

2016-04-27

# Investigating Gyroscope Sway Features, Normalization, and Personalization in Detecting Intoxication in Smartphone Users

Christina Jane Aiello  
*Worcester Polytechnic Institute*

Follow this and additional works at: <https://digitalcommons.wpi.edu/etd-theses>

---

## Repository Citation

Aiello, Christina Jane, "Investigating Gyroscope Sway Features, Normalization, and Personalization in Detecting Intoxication in Smartphone Users" (2016). *Masters Theses (All Theses, All Years)*. 388.  
<https://digitalcommons.wpi.edu/etd-theses/388>

This thesis is brought to you for free and open access by [Digital WPI](#). It has been accepted for inclusion in Masters Theses (All Theses, All Years) by an authorized administrator of Digital WPI. For more information, please contact [wpi-etd@wpi.edu](mailto:wpi-etd@wpi.edu).

# Investigating Gyroscope Sway Features, Normalization, and Personalization in Detecting Intoxication in Smartphone Users

by

Christina Aiello

A Thesis

Submitted to the Faculty

of the

**WORCESTER POLYTECHNIC INSTITUTE**

in partial fulfillment of the requirements for the

**Degree of Master of Science**

In

**Computer Science**

By

Christina Jane Aiello

April, 2016

Approved:

---

Major Advisor Professor Emmanuel Agu

---

Professor Lane Harrison, Thesis Reader

## Table of Contents

Acknowledgements .....	7
Abstract .....	8
1 Introduction .....	9
1.1 Alcohol Measures and Bodily Effects of Alcohol.....	9
1.2 Excessive Drinking and Alcohol Dependence.....	9
1.3 Drunk Driving .....	11
1.4 Smartphone Applications that Target Alcohol Abuse.....	11
1.5 Goals of this Thesis.....	12
2 Related Work .....	14
2.1 Intoxichck .....	14
2.2 Phone-based Gait Analysis to Detect Alcohol Usage .....	14
2.3 Mobile Social Tool that Moderates Drinking .....	15
2.4 Detecting Intoxication via Microsoft Smart Watch.....	15
2.5 SoberDiary: A Phone-based Support System for Assisting Recovery from Alcohol Dependence .....	16
2.6 First Version of AlcoGait Application .....	17
3 Methodology .....	17
3.1 Study Logistics and Procedure .....	18
3.1.1 Sensor-Impairment Goggles .....	18
3.1.2 Intoxicated Participants .....	20
3.1.3 Detecting Intoxication via Gyroscope and Accelerometer Data .....	21
3.2 Outlier Removal .....	25
3.3 Feature Normalization .....	26
3.4 User-Specific Classification Model Using Ensembling.....	27
4 AlcoGait System Design and Implementation .....	28

4.1	“Data Collector/Feature Extractor:” A Desktop Matlab Application for Data Collection and Normalization.....	28
4.1.1	Initial Sensor Data Collection Used to Extract Features From.....	30
4.1.2	Feature Generation .....	31
4.1.3	Data Normalization.....	32
4.2	Model Creation via Desktop Weka .....	32
4.3	AlcoGait Server.....	33
4.4	Android Application .....	33
4.4.1	Initial Application Setup.....	33
4.4.2	Real-Time In-App Classification .....	35
5	Analysis and Results.....	40
5.1	Initial Data Collection for Training the BAC Detection Model .....	40
5.2	Data Classification .....	40
5.2.1	Exploring Value of and Removal of Features.....	42
5.2.2	Including Additional Physical Attributes of Participants .....	43
5.2.3	Comparing Accuracy of Various Classifiers.....	43
5.2.4	Analysis of Sway Area and Sway Volume .....	46
5.3	Exploring the Concept of Maintaining One’s Walking Speed .....	47
5.4	Investigating Personalization .....	48
6	Conclusions and Future Work.....	49
	References .....	50
	APPENDIX A: Study Description / Consent Form for Data Collection.....	53
	APPENDIX B: Demographic Information for Study Participants.....	55
	APPENDIX C: Using Weka for Classification.....	56
	Classifying Data Using Weka .....	56
	APPENDIX D: Setting Up Windows IIS .....	58

