data, while the remaining 10% served as the testing data against which the algorithm had to perform. This process repeated itself with a different tenth serving as the testing data.

The participants were handed one of the following four possible Android-based phones:

1. Google (LG) Nexus 5
2. HTC One M8
3. Motorola Moto G (3rd Generation)
4. Sony Xperia Z1 Compact

(Note: the “Motorola Moto G” phone did not have a gyroscope; it only captured accelerometer data.)

The tester coordinators assisted the test participants by running the “Sensor Kinetics Pro” application, prepping the device for capture, and stood back while the test participant motioned the phone in accordance with the test protocol, and stepped in to stop the motion capture when the motion was completed. They then prepared the device for the next round of capture. Upon positive cue from the teste coordinator, the standing participant performed the following actions the test protocol:

1. First long motion: he/she raised the phone from waist height to view.
2. Reviewed the screen for a moment, gently shook the phone with a simple flick of the wrist. This helped to signify that one motion had ended and another began.
3. Second long motion: he/she raised the phone the rest of the way to their ear.
4. A fourth motion was also captured, as the device was removed downwards from the ear to end the motion capture.

4.5.2 Test Results for fall 2015 Research

Cross-validation (10 fold) was used to evaluate the learning algorithms. This meant that each data set was randomly broken into 10 equal parts, and then the algorithm was trained on nine of those parts, testing the algorithm's accuracies on the remaining data. This was repeated for each fold. Tables 4.4, 4.5, 4.6 show the results for four phones, each tested using five individual users, each user performing 25 trials, using Naïve Bayes (N-B), K-Nearest Neighbor (k-NN), and Multilayer Perceptron (MLP) algorithms. Further, the data from all the phones were aggregated together and put through the machine learning algorithms.

Unsurprisingly, there was degradation in performance when fewer data points were made available to the algorithms when only one sensor or only one motion was run.

Table 4.4 below shows the accuracy rate when Accelerometer only was used.

Table 4.4 Fall 2015 Research - Accuracy Rate with Accelerometer - Manual Extraction

	Nexus 5	HTC One M8	Moto G (3)	Xperia Z1	All Phones
Accelerometer					
Motion 1	N-B: 91.2%	N-B: 96.8%	N-B: 71.2%	N-B: 95.2%	N-B: 85.0%
	k-NN: 95.2%	k-NN: 95.2%	k-NN: 82.4%	k-NN: 90.4%	k-NN: 84.8%
	MLP: 95.2%	MLP: 98.4%	MLP: 83.2%	MLP: 96.0%	MLP: 86.2%
Motion 2	N-B: 94.4%	N-B: 99.2%	N-B: 84.0%	N-B: 84.0%	N-B: 86.6%
	k-NN: 95.2%	k-NN: 100%	k-NN: 91.2%	k-NN: 84.8%	k-NN: 87.6%
	MLP: 97.6%	MLP: 99.2%	MLP: 93.6%	MLP: 88.0%	MLP: 86.2%
Combined motions (1&2)	N-B: 96.0%	N-B: 98.4%	N-B: 86.4%	N-B: 91.2%	N-B: 91.4%
	k-NN: 98.4%	k-NN: 100%	k-NN: 90.4%	k-NN: 89.6%	k-NN: 91.4%
	MLP: 98.4%	MLP: 100%	MLP: 94.4%	MLP: 93.6%	MLP: 94.2%

Table 4.5 below shows the accuracy rate when Gyroscope only was used.

Table 4.5 fall 2015 Research - Accuracy Rate with Gyroscope – Manual Extraction

	Nexus 5	HTC One M8	Moto G (3)	Xperia Z1	All Phones
Gyroscope					
Motion 1	N-B: 77.6%	N-B: 95.2%	N-B: *	N-B: 68.8%	N-B: *
	k-NN: 75.2%	k-NN: 96.0%	k-NN: *	k-NN: 74.4%	k-NN: *
	MLP: 85.6%	MLP: 98.4%	MLP: *	MLP: 81.6%	MLP: *
Motion 2	N-B: 89.6%	N-B: 87.2%	N-B:*	N-B: 80.8%	N-B:*
	k-NN: 96.0%	k-NN: 91.2%	k-NN:*	k-NN: 78.4%	k-NN:*
	MLP: 96.0%	MLP: 93.6%	MLP:*	MLP: 82.4%	MLP:*
Combined motions (1&2)	N-B: 89.6%	N-B: 94.4%	N-B:*	N-B: 82.4%	N-B:*
	k-NN: 92.0%	k-NN: 96.0%	k-NN:*	k-NN: 91.2%	k-NN:*
	MLP: 96.0%	MLP: 98.4%	MLP:*	MLP: 92.8%	MLP:*

Table 4.6 fall 2015 Research: Accuracy Rate – Two Sensors – Manual Extraction

	Nexus 5	HTC One M8	Moto G (3)	Xperia Z1	All Phones
Accelerometer & Gyroscope combined					
Motion 1	N-B: 88.0%	N-B: 98.4%	N-B: 71.2%	N-B: 90.4%	N-B: 85.0%
	k-NN: 94.4%	k-NN: 96.8%	k-NN: 82.4%	k-NN: 94.4%	k-NN: 84.8%
	MLP: 96.8%	MLP: 98.4%	MLP: 83.2%	MLP: 96.0%	MLP: 90.4%
Motion 2	N-B: 94.4%	N-B: 98.4%	N-B: 84.0%	N-B: 88.0%	N-B: 88.8%
	k-NN: 95.2%	k-NN: 100%	k-NN: 91.2%	k-NN: 89.6%	k-NN: 92.2%
	MLP: 97.6%	MLP: 100%	MLP: 93.6%	MLP: 96.0%	MLP: 93.6%
Combined motions (1&2)	N-B: 93.6%	N-B: 98.4%	N-B: 86.4%	N-B: 88.8%	N-B: 91.0%
	k-NN: 97.6%	k-NN: 100%	k-NN: 90.4%	k-NN: 95.2%	k-NN: 93.4%
	MLP: 98.4%	MLP: 100%	MLP: 94.4%	MLP: 98.4%	MLP: 96.2%

Table 4.6 above shows the accuracy rate with both Accelerometer and Gyroscope (manual feature extraction run) in fall 2015 Research

Table 4.7 below shows the Weka learning algorithms performed on the data with no manual feature extraction; the entire run was broken into eighths, processed, and run in Weka. Table 4.7 shows the automated run for accelerometer only, gyroscope only, and both sensors were used. With the additional data from both sensors incorporated (that is, not taken out via the process of feature extraction), the accuracy tended to improve overall.

Table 4.7 fall 2015 Research: Accuracy Results (Automated Run)

	Nexus 5	HTC One M8	Moto G (3)	Xperia Z1	All Phones
Accelerometer					
Full Motion	N-B: 96.8%	N-B: 97.6%	N-B: 92.8%	N-B: 94.4%	N-B: 94.0%
	k-NN: 94.4%	k-NN: 99.2%	k-NN: 95.2%	k-NN: 96.8%	k-NN: 96.6%
	MLP: 98.4%	MLP: 99.2%	MLP: 97.6%	MLP: 96.8%	MLP: 98.4%
Gyroscope					
Full Motion	N-B: 96.0%	N-B: 94.4%	N-B: *	N-B: 92.0%	N-B: *
	k-NN: 94.4%	k-NN: 97.6%	k-NN: *	k-NN: 92.8%	k-NN: *
	MLP: 97.6%	MLP: 99.2%	MLP: *	MLP: 96.0%	MLP: *
Accelerometer & Gyroscope combined					
Full Motion	N-B: 97.6%	N-B: 96.0%	N-B: 92.8%	N-B: 91.2%	N-B: 94.0%
	k-NN: 97.6%	k-NN: 99.2%	k-NN: 95.2%	k-NN: 96.8%	k-NN: 96.6%
	MLP: 99.2%	MLP: 99.2%	MLP: 97.6%	MLP: 98.4%	MLP: 98.4%

4.5.3 Conclusion of the fall 2015 Research

The results supported that biometric authentication could reliably be used. The best performing algorithm, of the ones tested, was the Multilayer Perceptron (MLP) algorithm. It also carried the heaviest data-processing footprint of the three to build its model. It is worth noting the improvements in the performance of the MLP algorithm of this research compared to the analyses by the prior semester's research of spring 2015 (~78%) for comparable motions that had been tested. There were several possibilities to explain the causes of this improvement in quality:

- First, the drastic expansion in number of samples per person, likely played a role in better training the algorithm to recognize different individuals.
- Second, breaking the motions into quartiles during the manual feature extraction, and eighths during the automated runs of the full motion, created much more idiosyncratic data points for everyone, rather than the overall averaging of the x, y and z components across the entire motion. Furthermore, the motions analyzed here were from waist/pocket height to visual inspection level, then to one's ear (two separate motions), as well as having the entire trial run for the fully automated process. In other words, each dimension was broken into eight sections, rather than averaged overall.
- Third, cutting out the additional processing steps helped. For example, filtering out the effects of gravity via approximation, which likely served to introduce noise; or attempting to determine the most salient attributes and filtering out the less salient ones. These steps likely improved the results by providing the maximum number of data points with the minimum amount of noise to the MLP algorithm. The MLP algorithm internally handles assigning salience values to the various elements being compared, likely done during the longer (approximately four seconds in duration) model building process. After analyzing the three algorithm results, it was determined that MLP was accurate enough that fusing the results would result in degradation in accuracy. There were no samples found that MLP did not classify correctly that k-NN and N-B both could classify correctly. It would be more likely that MLP would make a correct classification that k-NN and N-B would misclassify. Therefore, in these

instances, fusing the results of all three would not achieve any improvement over MLP alone.

- Finally, the quality of the algorithms' models seemed positively correlated with having a better gyroscope sensor. The worst performance occurred on the "Moto 3 (3rd generation)" phone, which lacked the gyroscope entirely. The middle performer, the "Nexus 5", had a gyroscope that only sampled at approximately 50% the rate of the accelerometer. Moreover, the remaining phones with superior gyroscopes that sampled at the same rate as their accelerometers fared best.

4.6 Research Study in spring 2016

The final and most comprehensive tests were performed in spring 2016. There were 60 different and new test participants. The research was successful, and it proved again that behavioral biometrics (in this case, the ways a phone is held and moved for two large motions) could even reach higher level of accuracy rate to authenticate a user of an Android-based smartphone. The research was focused on two large motions: picking up a phone that had been laid flat on a table after the mobile app had started, and bringing it up to the holding position in the front of the face, pausing for couple of seconds, and then moving it to the ear level.

4.6.1 Data Gathering Protocol for Research of spring 2016

Behavioral biometrics require a larger set of training data for the phone to learn a user's unique use patterns when compared to physiological biometrics capture, such as fingerprinting or iris scanning. Therefore, in this final study in spring 2016, the data gathering sources like test participants and phones were increased:

a) To create large number of training datasets from different models of Android-based phones (to diversify and to provide transparency that the research was not dependent on only one model or from one manufacturer), and

b) To increase the value and depth of the study by having large number of test participants of different ages, genders, and locations, again to increase the value of the research. In this final study, there were six test coordinators, each tested and worked with nine additional test participants and tested with different Android-based phone models.

The following phones and models were used (two test coordinators used the same type and model):

1. LG Nexus 5
2. Motorola Moto G (3rd Gen)
3. HTC One m8
4. Samsung Galaxy Note 4 (2 testers)
5. Samsung Galaxy Note 5

(Note: the “Motorola Moto G” phone did not have a gyroscope; it only captured accelerometer data.)