APPENDIX E: Instructions to Run AlcoGait Thesis Project .....	59
How to Gather Data Via Matlab.....	59
Data Locations:.....	60
User profiles:.....	60
Raw data: .....	60
Generated features:.....	61
Sway area images: .....	61
Steps to Normalize Data.....	61
Steps to Classify Data (in Desktop Weka) .....	62
How To Get Model (.model file) From Desktop Weka to Put Into Java Application.....	63

## Table of Figures

Figure 1: "Past Month, Binge, and Heavy Alcohol Use among Full-Time College Students Aged 18-20" [32] ('binge drinking'=4+ drinks for women/5+ drinks for men in a single occasion, and 'heavy drinking'= 8+ per week for women/15+ per week for men) .....	10
Figure 2: Graph showing college freshmen drinking trends throughout the year [31] .....	10
Figure 3: Screenshots of the Intoxicheck application.....	14
Figure 4: Statistical page within the SoberDiary application.....	16
Figure 5: Screenshots of the first iteration of the AlcoGait Application .....	17
Figure 6: Flow diagram illustrating our entire data collection, feature extraction, and classifier-training process	18
Figure 7: Two pairs of Drunk Busters Goggles used in our data collection.....	19
Figure 8: Flow diagram illustrating our data collection process .....	20
Figure 9: Two of our participants wearing pairs of Drunk Busters Goggles.....	20
Figure 10: The three gyroscope axes in Android devices and how they correlate to the three axes of the body.	21
Figure 11: Sway Area and Volume Plots.....	22
Figure 12: Flowchart explaining the outlier-removal process.....	26
Figure 13: Diagrams showing AlcoGait project progression .....	28
Figure 14: Left image shows the Matlab Mobile terminal; right image shows Matlab Mobile settings .....	29
Figure 15: Screenshot of the features offered by our Data Collector/Feature Extractor, our desktop Matlab application .....	30
Figure 16: Screenshot showing some of the information that new study participants must enter .....	30
Figure 17: Screenshot of the Weka desktop application, with our data set loaded .....	33
Figure 18: Screenshot of initial setup screen .....	34
Figure 19: Screenshot of screen where sober walking data is gathered .....	35
Figure 20: Main screen displaying the results of our feature extraction and classification. ....	36
Figure 21: Flow diagram explaining the process of detecting walking and gathering sensor data .....	37
Figure 22: How activity detection affected battery life .....	38
Figure 23: These four graphs show the breakdown in various attributes of our participants.....	40
Figure 24: Graph showing the accuracies of the best-performing classifiers .....	42
Figure 25: The four confusion matrices for our four best-performing classifiers .....	46
Figure 26: Boxplots showing the increase in some of our gyroscope sway areas as a participant became more "intoxicated" .....	47

## Table of Tables

Table 1: Gyroscope features generated from our raw sensor data .....	23
Table 2: Accelerometer features generated from our raw sensor data .....	24
Table 3: Sample generated feature data from one of our participants, from walking done when participant was sober .....	31
Table 4: Sample of raw generated feature data versus normalized data .....	32
Table 5: P-Values and correlation coefficients for each classification feature .....	43
Table 6: Results of training various classifiers with our dataset .....	44
Table 7: Results of exploring personalization.....	48

## Acknowledgements

First I would like to thank Professor Emmanuel Agu for advising this project throughout the 2015-2016 school year, offering suggestions and advice regarding researching and developing this project. I would also like to thank Professor Lane Harrison for volunteering to be my substitute thesis reader last-minute.

Next I cannot thank Adriana Hera and Sergei Ponomarev enough for helping me better understand certain components of Matlab and setting up a server. The two of them offered lots of valuable information regarding those two topics, and I thank them for that assistance.

I would also like to thank Michael Voorhis, Will Temple, and Ethan Paul for offering me guidance regarding Windows servers and IIS, help that allowed me to overcome a huge roadblock in the development of this thesis.

Lastly I would like to thank all of my study participants. The data that they helped me gather is invaluable, and I truly appreciate their time.