Figure 4.2 below shows summary information about the data and participants' profile:

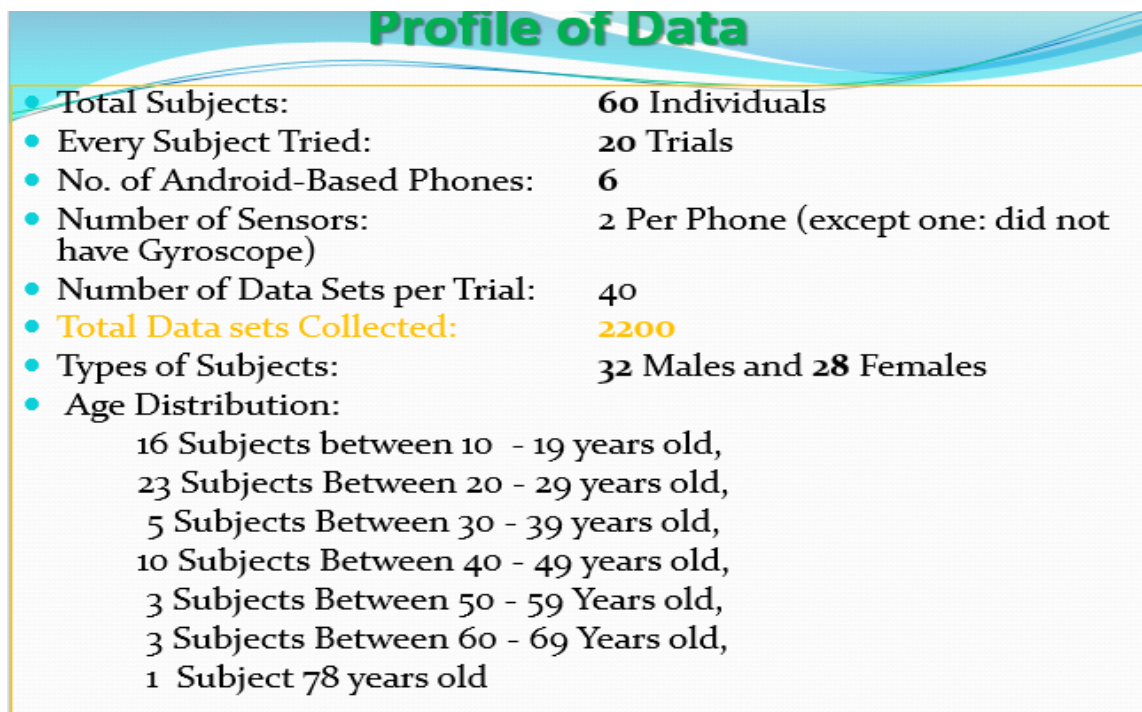


Figure 4.2 Profile of Test Participants in spring 2016 Research

The following in Figure 4.3 is a quick summary of the data and results.

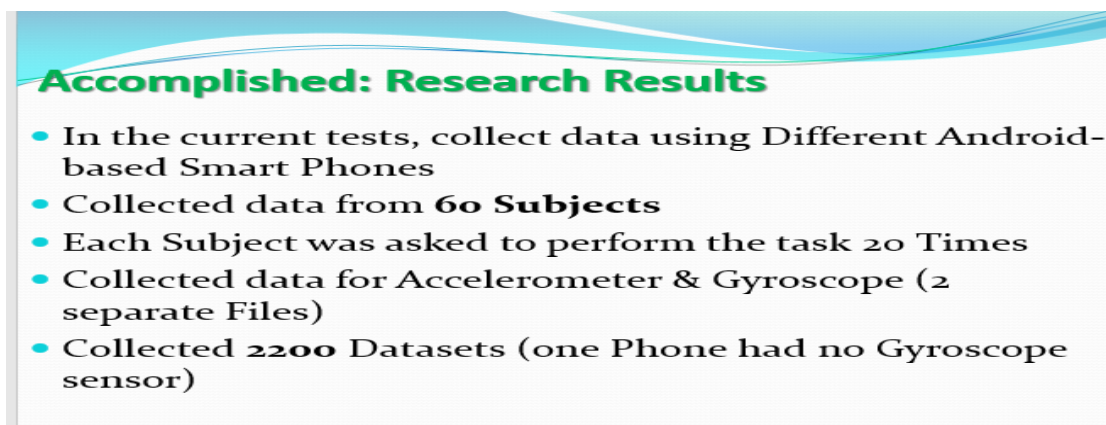


Figure 4.3 spring 2016: Summary of Research Data and Activities

Each test coordinator gathered data from 10 participants (many test coordinators were also participants themselves) who performed 20 trials each, for a total of 200 runs per tester or 1200 runs in total. There were 60 participants. As noted above, the “Motorola Moto G” did not have a gyroscope sensor; therefore, it only recorded data from its accelerometer.

The test coordinators facilitated by running the “Sensor Kinetics Pro” application, preparing the Android phone for capture, observing while the test participant motioned the phone in accordance with the test protocol, and getting the device ready for the next round of capture.

While various motions were considered for this study, the ones chosen were deemed best because it came naturally to most people and did not feel artificial. No time limit was set for the test participants. The only time enforcement involved was the 2-3 second pauses in between motions.

The trial procedure was straightforward. The participant either sat at or stood by a table or a desk with the phone face up on the surface of the table/desk. The participant pressed the “Start” button while the phone was still lying flat, and then lifted it to view. After a 2-3 second pause, the participant raised the phone with his or her hand of choice to his or her ear. The “Stop” button was then pressed at the ear after another 2-3 second pause. Again, as noted above, the test coordinator then stopped the motion capture of the device, and prepared for the next trial run until all the trials were completed. Table 4.8 shows a sample image of the data captured.

Table 4.8 Data Captured from the Sensor Kinetics Pro Application.

0.000	0.00000	0.00000	0.00000
0.060	0.20000	0.64000	10.28000
0.130	-0.05000	0.64000	9.94000
0.199	0.06000	0.62000	10.18000
0.259	0.05000	0.66000	10.18000
0.329	0.06000	0.63000	10.14000
0.400	0.03000	0.68000	10.14000
0.464	0.03000	0.64000	10.12000
0.529	0.03000	0.64000	10.11000
0.590	0.01000	0.67000	10.13000

4.6.2 Discarding Some Data

During the process of gathering the 20 trial runs from participants, the test coordinators would find that occasional trials would have to be discarded. The rate of discards per test participant ranged from 1 to 4, or a rate of 5% to 20%.

4.6.3 Data Processing Methods and Machine Learning for Research of spring 2016

Data was processed using two different methods. First, a Simple division of the recorded trial runs was executed, and then an advanced feature extraction process (referred here as Complex) to isolate the motions and pauses into separate segments.

4.6.3.1 Simple Division Method

The approach of simple division was entirely automated using a Java program. The Java program read in the 20 trials of participant motion data, provided in .CSV format from “Sensor Kinetics Pro” application. Then that data was divided into 16 sections, which was chosen to maintain parity with the second method that was the algorithmic feature extraction detailed below (four segments, each broken into four sections). The data points in the 16 sections were

averaged and their variances were calculated in the x, y, and z-axes separately. With two sensors, this totaled 192 data points.

The following graph in Figure 4.4 shows how the each of the four large segments were further divided into four to provide further detailing and observation for movements of the phones:

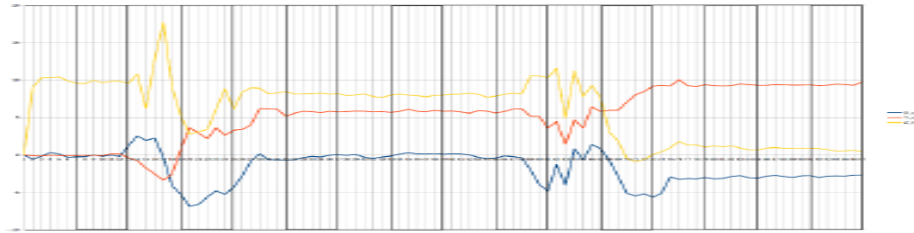


Figure 4.4 Simple Division Method

The Figure 4.4 above shows how the “Simple” division works: simple division method divides a trial run into 16 sections with no attention paid to areas of motion or stillness

4.6.3.2 Complex Division (Algorithmic Extraction) Method

For the algorithmic feature extraction, the following steps were performed: Using a graph of the accelerometer data, a moving average and variance was calculated for each time sample (separate x , y and z -axes). Summing the variances for each sample gave a picture of the motion of the phone at each point in the trial run. A temporary threshold of motion was then adjusted via processing each run until it produced four segments: two in motion and two at rest (held in front of the face and at the ear level). Depending on whether the accelerometer and gyroscope captured data at the same sampling rate (the “Nexus 5” phone did not), the time indices for the gyroscope were adjusted to match the timestamps identified for the accelerometer.

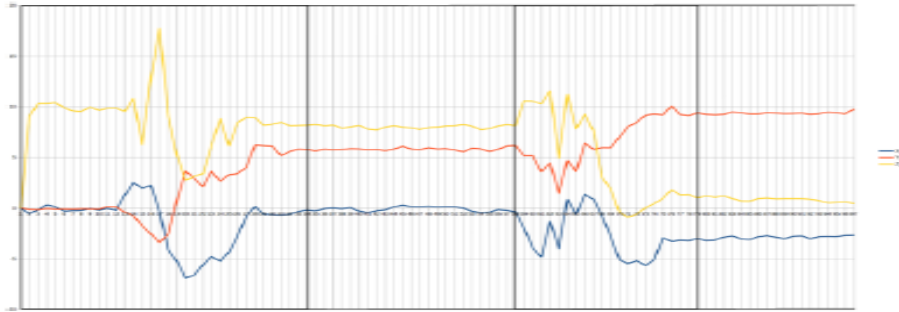


Figure 4.5 Complex Processing of the Motion to Isolate Movement

Figure 4.5 above shows the complex process: complex processing of the motion was to isolate movement from periods of stillness. Not shown are the divisions of these four segments into four sections.

Each segment was divided into quarters, and then averaged $\{x, y, z\}$ separately, and had a variance taken $\{x, y, z\}$ separately. This way, each segment had 24 attributes for the accelerometer, and 24 attributes for the gyroscope, if a gyroscope was present. Each trial consisted of four segments, and thus consisted of 192 total data points.

The Average and Variance per dimension $\{x, y, \text{ and } z\}$ were taken in each section with the intention of capturing everyone's idiosyncrasies. As the test participants performed the motions in accordance with their own preferences, the relative speed/acceleration of those motions should be distinguishable from the motions of others: using the quarter-section averages in each dimension representing the speed/angle of the motion of that section on average and the variances to approximate individual variation within each section.

The use of quartiles was intended to recapture individual idiosyncrasies, while making the data amenable to the WEKA library's numerous classification algorithms since dividing the data into smaller partitions would possibly create samples with too few data points that resulted in

meaningless variances” [6]. For both the Simple division and the feature extraction algorithm (i.e., Complex in this study), the information was aggregated into different .CSV files that were processed using Weka, and each 192-data-point row was post-pended with an identifier: the test participant’s number.

4.6.3.3 Feature Extraction Summary

The following is a summary of the feature extraction:

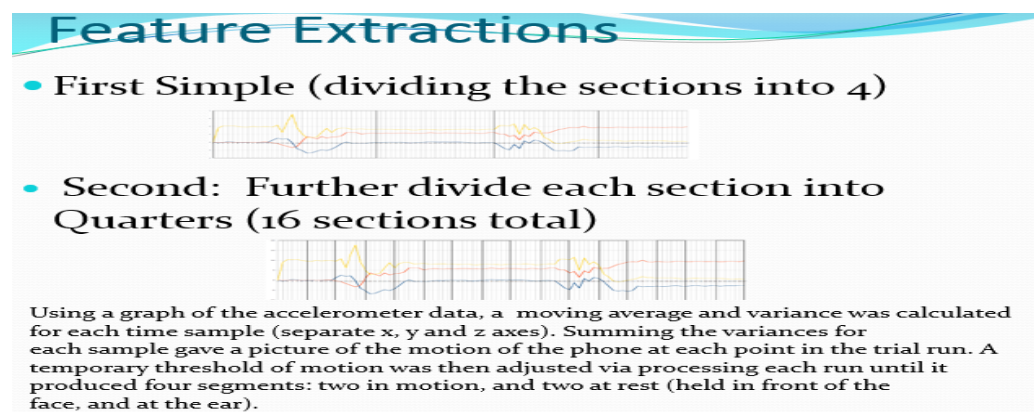


Figure 4.6 Feature Extraction Summary

4.7.4 Summary of the High-Level Data Processing

- Used an Automated process (Java Program) to divide the data into sixteen equal sections
- Calculated the average and variance at each data point
- Divided into sixteenths to match up evenly with the automated feature extraction
- 16x sections
- 2x sensors
- 3x dimensions
- 2x values (Average and Variance)
- Ended up with 192 data points per trial

4.7.5 Machine Learning Algorithms Used in the spring 2016 Research

With the data in this form, the algorithms selected to learn the participants and attempt to identify them were Naïve Bayes (N-B), K-Nearest Neighbors (k-NN), the Multilayer Perceptron (MLP), and Support Vector Machines (SVM).

Naïve Bayes is a well-known and widely used algorithm, selected by the team precisely for these reasons. Further, it was a very fast algorithm to run, however the ultimate performance of Naïve Bayes was not strong. It assumes features are independently formed from one another.

The K-Nearest Neighbors algorithm serves as a benchmark as being the standard in Machine Learning, and was a fast and simple algorithm to run. It is also a well understood and widely used algorithm, known for sensitivity to local structures in the feature space.

Multilayer Perceptron (MLP) was also selected in prior studies in spring 2015 and fall 2015. As noted earlier, the training of the artificial neural network is performed by backpropagation training data back up to the input layer. The connections to the hidden layer underneath are weighted to create the desired results. During the training, the weights are adjusted after an initial arbitrary setting to close in upon the intended results.

Support Vector Machines (SVM) was also selected for its wide popularity. It creates two categories, then takes the training data, and assigns them to one or the other category.

4.7.6 Separate and Additional Experiments Conducted in spring 2016

This study also conducted two categories of additional experiments to measure the behavior of Machine Learning algorithms if certain variations were made. First, the experiments were run to measure if statistical Average was better if used instead of statistical Variance for the data points. Second, the researcher conducted a series of different sample sizes, again to measure how size influenced the accuracy rate.

4.7.6.1 Additional Experiments: Using Average vs. Variance

The same data samples of “Simple” and “Complex” were used. The data samples for both Simple and Complex contained data for both accelerometer and gyroscope sensors, and these data sets were then updated to either contain datasets for Averages or for Variances. To test for Average, all columns that contained Variances were deleted and the file was saved. A similar approach used to test for Variances. All columns that had the data for Average were all deleted and the file was saved with just Variance data. The results showed that all algorithms (Naïve Bayes, k-NN, and MLP) had a significantly higher level of accuracies when Average (AVG) was used when compared with Variance (VAR). Naïve Bayes had a significantly higher accuracy if Average was used instead of Variance. For example, it was at 90% accuracies for Simple method for Average, and 64% for Simple method for Variance. k-NN also had a higher accuracy if Average was used instead of Variance. For example, the accuracy rate was at 88% for Simple method for Average and 64% for Simple method for Variance.

Below is a summary of improvements when different ML algorithms were used to test Average vs. Variance statistical methods. The study compared the results between these two-separate statistical approached, AVG (Average) vs. VAR (Variance).

	Simple All	Complex All	Simple - AVG	Simple - VAR	Complex - AVG	Complex - VAR
Naïve Bayes	83.17%	83.59%	90%	64%	91.49%	63.45%
KNN	87%	89.84%	88.16%	60.58%	88.97%	63.54%
MLP	92.50%	92.71%	92.25%	60.17%	93.40%	69.09%

✓ AVG Did Far Better

Figure 4.7 Accuracy Rate: Average vs. Variance

Figure 4.7 above shows the accuracy rate when either Average or Variance was used. The results showed that all three algorithms had higher accuracy rates when Average statistical method was used. MLP was higher in all instances.

Below is the summary of all the test runs in graph form that shows that MLP in most cases did better.

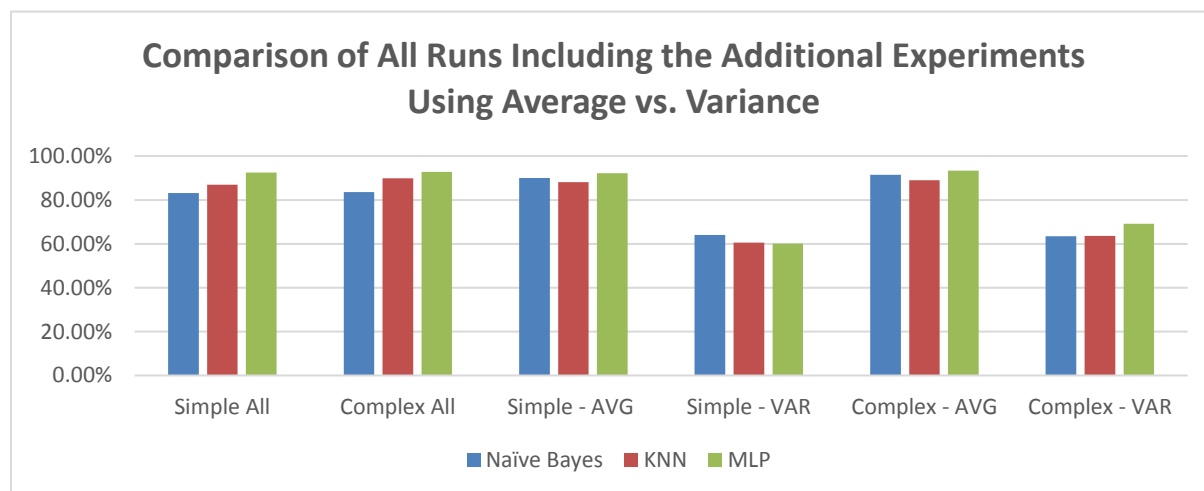


Figure 4.8 All Runs vs. the Additional Test Using Either Average or Variance

Figure 4.8 above shows the comparison of all runs including the additional experiment using Average vs. Variance.

4.7.6.2 Additional Experiments: Using Different Sample Sizes

The study then tried to use different sample sizes to measure the accuracies of the algorithms as the population size changed for both Simple and Complex methods. The study covered random samples sizes of five, ten and twenty. In all, as the population sizes grew, the accuracies changed (went down) or stayed almost the same when the same algorithms were used. However, when the algorithm was changed, the results were very different. Therefore, the choice of

algorithm mattered. Here, K-Nearest Neighbor (k-NN) fared far better than Naïve Bayes did, in all of the three sample sizes.

4.7.6.3 Results for Simple Method of Different Sample Sizes

Table 4.9, below, shows how algorithms behaved, and what type of accuracy rates were accomplished when random sizes of five, ten, or twenty were used. The study only used Naïve Bayes (N-B) and K-Nearest Neighbor (k-NN).

Table 4.9 Using Simple Methods Using Different Size Datasets

Average of Five Random Samples of Size Five, Ten, and Twenty			
Type: Simple	Random Samples of Five	Random Samples of Ten	Random Samples of Twenty
Naïve Bayes	85.20%	82.40%	82.10%
IBK-KNN	88.00%	89.70%	86.80%

Figure 4.9 below shows. In graph, that in Simple datasets, there were higher level of accuracies when k-NN algorithm was used as compared to Naïve Bayes.

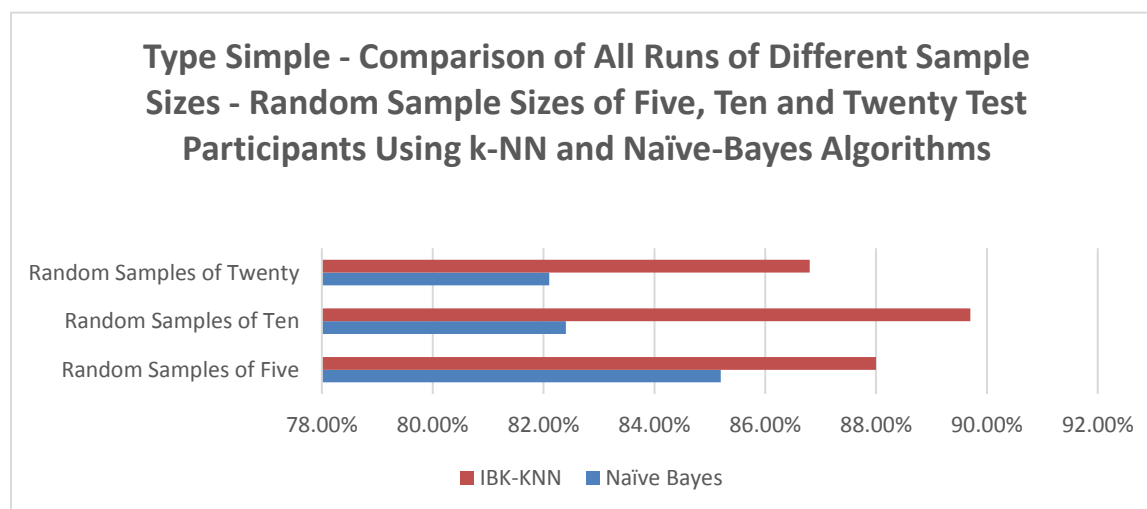


Figure 4.9 Using Simple Methods for Different Size Datasets

4.7.7 Results for Complex Method Using Different Sample Sizes

Table 4.10 below shows the accuracy rate of using Complex method when random sizes of five, ten and twenty were under Naïve Bayes and K-Nearest Neighbor. The algorithms had higher level of higher accuracy rates or stayed the same as population size grew.

Table 4.10 Results for Complex Method Different Sample Sizes

Average of Five Random Samples of Size Five, Ten, and Twenty			
Type: Complex	Random Samples of Five	Random Samples of Ten	Random Samples of Twenty
Naïve Bayes	85.91%	84.56%	85.04%
IBK-KNN	90.91%	88.94%	89.28%

Figure 4.10 below shows that in Complex method, there were significantly higher accuracies for k-NN algorithm as compared to Naïve Bayes.

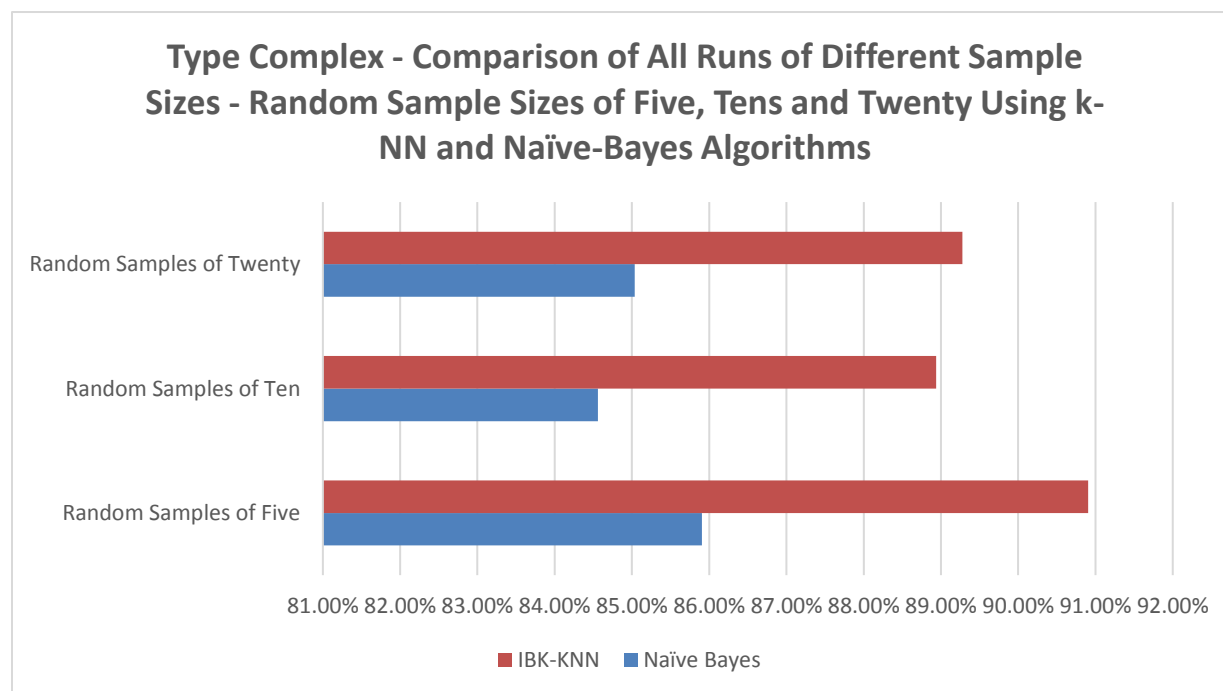


Figure 4.10 Complex Method: Accuracy Rates - k-NN Algorithm vs. Naive Bayes

4.7.8 The Final Test Results for Research of spring 2016

The Cross Validation using 10-folds was used with each of the algorithms. For each algorithm, the total test data was split into ten equal partitions, and the algorithms were trained on 9/10ths. Then the remaining 10th was tested against the trained algorithms. This process was repeated for each partition—a different tenth—serving as the testing data.