## Abstract

Alcohol abuse is the third leading lifestyle-related cause of death for individuals in the United States, causing 88,000 deaths each year in the United States from 2006-2010. Existing smartphone applications allow users to manually record their alcohol consumption or take cognitive tests to estimate intoxication levels; however, no smartphone application passively determines one's level of intoxication. After gathering smartphone sensor data from 34 "intoxicated" subjects, we generated time and frequency domain features such as sway area (gyroscope) and cadence (accelerometer), which were then classified using a supervised machine learning framework. Other novel contributions explored include feature normalization to account for differences in walking styles and automatic outlier elimination to reduce the effect of accidental falls by identifying and removing the top and bottom of a chosen percentage of the data. Various machine learning classifier types such as Random Forest and Bayes Net were compared, and J48 classifier was the most accurate, classifying user gait patterns into BAC ranges of [0.00-0.08), [0.08-0.15), [0.15-0.25), [0.25+) with an accuracy of 89.45%. This best performing classifier was used to build an intelligent smartphone app that will detect the user's intoxication level in real time.

## 1 Introduction

Substance abuse has become a public health issue, starting with the invention of beer by the Mesopotamians around 5000 BCE [8]. Alcohol is the most harmful substance abused, causing physical harm, increased mortality, and mental malfunction [10]. Consequences of improper alcohol use include damage to one's body, death, and in cases of drunk driving the death of others. Immediate effects of alcohol include slurred speech, delayed thinking, and impaired neuromotor functions [15].

### 1.1 Alcohol Measures and Bodily Effects of Alcohol

The heart and brain are most immediately affected as one's Blood Alcohol Content (BAC) rises. BAC is the "amount of alcohol in a person's body is measured by the weight of the alcohol in a certain volume of blood" [27]. BAC can be determined using a mathematical model, one example of which being the Widmark equation (equation 1) for achieving such a task, published in 1932 by E.M.P. Widmark [29]:

$$\text{Blood Alcohol Content} = \frac{A}{rW} - (\beta t) \quad (\text{Equation 1})$$

In equation 1,  $A$  is the mass of alcohol consumed,  $r$  and  $\beta$  are constants, and  $W$  is the individual's body weight. One's heart rate increases approximately ten minutes after consuming alcohol, and this increased heart rate is the body's attempt at filtering the alcohol out of one's system through the kidneys. Alcohol penetrates the blood-brain barrier approximately twenty minutes after alcohol consumption, impacting neuromotor and cognitive functions [25]. Gait, which is a coordinated effort between the body and brain to produce movement to move to a new location, is one of the neuromotor functions affected by alcohol consumption. The ability to walk or run is directly affected by one's alcohol consumption.

### 1.2 Excessive Drinking and Alcohol Dependence

Excessive drinking includes binge drinking (defined as 4 or more drinks for women on a single occasion, and 5 or more drinks for men on a single occasion) and heavy drinking (8 or more drinks per week for women and 15 or more drinks per week for men) [19]. The Substance Abuse and Mental Health Services Administration (SAMHSA) surveyed full time college students aged 18-20 between 2002 and 2005 to gather information regarding their alcohol use in the past month, binge alcohol use, and heavy alcohol use. Figure 1 displays these results, showing that over a third of men and women reported binge drinking and over a tenth reported heavy alcohol usage [32].

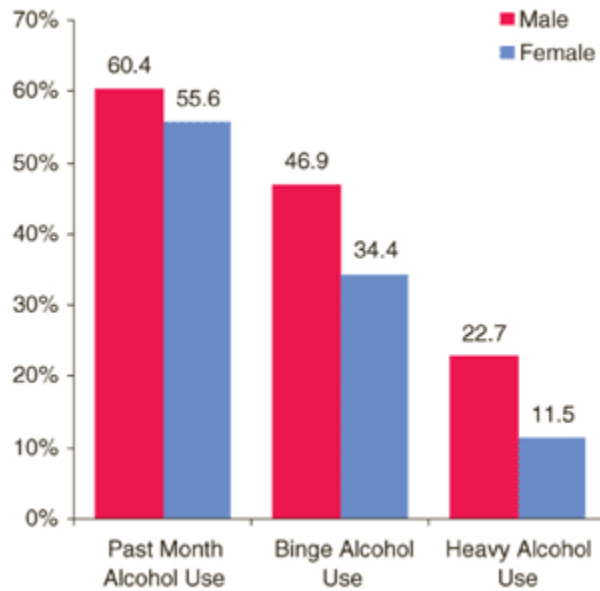


Figure 1: "Past Month, Binge, and Heavy Alcohol Use among Full-Time College Students Aged 18-20" [32] ('binge drinking'=4+ drinks for women/5+ drinks for men in a single occasion, and 'heavy drinking'= 8+ per week for women/15+ per week for men)

The National Institute on Alcohol Abuse and Alcoholism (NIAAA) reports that students tend to consume larger amounts of alcohol during certain key periods in the year, such as Thanksgiving week, the weeks of Christmas and New Year's, and during the week of Spring Break. The Figure 2 depicts these trends that occur throughout the year [31].