Table 4.11 shows the results for the 60 users, tested using six phones (five different models), using algorithms Naïve Bayes (N-B), K-Nearest Neighbors (k-NN), the Multilayer Perceptron (MLP), and Support Vector Machine (SVM).

First, worth noting is the clear improvement in results when the gyroscope is added to the analysis: As shown here and observed in the fall 2015 research paper [6], having the additional sensor aided the algorithms in correctly identifying individuals, resulting in nearly across the board improvements of 1.1%.

Second, the improvement in specifically isolating the motions using the complex segmentation method is visibly evident in the results. An improvement of 1.1% is noted by doing this additional work

Third, the choice of algorithm mattered and it was significant. Going from Naïve Bayes to K-Nearest Neighbors provided a jump of 6.4% for tests used the Complex extraction method and when both sensors were used. Then moving to Multilayer Perceptron resulted in a 5.0% improvement. From the worst to the best performing algorithm, there was a difference of 11.8%.

Table 4.11 Comparison of Results from Four Algorithms

%	Simple (accelerometer)	Simple (both)	Complex (accelerometer)	Complex (both)
N-B	82.7	83.2	81.1	83.6
k-NN	86.3	87	88.7	89.8
MLP	91.4	92.5	92.9	92.7
SVM	96.3	92.8	97.7	92.2

Table 4.11 above shows how the four algorithms (Naïve Bayes, K-Nearest Neighbors, Multi-Layer Perceptron, and Support Vector Machine) fared when the data from the sensors were used, either individually or when both were used, under either the Simple or the Complex method. Support Vector Machine (SVM) and Multilayer Perceptron (MLP) did far better than Naïve Bayes (N-B) and K-Nearest Neighbor (k-NN). SVM even reached 97.7% accuracies when both sensors were used under the Complex method.

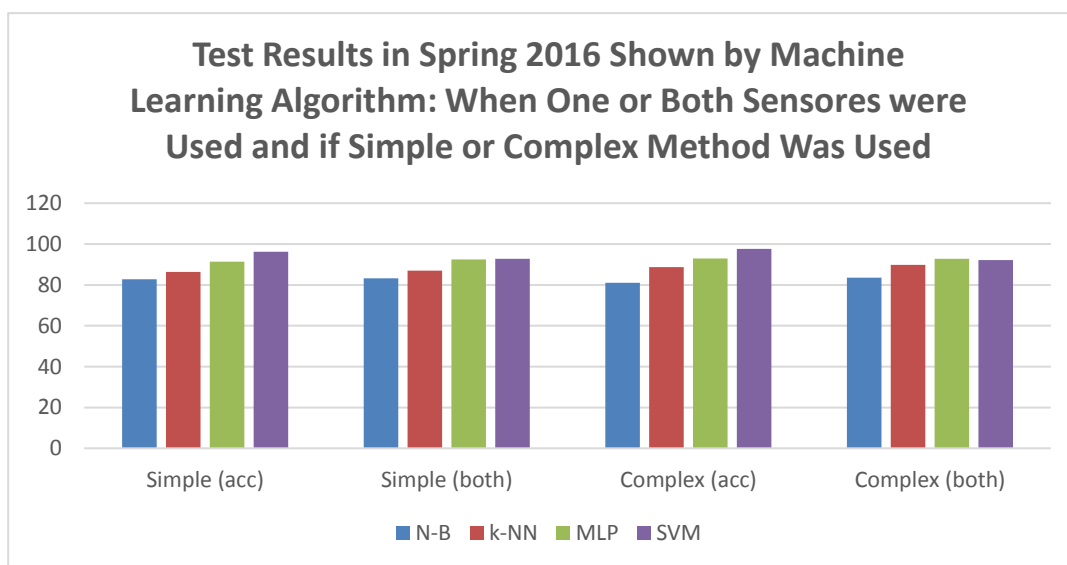
**Figure 4.11 Comparison of Results from Three Algorithms**

Figure 4.11 above shows how the four algorithms (Naïve Bayes, K-Nearest Neighbors, Multi-Layer Perceptron, and Support Vector Machine) fared when the data from the sensors were used, either individually or when both were used, under either the Simple or the Complex method. Table 4.12 below shows the percentage improvements when the study applied either of the three algorithms (MLP, k-NN, and N-B), from one sensor to two sensors, or when either Simple or Complex methods were used:

In one specific case, changing algorithm to MLP instead of N-B provided 9.1% performance increase.

Table 4.12 Approximate Values of Improvements of Research in Spring 2016

Improvement	Percent
From 1 sensor to 2 sensors (for MLP)	1.1%
From simple to complex (for K-NN)	2.1%
From N-B to k-NN (For Complex)	6.2%
From k-NN to MLP (for Simple)	5.5%
From N-B to MLP (for Complex)	9.1%

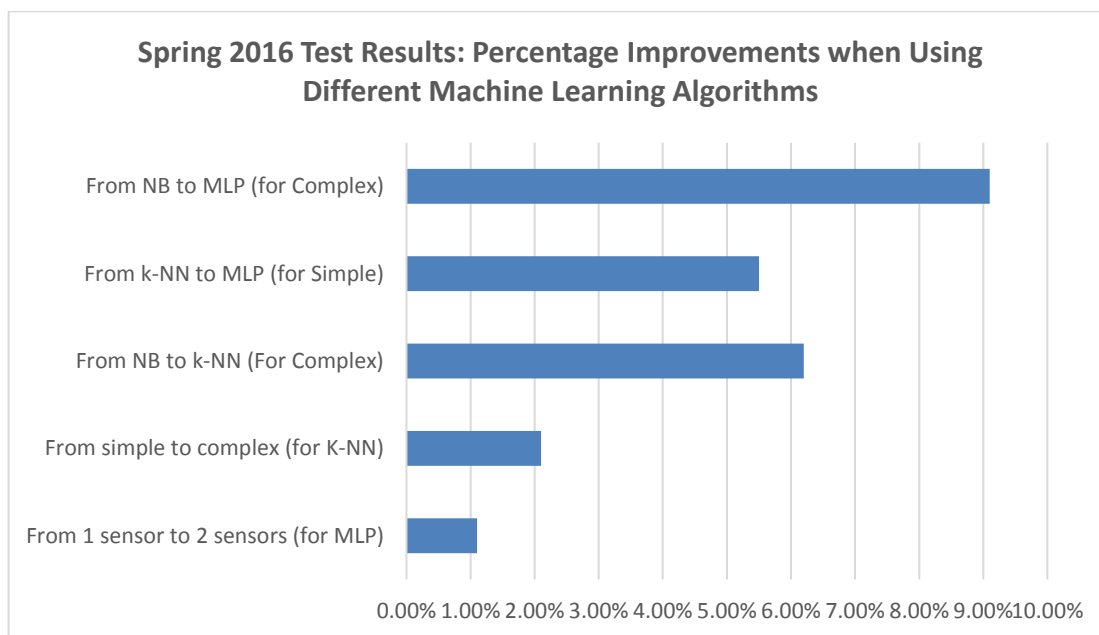


Figure 4.12 Approximate Values of Improvements Using Different Parameters

Figure 4.12, shows the approximate improvements when increasing the number of sensors, using Complex segmentation, or using a sophisticated algorithm.

One minor detail worth noting was that the Multilayer Perceptron algorithm seemed to perform slightly worse moving from one to two sensors in the Complex extraction. Given the general propensity of the algorithms to improve when (1) more sensors are used, and (2) the Complex method was applied, the research showed that overall, the best combination of factors was two sensors, Complex extraction, and the Multilayer Perceptron algorithm.

4.8 False positive Rates per Individual Test Participant

One final piece about the data was that Weka provides some false positive rates per individual test participant that helps to determine the relevance. The figure below shows sample results from Weka displaying true positive and false positive rates per individual test participant:

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.99	0	1	0.9	0.947	1	subject
0.995	0.001	0.95	0.95	0.95	1	subject
0.99	0.001	0.947	0.95	0.923	0.971	subject
1	0	1	1	1	1	subject
0.85	0.001	0.944	0.85	0.895	0.941	subject
0.9	0.002	0.9	0.9	0.9	0.995	subject
1	0	1	1	1	1	subject
1	0	1	1	1	1	subject
0.95	0.002	0.905	0.95	0.927	1	subject
0.8	0.004	0.833	0.8	0.809	0.989	subject
0.883	0.002	0.883	0.8	0.842	0.985	subject
0.955	0.002	0.895	0.95	0.974	0.998	subject
0.955	0.002	0.895	0.95	0.872	0.985	subject
1	0.002	0.909	1	0.974	0.999	subject
1	0.001	0.909	1	0.952	1	subject
0.882	0.001	0.882	0.882	0.976	1	subject
0.9	0.001	0.947	0.9	0.923	0.997	subject
0.95	0.002	0.905	0.95	0.927	1	subject
1	0.001	0.952	1	0.976	1	subject
1	0	1	1	1	1	subject
1	0	1	1	1	1	subject
0.95	0	1	0.95	0.974	1	subject
0.95	0.004	0.826	0.95	0.884	0.999	subject
1	0	1	1	1	1	subject
1	0.001	0.905	1	0.976	1	subject
0.883	0.001	0.883	0.883	0.882	1	subject
0.953	0.001	0.953	0.953	0.95	1	subject
0.953	0.001	0.953	0.953	0.95	1	subject
0.9	0.001	0.952	1	0.976	1	subject
0.58	0.002	0.667	0.9	0.9	0.997	subject
0.58	0.004	0.587	0.588	0.625	0.937	subject
0.95	0.001	0.955	0.955	0.95	1	subject
0.95	0.001	0.955	1	0.976	1	subject
0.7	0.003	0.733	0.7	0.739	1	subject
0.85	0.005	0.85	0.85	0.85	0.995	subject
0.85	0.001	0.739	0.85	0.791	0.991	subject
0.85	0.004	0.841	0.85	0.865	0.995	subject
0.85	0.002	0.818	0.8	0.857	0.993	subject
0.85	0.001	0.841	0.8	0.842	0.991	subject
1	0.001	0.841	1	0.865	0.996	subject
1	0	1	1	0.976	1	subject
0.9	0	1	1	1	1	subject
0.845	0.003	0.864	0.905	0.905	0.999	subject
0.845	0.004	0.733	0.845	0.786	0.999	subject
0.933	0	1	0.933	0.968	0.995	subject
0.933	0.001	0.933	0.933	0.929	0.991	subject
0.714	0.003	0.714	0.714	0.625	0.971	subject
0.714	0.004	0.714	0.75	0.789	0.955	subject
0.95	0.001	0.95	0.714	0.714	0.994	subject
0.95	0.001	0.95	0.95	0.95	1	subject
0.95	0.002	0.905	0.95	0.927	1	subject

Figure 4.13 True Positive and False Positive Rates per Individual Test Participant

4.9 Chapter Conclusion

In this chapter, the study concluded that the motion of hands could in fact be used as a valid behavioral biometrics method. The study showed that the research could be conducted multiple times (three times in this study) and the results could be duplicated to support the hypotheses noted in the problem statement described in chapter one. The study also concluded that the tests

could run data on Weka Machine Learning (ML) under several different ML algorithms to measure and compare the accuracy levels produced by the different algorithms. The study employed different tests on a specific statistical method, and it proved that selecting a specific statistical method matters. For example, using Average instead of Variance could result in vastly different and much higher accuracy rates. The next chapter will discuss the results of the study and research conducted and will recommend ways to enhance and add to the work in future similar studies.

Chapter 5

Conclusions

This chapter summarizes the results from the three different research studies and answers the question in the problems statement. It will discuss the limitations of the research and will provide recommendations for future study in this area.