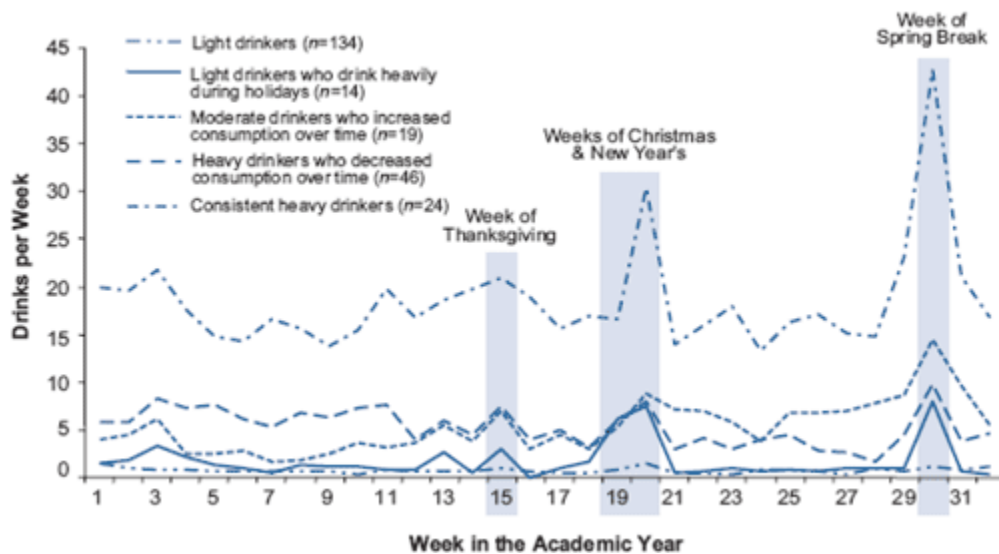


Figure 2: Graph showing college freshmen drinking trends throughout the year [31]

Between the years 2006 and 2010, each year excessive drinking was responsible for 88,000 deaths in the United States, 1 in 10 deaths among working-age adults aged 20-64 years [19]. Even when treated for alcohol dependence, over 50% of patients with alcohol dependence relapse two years after treatment [12].

### **1.3 Drunk Driving**

Drunk driving is a problem that endangers not only the intoxicated driver but also pedestrians and other drivers on the road (who are sober). The Bureau of Transportation Statistics reported in 2010 that 47.2% of pedestrian fatalities and 39.9% of vehicle occupant fatalities were caused by drunk driving [18]. The Centers for Disease Control and Prevention (CDC) reports that nearly 30 people die per day in drunk-driving-related motor vehicle crashes in the United States, meaning that approximate every 51 minutes an individual is killed in an impaired-driving accident. The issue of drunk driving even affects children, with the CDC reporting in 2013 that 200 children ages 0 to 14 were killed due to someone's impaired driving [20].

### **1.4 Smartphone Applications that Target Alcohol Abuse**

With a projection of approximately 2.08 billion individuals owning smartphones in the year 2016, smartphone application developers have the ability to leverage phones to monitor and improve many individuals' health [26]. Existing smartphone applications targeting alcohol abuse allow users to manually record their alcohol consumption, estimate BAC levels using built-in formulas, and present manual cognition tests to assess users' intoxication levels. The Smartphone Gait Inference Android Application, AlcoGait, researched developed by Zach Arnold and Danielle LaRose of Worcester Polytechnic Institute (WPI) [15] classifies features generated from a smartphone's accelerometer data to infer user drink levels using a machine learning approach. The best performing classifier was used to develop an Android application that estimated user intoxication levels from their gait in real time; however the application only had a prediction accuracy of approximately 56%, the application tried to estimate number of drinks rather than BAC, and the application did not attempt to normalize data per participant.