5.1 Research Conclusion

All three research studies conducted show that there were advantages in using behavioral biometrics for authentication. However, given that while the research tests reached very high percentage accuracies in certain trials and in using certain algorithms, the test never reached 100% of accuracies. The results do not support the lone use of behavioral biometrics for authentication; it instead, it should be used as a complement or as an added authentication method. As an example, users could use behavioral biometrics in conjunction with a password or could combine it with some physiological biometrics like fingering to better authenticate users. In some cases, for example to unlock a device, it might be more advisable to rely upon physiological biometrics. Then, after being unlocked with a physiological biometric, it might make more sense to take advantage of behavioral biometrics to keep the phone unlocked over extended usage.

Worth noting is that this study's results in spring 2016 are slightly worse than the previous iteration of this experiment performed in fall 2015. In the previous study performed in fall 2015, a ~98% result was achieved when MLP algorithm, with dual sensors, and complex extraction method were used. One explanation for the mild decrease in the performance was due to the drastic increase in the number of participants, from 20 to 60. It was expected that the overall

performance would suffer because of the sizeable increase. It is also worth noting that, in the final study, data from identical twins was included. The additional discovery regarding the twins (detailed below) was confusing the algorithms to such a strong degree that it became a great concern with respect to security.

However, one improvement in performance was a result of the automated method for feature extraction, compared to the simplified version, which was a complete inversion from the research in fall 2015. First, the previous research in fall 2015 used a manual effort to visually identify the lifting motions and still portions, which compared unfavorably to the simple division into an arbitrary number of sections. In the last research, the simple division method underperformed compared to the algorithm that automated isolating the motions.

Like the research of fall 2015, the research in spring 2016 did not include extra processing steps such as filtering out the effects of gravity. This meant that when the algorithms were run, they had the maximum number of data points to work with along with a minimum amount of noise created by superfluous calculations.

It is worth noting that the MLP algorithm, when fed 1200 test trials and 196+1 attributes (+1 being the identification label), struggled under the weight of the learning data. With the 236,400 data points (1200 trials multiplied by 197), the MLP with 10-fold learning took over 40 minutes to complete.

After the experiment, the results dictate that MLP was accurate enough that combining it with any of the other algorithms would weaken the results. There were no instances where MLP misclassified a sample, and k-NN and N-B classified it correctly. Therefore, fusing the results of all three algorithms would serve no purpose other than to lower the results of MLP alone.

As with the previous research of spring 2015 and fall 2015, the learning algorithms in spring 2016 performed better with additional data: having the gyroscope data added to the analysis improved the results across the board. Some phones had middling sensors, particularly the gyroscope that sampled at lower rates. The “Nexus 5’s” gyroscope sampled at half the rate of its accelerometer. When isolated, the phones with superior sensors fared better, per the previous research in fall 2015 [6].

For phones with sensors that have a lower sampling rate but still wish to offer biometric authentication, cubic interpolation could possibly be an option. Cubic interpolation makes use of neighboring data points to approximate missing data. This idea, however, would require significant additional testing to see whether such virtual data would be able to compete with phone with high sample rates.

As noted earlier, one particularly noteworthy feature of the results was the fact that the Machine Learning algorithms frequently confused the data samples provided by the twins, participants 43 and 44. Their data samples were not correctly ascribed to the correct individual. This is likely due to their identical genetic makeup causing them to have identical or nearly identical physical characteristics. This finding had some influence on the study’s outcome, but it is a large factor in the final recommendation that the sort of behavioral biometric described in this study would not be effective as a sole means of authentication.

5.3 Future Development

After completing the analysis, there are some suggestions for future iterations of the experiment. One change would be to break up the data gathering into sections. In these experiments, the test participants had to perform all 20 trials in a single sitting. It would be interesting to see if collecting trials over multiple sessions for a test participant would have any effect on the results.

Another change would be to increase the number of participants to at least 100 or even more. Future studies could also benefit from having multiple populations from different ethnic backgrounds and from different locations around the world. In addition, this study recommends adding tests from more identical twins to see if the findings in the spring 2016 study could be duplicated with larger number of identical twin sets. Finally, the study recommends conducting additional tests to compare motions of right hands versus left hands of the same participant.

Appendix A

Execution Results of Random Sets of Five, Ten and Twenty

Below is the summary execution of sets of five by the Naïve Bayes and k-NN algorithms – based on either Simple or Complex data sets:

Simple Data Sets 0 to 4 (Total of five sets)

Simple: Naïve Bayes

Scheme:weka.classifiers.bayes.: NaïveBayes

Relation: simple Dataset 0 to 4

Simple: Naïve Bayes

Correctly Classified Instances 93

93%

Total Number of Instances 100

Simple Data Sets 0 to 4 (Total of five sets)

Simple: K-NN- ILK

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A Relation: simple Dataset 0 to 4

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 96

96%

Incorrectly Classified Instances 4 4%

Kappa statistic 0.95

Mean absolute error 0.032

Root mean squared error 0.1249

Relative absolute error 10 %

Root relative squared error 31.2261 %

Total Number of Instances 100

=== Detailed Accuracy By Class ===

Complex Data Sets 0 to 4 (Total of five sets)

Complex: Naive Bayes

Scheme:weka.classifiers.bayes.NaiveBayes

Relation: complex Datasets 0 to 4

Time taken to build model: 0.03 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	92	92%
Incorrectly Classified Instances	8	8%
Kappa statistic	0.9	
Mean absolute error	0.032	
Root mean squared error	0.1789	
Relative absolute error	10 %	
Root relative squared error	44.7214 %	
Total Number of Instances	100	

Complex Data Sets 0 to 4 (Total of five sets)

Complex: IBK-K-NN

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A

Relation: complex Datasets 0 to 4

Correctly Classified Instances	99	99%
Incorrectly Classified Instances	1	1%
Kappa statistic	0.9875	
Mean absolute error	0.0206	
Root mean squared error	0.0651	
Relative absolute error	6.4474 %	
Root relative squared error	16.2648 %	
Total Number of Instances	100	

Simple Data Sets 10 to 14 (Total of five sets)

Data sets 10 to 14

Simple: Naive Bayes

Scheme:weka.classifiers.bayes.NaiveBayes

Relation: simple 10 to 14

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	87	87%
Incorrectly Classified Instances	13	13 %
Kappa statistic	0.8375	
Mean absolute error	0.052	
Root mean squared error	0.228	
Relative absolute error	16.2529 %	
Root relative squared error	57.0088 %	
Total Number of Instances	100	

Simple Data Sets 10 to 14 (Total of five sets)

Data sets 10 to 14

Simple: IBK-K-NNScheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
\"weka.core.EuclideanDistance -R first-last\""

Relation: simple 10 to 14

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	92	92%
Incorrectly Classified Instances	8	8%
Kappa statistic	0.9	
Mean absolute error	0.0472	
Root mean squared error	0.1754	
Relative absolute error	14.7368 %	
Root relative squared error	43.8456 %	
Total Number of Instances	100	

=====

Complex Data Sets 10 to 14 (Total of five sets)

Data sets 10 to 14

Complex: Naive Bayes

Scheme:weka.classifiers.bayes.NaiveBayes

Relation: complex 10 to 14

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	86	86%
Incorrectly Classified Instances	14	14%
Kappa statistic	0.825	
Mean absolute error	0.0557	
Root mean squared error	0.2353	
Relative absolute error	17.4109 %	
Root relative squared error	58.8367 %	
Total Number of Instances	100	

=====

Complex Data Sets 10 to 14 (Total of five sets)

Data sets 10 to 14

Complex: IBK-K-NN

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
\"weka.core.EuclideanDistance -R first-last\""

Relation: complex 10 to 14

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	92	92%
Incorrectly Classified Instances	8	8%
Kappa statistic	0.9	
Mean absolute error	0.0472	
Root mean squared error	0.1754	
Relative absolute error	14.7368 %	
Root relative squared error	43.8456 %	
Total Number of Instances	100	

Simple Data Sets 32 to 36 (Total of five sets)

Data sets 32 to 36

Simple: Naive Bayes

Scheme:weka.classifiers.bayes.NaiveBayes

Relation: simple 32 to 36

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	88	88%
Incorrectly Classified Instances	12	12%
Kappa statistic	0.85	
Mean absolute error	0.048	
Root mean squared error	0.2191	
Relative absolute error	15 %	
Root relative squared error	54.7723 %	
Total Number of Instances	100	

Simple Data Sets 32 to 36 (Total of five sets)

Simple: IBK - K-NN

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A

Relation: simple 32 to 36

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	96	96%
Incorrectly Classified Instances	4	4 %
Kappa statistic	0.95	
Mean absolute error	0.032	
Root mean squared error	0.1249	
Relative absolute error	10 %	
Root relative squared error	31.2261 %	
Total Number of Instances	100	

Complex Data Sets 32 to 36 (Total of five sets)

Complex: Naive Bayes

Scheme:weka.classifiers.bayes.NaiveBayes

Relation: complex 32 to 36

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	82	84.53 %
Incorrectly Classified Instances	15	15.4639 %
Kappa statistic	0.8071	
Mean absolute error	0.0614	
Root mean squared error	0.2467	
Relative absolute error	19.2048 %	
Root relative squared error	61.6813 %	
Total Number of Instances	97	

Complex Data Sets 32 to 36 (Total of five sets)

Complex: K-NN

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
 \"weka.core.EuclideanDistance -R first-last\""
 Relation: complex 32 to 36

Time taken to build model: 0 seconds

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	87	89.69 %
Incorrectly Classified Instances	10	10.3093 %
Kappa statistic	0.8708	
Mean absolute error	0.0563	
Root mean squared error	0.1987	
Relative absolute error	17.6195 %	
Root relative squared error	49.6818 %	
Total Number of Instances	97	

Simple Data Sets 40 to 44 (Total of five sets)

Data Set 40 to 44

Simple: Naive Bayes

Scheme:weka.classifiers.bayes.NaiveBayes
 Relation: simple 40 to 44

Time taken to build model: 0.02 seconds

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	72	72 %
Incorrectly Classified Instances	28	28 %
Kappa statistic	0.65	
Mean absolute error	0.112	
Root mean squared error	0.3346	
Relative absolute error	34.9931 %	
Root relative squared error	83.6495 %	
Total Number of Instances	100	

Simple Data Sets 40 to 44 (Total of five sets)

Simple: IBK

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
\"weka.core.EuclideanDistance -R first-last\""

Relation: simple 40 to 44

Time taken to build model: 0.02 seconds

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	72	72 %
Incorrectly Classified Instances	28	28 %
Kappa statistic	0.65	
Mean absolute error	0.1229	
Root mean squared error	0.3264	
Relative absolute error	38.4211 %	
Root relative squared error	81.6044 %	
Total Number of Instances	100	

Complex Data Sets 40 to 44 (Total of five sets)

Complex: Naive Bayes

Scheme:weka.classifiers.bayes.NaiveBayes

Relation: complex 40 to 44

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	75	75 %
Incorrectly Classified Instances	25	25 %
Kappa statistic	0.6875	
Mean absolute error	0.0997	
Root mean squared error	0.3134	
Relative absolute error	31.1473 %	
Root relative squared error	78.3467 %	
Total Number of Instances	100	

Complex Data Sets 40 to 44 (Total of five sets)

Complex: K-NN - IBK

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
\"weka.core.EuclideanDistance -R first-last\""

Relation: complex 40 to 44

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	82	82 %
Incorrectly Classified Instances	18	18 %
Kappa statistic	0.775	
Mean absolute error	0.0851	
Root mean squared error	0.262	
Relative absolute error	26.5789 %	
Root relative squared error	65.5046 %	
Total Number of Instances	100	

Simple Data Sets 57 to 62 (Total of five sets)

Data Set 57 to 62

Simple: Naive Bayes

Scheme:weka.classifiers.bayes.NaiveBayes

Relation: simple 57 to 62

Time taken to build model: 0.02 seconds

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	86	86 %
Incorrectly Classified Instances	14	14 %
Kappa statistic	0.825	
Mean absolute error	0.0571	
Root mean squared error	0.2373	
Relative absolute error	17.8419 %	
Root relative squared error	59.3186 %	
Total Number of Instances	100	