An iPhone application, Intoxicheck, presents users with challenges/tests to infer intoxication levels. While this application successfully detected intoxication, the need for users to actively interact with the application may be a deterrent. Kao et al. [9] designed a passive phone-based system that used the smartphone's accelerometer data to generate gait data, data which was then used in an attempt to detect whether users had consumed alcohol or not. These researchers did not try to estimate how much was consumed, just whether or not a user had consumed alcohol. Tjondronegoro et al. [21] details the authors' smartphone application developed to encourage positive drinking habits in users. While this application did not make any attempts to estimate a

user's BAC or number of drinks consumed, it surveyed individuals to understand what user-interactive features would be desired in an application that moderates drinking. Gutierrez et al. [18] passively estimated if individuals wearing smart watches were intoxicated using smartwatch sensor data. This application did not attempt to estimate a user's BAC, rather, it simply tried to detect if one's BAC was at or above 0.065. Trying to specifically detect a user's BAC or categorize one's BAC into more bins could provide more useful information to the user. Lastly, Wang et al. [28] developed SoberDiary, a phone-based support system that lets individuals recovering from alcohol addiction self-manage and self-monitor their alcohol consumption over time. SoberDiary helped patients maintain sobriety by significantly decreasing patients' total alcohol consumption, number of drinks, and heavy drinking days by 96.5%, 82.3%, and 86.5%, respectively; however the application both requires heavy user interaction and the possession of a portable Bluetooth-enabled breathalyzer. An application that does not require interaction from the user or a Bluetooth-enabled breathalyzer could potentially increase the number of individuals who use the application due to more simplicity and less effort required.

## 1.5 Goals of this Thesis

The first version of the AlcoGait Smartphone Gait Inference application [15] had the following limitations that were addressed by this thesis:

1. *Explore Gyroscope features:* Time and frequency domain features were generated based solely on accelerometer sensor data. This thesis explores the use of features generated from the smartphone's gyroscope as well as its accelerometer. The accelerometer measures the acceleration force in  $m/s^2$  applied to a device on all three axes (x, y, and z) including gravity, whereas the gyroscope measures a device's rate of rotation in rad/s around each of the three axes (x, y, and z) [4].
2. *Expanding classification attributes:* Including more information about participants in our classification model, such as gender and height, to potentially increase classification accuracy.
3. *Gather data from more users:* The previous AlcoGait work gathered smartphone accelerometer data from only seven participants, whereas in this thesis we attempted to gather more gait data. With the time given, data was gathered from 34 participants in via sensor-impairment goggles that simulate the effects of intoxication.

4. *Exploring feature Normalization:* to reduce inaccuracies caused by different walking styles

5. *Investigating personalization (machine learning models per-user) and ensembling:* which combines personalized and general models.

and

6. *Alcohol measured using BAC Level rather than number of drinks:* This first version of the AlcoGait Android application had a classification accuracy of 56% for the task of classifying the number of drinks a user had consumed into 3 bins: 0-2 drinks (sober), 3-6 drinks (tipsy) and > 6 drinks (drunk). A major overarching goal of this thesis is to improve on this accuracy to a clinically viable level by classifying based on BAC rather than number of drinks consumed.

If successful, AlcoGait could be integrated into a health care system for individuals in alcohol abuse therapy. Patients can continuously track their alcohol consumption with minimal burden. In future, AlcoGait can be used to prevent alcohol abuse by notifying users (e.g. pop-up message, SMS text, email, or phone call) when they have reached a certain intoxication level.

## 2 Related Work

Existing smartphone applications targeting alcohol abuse allow users to manually record their alcohol consumption, estimate Blood Alcohol (BAC) levels using built-in formulas, and offer manual cognition tests to assess users' intoxication levels. Other applications do not estimate one's BAC but attempt to encourage positive drinking habits in users. Researchers also developed an application for Microsoft Smart Watches [18] that will passively detect one's intoxication levels; however this application relies on heart rate and temperature (data unavailable to smartphones) rather than the gyroscope and accelerometer sensors.