Simple Data Sets 57 to 62 (Total of five sets)

Simple: IBK - K-NN

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
\"weka.core.EuclideanDistance -R first-last\""

Relation: simple 57 to 62

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	84	84 %
Incorrectly Classified Instances	16	16 %
Kappa statistic	0.8	
Mean absolute error	0.0775	
Root mean squared error	0.2471	
Relative absolute error	24.2105 %	
Root relative squared error	61.7833 %	
Total Number of Instances	100	

Complex Data Sets 57 to 62 (Total of five sets)

complex: Naïve Bayes

Scheme:weka.classifiers.bayes.NaïveBayes

Relation: complex 57 to 62

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	87	92.55%
Incorrectly Classified Instances	7	7.46 %
Kappa statistic	0.9064	
Mean absolute error	0.0298	
Root mean squared error	0.1717	
Relative absolute error	9.3416 %	
Root relative squared error	42.9884 %	
Total Number of Instances	94	

Complex Data Sets 57 to 62 (Total of five sets)

Complex: IBK - K-NN

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
\"weka.core.EuclideanDistance -R first-last\""

Relation: complex 57 to 62

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	86	91.49 %
Incorrectly Classified Instances	8	8.5106 %
Kappa statistic	0.8931	
Mean absolute error	0.05	
Root mean squared error	0.1806	
Relative absolute error	15.68 %	
Root relative squared error	45.2388 %	
Total Number of Instances	94	

Appendix B

Java Program for Automated Feature Extraction

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.PrintWriter;

public class Demo {

    //RUNTIME SETTINGS
    public static final String directory =
"/home/jonl3379/Downloads/data/";
    public static final int[] subjects = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 47, 48, 49,
50, 51, 52, 53, 54, 55, 57, 58, 59,
60, 62};
    public static final int trials = 20;

    //GLOBAL SETTINGS
    //segments will be 4; sections per segment will be 4; to be defined
in feature_extraction()
    public static final String[] sensors = {"acc", "gyr"}; //{0,1} in
array notation
    public static final int[] sections = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12, 13, 14, 15, 16};
    public static final String[] statistics = {"avg", "var"};
    public static final String[] dimensions = {"x", "y", "z"}; //{1,2,3}
in array notation

    //DEMO MAIN BEGINS HERE
    public static void main(String[] args) throws Exception {

        //SETUP FILES START
        PrintWriter simple = new PrintWriter(directory + "simple.csv");
        PrintWriter complex = new PrintWriter(directory +
"complex.csv");

        //WRITES TOP LINE OF COLUMN LABELS INTO CSV, EG
"acclavgx,acclavy..."
        top_line(simple);
        top_line(complex);

        //SUBJECT RUNS READ & WRITE
        for (int subject : subjects)
            subject_runs(subject, simple, complex);
    }
}

```

```

//COMMIT FILES
simple.close();
complex.close();
System.out.println("***FIN***");
}

//POPULATE TOP LINE OF CSV
public static void top_line(PrintWriter p) {
    for (String sensor : sensors)
        for (int section : sections)
            for (String statistic : statistics)
                for (String dimension : dimensions)
                    p.print(sensor + section + statistic + dimension
+ ",");
    p.println("subject_number");
}

//SUBJECT DATA READ THEN WRITE TO CSV
public static void subject_runs(int subject, PrintWriter simple,
PrintWriter complex) throws Exception {

    //ARRAYS STORE DATA AND TRIAL LENGTHS
    double[][][][] data = new
double[trials][sensors.length][dimensions.length + 1][999];
    int[][] sample_length = new int[trials][sensors.length];

    //POPULATING DATA & SAVING SAMPLE LENGTHS
    for (int trial = 0; trial < trials; trial++) {
        sample_length[trial][0] = read(subject, trial, data, "acc");
        sample_length[trial][1] = read(subject, trial, data, "gyr");
        System.out.print(" read" + subject + "_" + trial);
    }
    System.out.println();

    //CALCULATING DATA FOR SIMPLE SECTIONS
    for (int trial = 0; trial < trials; trial++) {
        simple_sections(subject, trial, data, "acc",
sample_length[trial][0], simple);
        simple_sections(subject, trial, data, "gyr",
sample_length[trial][1], simple);
        simple.println("subject#" + subject);
    }

    //CALCULATING DATA FOR FEATURE EXTRACTION
    for (int trial = 0; trial < trials; trial++) {
        //more segments than needed, seeking algo will narrow to 4
        int[] segments = new int[16];
        //first, feature extraction using accelerometer
        boolean successful = feature_extraction(subject, trial,
data, sample_length[trial][0], segments);
        //then, sections within segmentation, write
        if (!successful) { //for unsuccessful extraction

```



```

        //simple_sections(subject, trial, data, "acc",
sample_length[trial][0], complex);
        //simple_sections(subject, trial, data, "gyr",
sample_length[trial][1], complex);
        //complex.println("subject#" + subject);
    } else {
        complex_sections(subject, trial, data, "acc",
sample_length[trial][0], segments, complex);
        complex_sections(subject, trial, data, "gyr",
sample_length[trial][1], segments, complex);
        complex.println("subject#" + subject);
    }
}
}

//DATA READING INTO ARRAY
public static int read(int subject, int trial, double[][][][] data,
String sensor) throws Exception {
    int sample = 0;
    int sensor_num = 0;
    if (sensor.equals("gyr"))
        sensor_num = 1;

    if (sensor_num == 1 && subject > 9 && subject < 20) return 0; //
skip Aliza's no gyr phone

    BufferedReader r = new BufferedReader(new FileReader(directory +
subject + "/Chart " + (trial + 1) + "_" + sensor + ".csv from
SensorKineticsPro"));
    String line = r.readLine(); //discard top row of labels
    line = r.readLine();
    while (line != null) {
        sample++;
        String[] token = line.split(",");
        //reserving data[trial][sensor_num][0][trial] for segment
number(in future), token[0] is time stamp, skip
        data[trial][sensor_num][1][sample] =
Double.parseDouble(token[1]);
        data[trial][sensor_num][2][sample] =
Double.parseDouble(token[2]);
        data[trial][sensor_num][3][sample] =
Double.parseDouble(token[3]);
        line = r.readLine();
    }
    r.close();
    return sample; //returns the last occupied sample
}

//SIMPLE SECTIONS DIVISION AVG/VAR CALCULATION
public static void simple_sections(int subject, int trial,
double[][][][] data, String sensor, int sample_length, PrintWriter
simple) {
    int sections = Demo.sections.length;

```

```

int sensor_num = 0;
if (sensor.equals("gyr")) sensor_num = 1;

for (int section = 0; section < sections; section++) {
    if (subject >= 10 && subject <= 19 && sensor_num == 1) {
//Aliza's no GYR phone
        simple.print("0,0,0,0,0,0,");
        continue;
    }
    double avg_x, avg_y, avg_z, var_x, var_y, var_z;

    //AVERAGE
    avg_x = average(trial, sensor_num, 1, data, section *
sample_length / sections, (section + 1) * sample_length / sections);
    avg_y = average(trial, sensor_num, 2, data, section *
sample_length / sections, (section + 1) * sample_length / sections);
    avg_z = average(trial, sensor_num, 3, data, section *
sample_length / sections, (section + 1) * sample_length / sections);

    //VARIANCE
    var_x = variance(trial, sensor_num, 1, data, section *
sample_length / sections, (section + 1) * sample_length / sections,
avg_x);
    var_y = variance(trial, sensor_num, 2, data, section *
sample_length / sections, (section + 1) * sample_length / sections,
avg_y);
    var_z = variance(trial, sensor_num, 3, data, section *
sample_length / sections, (section + 1) * sample_length / sections,
avg_z);

    simple.print(avg_x + "," + avg_y + "," + avg_z + "," + var_x
+ "," + var_y + "," + var_z + ",");
    }
}

//AVERAGE
public static double average(int trial, int sensor_num, int
dimension, double[][][][] data, int start, int end) {
    double avg = 0;
    for (int sample = start; sample < end; sample++)
        avg += data[trial][sensor_num][dimension][sample];
    if (end - start <= 0) {
        avg = avg / 1;
        //System.out.println("avg warn on trial " + trial);
    } else
        avg = avg / (end - start);
    return avg;
}

//VARIANCE
public static double variance(int trial, int sensor_num, int
dimension, double[][][][] data, int start, int end, double average) {
    double var = 0;

```

```

        for (int sample = start; sample < end; sample++)
            var += (data[trial][sensor_num][dimension][sample] -
average) * (data[trial][sensor_num][dimension][sample] - average);
        if (end - start - 1 <= 0) { //in the event of a very small
segment
            var = 0;
            //System.out.println("var warn on trial " + trial);
        } else
            var = var / (end - start - 1);
        return var;
    }

    //COMPLICATED FEATURE EXTRACTION: FIND THRESHOLDS TO CREATE 4
SEGMENTS
    public static boolean feature_extraction(int subject, int trial,
double[][][][] data, int sample_length, int[] segments) {
        int sensor_num = 0; //using accelerometer

        int range = 12; //looks forward and back "range" samples
        double threshold = 8.5; //start high
        int segment = 1; //target is 4 segments (of 4 sections apiece;
16 total sections)
        int count = 0;

        //FIND THRESHOLD
        while (segment != 4) {
            boolean motion = false;
            segment = 1;

            //CHECK SAMPLES AGAINST THRESHOLD
            for (int sample = range; sample < sample_length - range;
sample++) {

                double avg_x = average(trial, sensor_num, 1, data,
sample - range, sample + range);
                double avg_y = average(trial, sensor_num, 2, data,
sample - range, sample + range);
                double avg_z = average(trial, sensor_num, 3, data,
sample - range, sample + range);

                double var_x = variance(trial, sensor_num, 1, data,
sample - range, sample + range, avg_x);
                double var_y = variance(trial, sensor_num, 2, data,
sample - range, sample + range, avg_y);
                double var_z = variance(trial, sensor_num, 3, data,
sample - range, sample + range, avg_z);

                double delta = var_x + var_y + var_z;

                //IF THRESHOLD TYPE CROSSED, CHANGE TYPE AND NOTE
CROSSING PLACE
                //odd segments for motion, even segments for stillness
                if (segment % 2 == 0 && delta > threshold) {

```

```

        motion = true;
        segments[segment] = sample; //save crossover point
        segment++;
    } else if (segment % 2 == 0 && delta <= threshold) {
        motion = false;
    } else if (segment % 2 == 1 && delta > threshold) {
        motion = true;
    } else if (segment % 2 == 1 && motion && delta <=
threshold) {
        // "motion == true" here to help discard initial
stillness of phone
        motion = false;
        segments[segment] = sample; //save crossover point
        segment++;
    }
}
if (segment > 4) threshold = 1.01 * threshold;
else if (segment < 4) threshold = 0.99 * threshold;
count++;
if (count > 9999) {
    System.out.println(trial+" extraction failed, leaving
blank"); //or reverting to simple
    return false;
}
}
segments[4] = sample_length;
return true;
}

public static void complex_sections(int subject, int trial,
double[][][][] data, String sensor, int sample_length, int[] segments,
PrintWriter complex) {
    int sensor_num = 0;

    //SCALE ACCELEROMETER VALUES TO GYROSCOPE
    if (sensor.equals("gyr")) {
        sensor_num = 1;
        segments[1] = segments[1] * sample_length / segments[4];
//segments[4] holds acc sample length
        segments[2] = segments[2] * sample_length / segments[4];
        segments[3] = segments[3] * sample_length / segments[4];
        segments[4] = segments[4] * sample_length / segments[4];
    }
    //debugger
    //else {
    //    System.out.println(subject+" "+trial+" "+segments[0]+"
"+segments[1]+" "+segments[2]+" "+segments[3]+" "+segments[4]);
    //}

    //NOW DIVIDE SEGMENTS INTO 4 APIECE, CALCULATE AVG/VAR IN EACH
SECTION
    for (int segment = 0; segment < 4; segment++)
        for (int section = 0; section < 4; section++) {

```

```

        if (subject >= 10 && subject <= 19 && sensor_num == 1) {
//Aliza's no GYR phone
            complex.print("0,0,0,0,0,0,");
            continue;
        }

        int length = segments[segment + 1] - segments[segment];
        double avg_x, avg_y, avg_z, var_x, var_y, var_z;

        //AVERAGE
        avg_x = average(trial, sensor_num, 1, data, section *
length / 4 + segments[segment], (section + 1) * length / 4 +
segments[segment]);
        avg_y = average(trial, sensor_num, 2, data, section *
length / 4 + segments[segment], (section + 1) * length / 4 +
segments[segment]);
        avg_z = average(trial, sensor_num, 3, data, section *
length / 4 + segments[segment], (section + 1) * length / 4 +
segments[segment]);

        //VARIANCE
        var_x = variance(trial, sensor_num, 1, data, section *
length / 4 + segments[segment], (section + 1) * length / 4 +
segments[segment], avg_x);
        var_y = variance(trial, sensor_num, 2, data, section *
length / 4 + segments[segment], (section + 1) * length / 4 +
segments[segment], avg_y);
        var_z = variance(trial, sensor_num, 3, data, section *
length / 4 + segments[segment], (section + 1) * length / 4 +
segments[segment], avg_z);

        complex.print(avg_x + "," + avg_y + "," + avg_z + "," +
var_x + "," + var_y + "," + var_z + ",");
    }
}

```

References

- [1] Goel, V., Lichtblau E. (2017). Russian agents were behind Yahoo hack, U.S. says. Retrieved from <https://www.nytimes.com/2017/03/15/technology/yahoo-hack-indictment.html>. March 15, 2017.
- [2] Nakashima, E. (2015). Hack of security clearance system affected 21.5 million people, federal authorities say. Retrieved from https://www.washingtonpost.com/news/federal-eye/wp/2015/07/09/hack-of-security-clearance-system-affected-21-5-million-people-federal-authorities-say/?utm_term=.055578d821bf. July 9, 2015.
- [3] Greenberg, A. (2016, February 16). Android Phone Hacks Could Unlock Millions of Cars. Retrieved from <https://www.wired.com/2017/02/hacked-android-phones-unlock-millions-cars/>
- [4] Apple confirms accounts compromised but denies security breach. (2014, September 2). Retrieved from <http://www.bbc.com/news/technology-29039294>.
- [5] Pagliery J. (2016, May 19). Hackers selling 117 million LinkedIn passwords. Retrieved from <http://money.cnn.com/2016/05/19/technology/linkedin-hack/>
- [6] O'Gorman, L. (2002). Securing Business's Front Door—Password, Token, and Biometric Authentication. *Technical Report ALR-2002-042*, Avaya Labs Research, Basking Ridge, NJ, 2002.
- [7] Risen T. (2014). Study: Hackers Cost More Than \$445 Billion Annually. Retrieved from <https://www.usnews.com/news/articles/2014/06/09/study-hackers-cost-more-than-445-billion-annually>. June 9, 2014.
- [8] Zviran, M., & Haga, W. J. (1999). Password security: an empirical study. *Journal of Management Information Systems* 15.4 (1999): 161-185.
- [9] Tom, P. L. (1991). *Managing information as a corporate resource*. HarperCollins Publishers.
- [10] Stoll, C. (2005). *The cuckoo's egg: tracking a spy through the maze of computer espionage*. Simon and Schuster.
- [11] Stoll, C. (1988) Stalking the willy hacker. *Communications of the ACM*, 31(5), 484-497.
- [12] Ungood-Thomas, J. (1998, March 29). The schoolboy spy. *Sunday Times (London)*, section 5, 1-2.
- [13] DiDio, L. (1998, April). Major hacks raise hackles, spur defenders. *ComputerWorld*, 32(13), 49-50.
- [14] DiDio, L. (1998, March). Cyberattack prompts DoD to boost security. *ComputerWorld*, 32(9), 14.

- [15] O’Gorman, L. (2003). Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE* 91.12 (2003): 2021-2040.
- [16] Weiss, K. P. (1988). Method and apparatus for positively identifying an individual. *U.S. Patent No. 4,720,860*. 19 Jan. 1988.
- [17] Advantages and Disadvantages of using tokens. (2011, December 24). Retrieved from <https://kamiavg.wordpress.com/introduction/advantages-and-disadvantages/>
- [18] Kaufman, C. & Perlman R. & Speciner M (2002). *Network security: private communication in a public world*. Prentice Hall Press, 2002.
- [19] Chown, P. (2002). Advanced encryption standard (AES) ciphersuites for transport layer security (TLS). No. RFC 3268. 2002.
- [20] Morris, R. & Thompson K. (1979) Password security: A case history. *Communications of the ACM* 22.11 (1979): 594-597.
- [21] Jobusch, D. L. & Oldehoeft A. E. (1989). A survey of password mechanisms: Weaknesses and potential improvements. *Computer Security*, vol. 8, no. 8, pp. 675–689, 1989.
- [22] Feldmeier, D. C., Karn, P. R. (1990). UNIX password security—ten years later. In *Advances in Cryptology—CRYPTO’89 Proc.*, 1990, pp. 44–63.
- [23] Bishop, M., & Klein, D. V. (1995). Improving system security via proactive password checking. *Computer Security*, vol. 14, no. 3, pp. 233–249, 1995.
- [24] Bunnell, J., Podd, J., Henderson, R., Napier, R., & Kennedy-Moffat, J. Cognitive, associative, and conventional passwords: Recall and guessing rates. *Computer Security*, vol. 16, no. 7, pp. 645–657, 1997.
- [25] Furnell, S. M., Dowland, P. S., Illingworth, H. M., & Reynolds, P. L. (2000). Authentication and supervision: A survey of user attitudes. *Computer Security*, vol. 19, no. 6, pp. 529–539, 2000.
- [26] Pond, R., Podd, J. Bunnell, J., & Henderson, R. Word association computer passwords: The effect of formulation techniques on recall and guessing rates. *Computer Security*, vol. 19, no. 7, pp. 645–656, 2000.
- [27] Jain, A., Bolle, R., & Pankanti, S. (1998). Eds., *Biometrics: Personal Identification in Networked Society*. Dordrecht, The Netherlands: Kluwer, Nov. 1998.
- [28] *Computer* (Special Issue on Biometrics: The future of identification), vol. 33, no. 2, pp. 46–80, Feb. 2000.

- [29] Weiss, G. M., & Lockhart, J. W. (2011). Identifying user traits by mining smart phone accelerometer data. *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*. ACM, 2011.
- [30] Kwapisz, J.R., Weiss, G.M., & Moore, S.A. 2010. Cell phone-based biometric identification. *In Proceedings of the IEEE 4th International Conference on Biometrics: Theory, Applications and Systems (BTAS-10)*, Washington DC.
- [31] Lockhart, J.W., & Weiss, G. 2011. Design considerations for the WISDM smart phone-based sensor mining architecture. *In Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, San Diego, CA.
- [32] Witten, I., & Frank, E. *Data Mining Practical Machine Learning Tools and Techniques*. San Francisco, Morgan Kaufmann Publishers, 2005.
- [33] Class J48. Retrieved from <http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html>
- [34] Weka Knowledge Explorer. Retrieved from http://www.cs.waikato.ac.nz/~ml/weka/gui_explorer.html
- [35] Lee, W. H., & Lee. R. B. (2015). Multi-sensor authentication to improve smartphone security. *Information Systems Security and Privacy (ICISSP), 2015 International Conference on IEEE*, 2015. Retrieved from <http://palms.princeton.edu/system/files/Multi-sensor+authentication+to+improve+smartphone+security.pdf>
- [36] Support Vector Machine (2017, August). Retrieved from https://en.wikipedia.org/wiki/Support_vector_machine
- [37] Xu, Z., Bai, K., and Zhu, S. (2012). Taplogger: Inferring user inputs on smartphone touchscreens using onboard motion sensors. *In Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*. WISEC '12. doi:10.1145/2185448.2185465
- [38] Spadafora A. (April 11, 2017). Nationwide tests behavioral biometrics for online banking. Retrieved from <http://www.itproportal.com/2016/04/11/nationwide-tests-behavioural-biometrics-online-banking>. .
- [39] Crawford. H. (2010). Keystroke dynamics: Characteristics and opportunities. *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference, IEEE*, 2010, pp. 205-212
- [40] Yampolskiy, R., Govindaraju, V., (2008). Behavioral biometrics: a survey and classification. *International Journal of Biometrics*, Inderscience Publishers, pp. 81-113, January 2008.

- [41] Carlson, C., Chen, C., Cruz, J., Maghsoudi, J., Zhao, H., Monaco, J. (2015). User Authentication with Android Accelerometer and Gyroscope Sensors. *Student-Faculty Research Day*, CSIS, Pace University, May 2015.
- [42] Guse, D. (2011). Gesture-based User Authentication on Mobile Devices using Accelerometer and Gyroscope. *Master's Report, Berlin Institute of Technology*, 2011.
- [43] Kenneth, N. (2015). Biometric User Authentication on Smartphone Accelerometer Sensor Data. Proc. *Student-Faculty Research Day*, CSIS, Pace University, May 2015.
- [44] Nowlan, M. F. (2015). Human Identification via Gait Recognition Using Accelerometer Gyro Forces. Retrieved from http://cs-www.cs.yale.edu/homes/mfn3/pub/mfn_gait_id.pdf, accessed October 2015 [Accessed: Oct- 2015]
- [45] Trewin, S., Swart, C., Koved, L., Martino, J., Singh, K., & Ben-David, S. (2012). Biometric Authentication on a Mobile Device. Proc. 28th Annual Computer Security Applications Conference on - ACSAC '12, 2012.
- [46] Password. (2017, March 17). Wikipedia. Wikimedia Foundation, n.d. Web. 23 Jan. 2017. Retrieved from <https://en.wikipedia.org/wiki/Password>.
- [47] Liu, S., & Silverman, M. (2002). A practical guide to biometric security technology. *IT Professional 3.1 (2001): 27-32 and - IEEE Xplore Document. IEEE*, 07 Aug. 2002. Web. 12 Apr. 2017. Retrieved from <http://ieeexplore.ieee.org/abstract/document/899930/>.
- [48] Security Token. Wikipedia. (2017, April 8). Wikimedia Foundation. Retrieved from https://en.wikipedia.org/wiki/Security_token.
- [49] Mobile Security | Mobile Device Security | Entrust. (n.d.). Retrieved from <https://www.entrust.com/solutions/mobile/>
- [50] What Is a Security Token? - Definition from Techopedia. (n.d.). Retrieved from <https://www.techopedia.com/definition/16148/security-token>.
- [51] Jain, A.k., Ross, A., & Prabhakar, S. (2004). An Introduction to Biometric Recognition. *IEEE Transactions on Circuits and Systems for Video Technology* 14.1 (2004): 4-20. Web
- [52] Watson, S. (2008, March 24). *How Fingerprinting Works*. HowStuffWorks Science. HowStuffWorks. Web. 12 Apr. 2017. Retrieved from <http://science.howstuffworks.com/fingerprinting1.htm>.
- [53] Maio, D., Maltoni, D., Cappelli, R., Wayman, J.L., and Jain, A. K. (2002). FVC2002: Fingerprint verification competition. In *Proc. Int. Conf. Pattern Recognition (ICPR)*, Quebec City, QC, Canada, Aug. 2002, pp. 744–747.

- [54] Roberts, J. J. (2016, September 12). Homeland Security Plans to Expand Fingerprint and Eye Scanning at Borders. Retrieved from <http://fortune.com/2016/09/12/border-security-biometrics>
- [55] Subramanian, K. (2015). Airport Security Technologies: Is It Improved after Terrorism? N.p., 19 Mar. 2015. Web. Retrieved from <https://www.linkedin.com/pulse/airport-security-technologies-improved-after-kishor-subramanian>.
- [56] Tappert et al. A Behavioral Biometrics User Authentication Study Using Multi-Sensor Android Devices. Retrieved from <http://csis.pace.edu/~ctappert/srd2016/2016PDF/c7.pdf>.
- [57] Coakley, M. J., Monaco, J. V., & Tappert, C. C. (2016). Keystroke biometric studies with short numeric input on smartphones. *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. doi:10.1109/btas.2016.7791181
- [58] Schiphol backs eye scan security. (2002). *CNN World News*. [Online]. Retrieved from: <http://www.cnn.com/2002/WORLD/europe/03/27/schiphol.security/>
- [59] Daugman, J., (1999). Recognizing persons by their Iris patterns. *In Biometrics: Personal Identification in a Networked Society*, pp. 103–121. doi:10.1007/0-306-47044-6_5
- [60] Lucian, C. (2017). AI-based Typing Biometrics Might Be Authentication's next Big Thing. *CIO. IDG News Service*, 27 Jan. 2017. Retrieved from <http://www.cio.com/article/3162392/security/ai-based-typing-biometrics-might-be-authentications-next-big-thing.html>.
- [61] Monroe, F. Keystroke Dynamics as a Biometric for Authentication. Retrieved from <http://www.cs.columbia.edu/4180/hw/keystroke.pdf>.
- [62] Joyce, R., Gupta, G. (1990). Identity authorization based on keystroke latencies. *Communications of the ACM* 33 (2) (1990) 168–176. doi:10.1145/75577.75582
- [63] The Role of Biometrics in IT Security and Continuous Authentication. Retrieved from <https://canvas.uw.edu/courses/1061960/files/36547087/download>.
- [64] Boyer, K. W., Govindaraju, V., & Ratha, N. K. (2007). Introduction to the special issue on recent advances in biometric systems. *IEEE Trans. Syst., Man, Cybern. B, Cybern.* vol. 37, no. 5, pp. 1091–1095, May 2007.
- [65] Crawford, H. Keystroke dynamics: Characteristics and opportunities. In *Privacy Security and Trust (PST). 2010 Eighth Annual International Conference on, 2010 (pp. 205-212): IEEE. IEEE Xplore*. Retrieved from <http://ieeexplore.ieee.org/document/5593258/>.
- [66] Adams, A. & Sasse, M. A. (1999). Users Are Not the Enemy. *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, December 1999.
- [67] Spillane, R. (1975). Keyboard Apparatus for Personal Identification. *IBM Technical Disclosure Bulletin, Tech. Rep.* 17, 1975.

- [68] Brown, M., & Rogers, S. (1996). Method and apparatus for verification of a computer user's identification, based on keystroke dynamics. *U.S. Patent Number 5,557,686*, September 17, 1996.
- [69] Garcia, J. (1986). Personal identification apparatus. *U.S. Patent Number 4,621,334*, November 4, 1986.
- [70] Young J., & Hammond, R. (1989). Method and apparatus for verifying an individual's identity. *U.S. Patent Number 4,805,222*, February 14, 1989.
- [71] Bergadano, F., Gunetti, D., & Picardi, C. (2002). User Authentication through Keystroke Dynamics. *ACM Transactions on Information and System Security*, vol. 5, no. 4, pp. 367–397, November 2002.
- [72] Bleha, S., Slivinski, C., & Hussien, B. (1990). Computer-Access Security Systems Using Keystroke Dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1217–1222, December 1990.
- [73] Monroe, F., & Rubin, A. (2000). Keystroke Dynamics as a Biometric for Authentication. *Future Generation Computer Systems*, vol. 16, pp. 351–359.
- [74] Cho, S., Han, C., Han, D., & Kim, H. (2000). Web based Keystroke Dynamics Identity Verification Using Neural Network. *Journal of Organizational Computing and Electronic Commerce*, vol. 10, no. 4, pp. 295–307.
- [75] Clarke, N., & Furnell, S. (2007). Authenticating Mobile Phone Users Using Keystroke Analysis. *International Journal of Information Security*, vol. 6, no. 1, pp. 1–14, January 2007.
- [76] 10 Hot Consumer Trends 2014. (2014). Retrieved from <http://www.ericsson.com/res/docs/2013/consumerlab/10-hot-consumer-trends-report-2014.pdf>
- [77] Fischer, A. & Plamondon, R. (2017). Signature Verification Based on the Kinematic Theory of Rapid Human Movements. *IEEE Transactions on Human-Machine Systems* 47.2 (2017): 169–80. Web
- [78] Plamondon, R., & Lorette, G. (1989). Automatic signature verification and writer identification—the state of the art. *Pattern Recognition*, vol. 22, no. 2 pp. 107–131.
- [79] Shimizu, H., Kiyono, S., Motoki, T., & Gao, W. (2004). An electrical pen for signature verification using a two-dimensional optical angle sensor. *Sensors and Actuators A: Physical*, 111(2), 216–221.
- [80] V. Wilson, T. (2005, November 11). How Biometrics Works. Retrieved from <http://science.howstuffworks.com/biometrics1.htm>
- [81] Facial Recognition. (2014). (n.d.). Retrieved from <http://findbiometrics.com/solutions/facial-recognition/>

- [82] Facial Recognition Biometric Time Clock. (2017). (n.d.). Retrieved from <http://biometrictimeclock.com/facial-recognition/>
- [83] Yan, Y., Chen, Q., Lee, F. (2016). Face Recognition Using Extended Vector Quantization Histogram Features. *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*.
- [84] Kotani, K., Chen, Q., & Ohmi, T. (2002). Face Recognition Using Vector Quantization Histogram Method. *Int'l Conf. on Image Processing*, vol.2, pp. 105-108, 2002.
- [85] Horiuchi, T., & Hada, T. (2013). A complementary study for the evaluation of face recognition technology. *47th International Carnahan Conference on Security Technology (ICCST)*, pp. 1-5, 2013.
- [86] Soldera, J., Behaine, C. A., & Scharcanski, J. (2015). Customized Orthogonal Locality Preserving Projections With Soft-Margin Maximization for Face Recognition. *IEEE Transactions on Instrumentation and Measurement*, 64(9), 2417-2426. doi:10.1109/tim.2015.2415012
- [87] Lone, M. A. Lone, Zakariya, S. M., & Ali, R. (2011). Automatic Face Recognition System by Combining Four Individual Algorithms. *Int'l Conf. on Computational Intelligence and Communication Networks (CICN)*, pp. 222-226, 2011.
- [88] He, X. F., Yan, S. C., Hu, Y. X., Niyogi, P., & Zhang, H. J.(2005). Face Recognition Using Laplacianfaces. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol.27, no.3, pp. 328-340, Mar.2005.
- [89] Mallipeddi, R., & Lee, M. (2012). Ensemble based face recognition using discriminant PCA Features. *IEEE Congress on Evolutionary Computation (CEC)*, pp.1-7, Jun. 2012.
- [90] Jelšovka, D., Hudec, R. & Brežňan, M. (2011). Face recognition on FERET face database using LDA and CCA method. *International Conference on Telecommunications and Signal Processing (TSP)*, pp.570-574, 2011.
- [91] Lei, J., Lu, C., & Pan Z. (2011). Enhancement of Components in ICA for Face Recognition. *International Conference on Software Engineering Research, Management and Applications (SERA)*, pp. 33-38, 2011.
- [92] Chen, J., Yuan, B., & Li, B. (2008). Face recognition based on LFA features and DLPP algorithm. *IET 2nd International Conference on Wireless, Mobile and Multimedia Networks (ICWMMN 2008)*, pp. 494-497, doi:10.1049/cp: 20081044
- [93] Qiao, H., Zhang, S. Y., Zhang, B., & Keane, J. (2004). Face Recognition using SVM Decomposition Methods. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'04)*, pp.2015-2020, 2004.
- [94] Face Recognition - Technology Overview. What is Face Recognition? (n.d.). Retrieved from <http://www.ex-sight.com/technology.htm>

- [95] Pike, J. (n.d.). Voice Verification. Homeland Security. Retrieved from <http://www.globalsecurity.org/security/systems/biometrics-voice.htm>
- [96] Sensors Overview. (2017, July 20). Retrieved from https://developer.android.com/guide/topics/sensors/sensors_overview.html
- [97] Goodrich, R. (2013, October 01). Accelerometers: What They Are & How They Work. Retrieved from <http://www.livescience.com/40102-accelerometers.html>
- [98] Goodrich, R. (2013, October 01). Accelerometer vs. Gyroscope: What is the Difference? Retrieved from <http://www.livescience.com/40103-accelerometer-vs-gyroscope.html>
- [99] Gyroscope. (2017, April 11). Retrieved from <https://en.wikipedia.org/wiki/Gyroscope>
- [100] Machine Learning. (2017, April 14). Retrieved from https://en.wikipedia.org/wiki/Machine_learning
- [101] Ray, S., Shaikh, F., & Kumar, S. (2017, May 04). Essentials of Machine Learning Algorithms (with Python and R Codes). Retrieved from <https://www.analyticsvidhya.com/blog/2015/08/common-machine-learning-algorithms/>
- [102] Distance and similarity measuring methods in Machine Learning. (2014, March 11). Retrieved from <https://lylelin317.wordpress.com/2014/03/11/distance-and-similarity-measuring-methods-in-machine-learning/>
- [103] Multilayer Perceptron. (2017, February 1). Retrieved from https://en.wikipedia.org/wiki/Multilayer_perceptron
- [104] Rosenblatt, F. (1961) Principles of Neurodynamic: Perceptron and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961 and No. VG-1196-G-8. CORNELL AERONAUTICAL LAB INC BUFFALO NY, 1961.
- [105] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. and the PDP research group (Editors). (1986). Learning Internal Representations by Error Propagation. (Editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations. MIT Press, 1986.
- [106] (n.d.). Multilayer Perceptron. Retrieved from <http://neuroph.sourceforge.net/tutorials/MultiLayerPerceptron.html>
- [107] Yegulalp, S. (2014, December 04). 11 open source tools to make the most of machine learning. Retrieved from <http://www.infoworld.com/article/2853707/robotics/11-open-source-tools-machine-learning.html#slide10>
- [108] Weka (machine Learning). (2017). Wikipedia. Wikimedia Foundation, n.d. Web. 22 Jan. 2017. Retrieved from [https://en.wikipedia.org/wiki/Weka_\(machine_learning\)](https://en.wikipedia.org/wiki/Weka_(machine_learning)). (2017, March 22).

- [109] Coakley, M. J. (2016). Keystroke biometric studies with short numeric input on smartphones (Unpublished doctoral dissertation). CSIS, Pace University
- [110] Monaco, J. V., Bakelman, N., Cha, S.-H., & Tappert, C. C. (2013). Recent Advances in the Development of a Long-Text-Input Keystroke Biometric Authentication System for Arbitrary Text Input. 2013 European. Intelligence Security Informatics Conference (EISIC), pp. 60–66, Aug. 2013.
- [111] Desk, E. N. (n.d.). (2015, July 28). Specifying an Accelerometer: Function and Applications. Retrieved from <http://insights.globalspec.com/article/1263/specifying-an-accelerometer-function-and-applications>.
- [112] Carlson, C., et al. (2015). User Authentication with Android Accelerometer and Gyroscope Sensors. Proceedings of Student-Faculty Research Day 2015, CSIS, Pace University.
- [113] Lee, J., et al. (2016). A Behavioral Biometrics User Authentication Study Using Multi-Sensor Android Devices. Proceedings of Student-Faculty Research Day 2016, CSIS, Pace University.
- [114] Maghsoudi, J. & Tappert, C. C. (2016). A Behavioral Biometrics User Authentication Study Using Motion Data from Android Smartphones. 2016 European Intelligence and Security Informatics Conference (EISIC). doi:10.1109/eisic.2016.047
- [115] Kumar, M., Mahesh123slideshre Follow. (2014, December 23). An ATM with IRIS Recognition. Retrieved from <https://www.slideshare.net/mahesh123slideshre/an-atm-with-an-iris-recognition-42962071>
- [116] Sensor Kinetics Pro: ABOUT INNOVENTIONS®. (n.d.). Retrieved from http://www.rotoview.com/sensor_kinetics_pro.htm
- [117] Sensor Kinetics: ABOUT INNOVENTIONS®. (n.d.). Retrieved from http://www.rotoview.com/sensor_kinetics.htm
- [118] Weka 3: Data Mining Software in Java. (n.d.). Retrieved from <http://www.cs.waikato.ac.nz/~ml/weka/>