### 2.1 Intoxichack

The smartphone application "Intoxichack" can detect alcohol impairment in users [5]. Users take a "series of reaction, judgment and memory challenges before s/he starts drinking" to create a baseline of the user's performance. The user repeats these challenges after consuming alcohol, and the results are compared to the baseline to estimate their intoxication level. Evaluation showed that the application "provided a reasonably accurate assessment of a person's impairment level" [5]. However, Intoxichack requires manual supervision, which may deter users from using it and reduce its scalability.



Figure 3: Screenshots of the Intoxichack application

### 2.2 Phone-based Gait Analysis to Detect Alcohol Usage

Kao et al. [9] designed a passive phone-based system that used the smartphone's accelerometer data to detect whether users had consumed alcohol or not, but these researchers did not try to estimate how much was consumed. Baseline data was gathered from the 3 participants when they were sober, and after consuming wine to bring the participants' BAC above 0.05, participants were asked to walk 40 meters while accelerometer data was recorded. This research was a proof-of-concept that one's step becomes longer (in distance and in time) when one is intoxicated, and more data may have been gained had they recruited more participants.

### 2.3 Mobile Social Tool that Moderates Drinking

Tjondronegoro et al. [21] details a mobile social smartphone application developed to encourage positive drinking habits in users. The application has many components including: The ability to create and share social drinking events (in an effort to promote user socialization and a group effort to monitor each other's drinking), a place to record how many drinks a user has consumed, a call option to contact cabs, the police, and other individuals, and an "I'm home!" feature to let others know when a user has safely arrived home after a night of drinking. While focus group participants disliked the idea of manually inputting how many drinks one has consumed, these individuals were interested in the "I'm home!" feature and the quick dial feature. The application also has the ability to account for one's drinks and share one's drink count with others. While the focus group participants weren't interested in sharing their drinking data with all of their Facebook friends, they were interested in sharing the data with a select group of people. The two individuals who tested the application noted that letting others know how much they'd had to drink, in addition themselves noting when they start a new drink, made them feel more responsible for what they were consuming. This application did not make any attempts to estimate a user's BAC or number of drinks consumed, however it surveyed individuals to understand what user-interactive features would be desired in an application that moderates drinking.

### 2.4 Detecting Intoxication via Microsoft Smart Watch

Gutierrez et al. [18] passively estimated if individuals wearing smart watches were intoxicated using smartwatch sensor data. The researchers had five participants in their study, each wearing a Microsoft Band smartwatch. These participants consumed enough alcohol over a two-hour period to reach a BAC of 0.08, and during this time the participants were breathalyzed and data was gathered from their smartwatches at three-second intervals. [18] These investigators did not use accelerometer and gyroscope data from the smartwatch (unlike AlcoGait, which relies solely on these two sensors); rather, they focused on the participants' heart rate and temperature. While their initial research provided them with 233,538 samples from the five participants, a point they make is that "Ideally, [they] would want a data set from a thousand volunteers in candid situations over several days" [18]. In terms of correctly predicting intoxication, the model achieved a " $R^2$  value of  $0.524 \pm 0.015$ " [18], where  $R^2$  is a statistical measure of how close data is to a fitted regression line and a higher  $R^2$  value is preferred. They also explored an SVM model, achieving a precision (proportion of instances that are actually of a class divided by the total instances classified as that class) of  $0.886 \pm 0.002$ , recall (proportion of instances classified as a certain class divided by the actual total in that class) of  $0.930 \pm 0.002$ , and F-score (a combined measure for precision and recall calculated as " $(2 * \text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$ ") of  $0.907 \pm 0.001$ . One component that this project did not contain was an attempt to specifically detect a user's BAC from



walking; it simply tried to detect whether or not the participant was sober or drunk (“drunk” being a BAC of 0.065 or greater).

## 2.5 SoberDiary: A Phone-based Support System for Assisting Recovery from Alcohol Dependence

Wang et al. [28] developed SoberDiary, a phone-based support system that lets individuals recovering from alcohol addiction self-manage and self-monitor their alcohol consumption over time. The application has three components: a portable Bluetooth breathalyzer that sends BAC levels to one’s smartphone via Bluetooth, a smartphone application, and a backend server. These researchers conducted a four-week study with eleven patients, having all of them use the SoberDiary application. This application was able to help patients maintain sobriety by significantly decreasing patients’ total alcohol consumption, number of drinks, and heavy drinking days by 96.5%, 82.3%, and 86.5%, respectively. The application also let users see how they ranked in comparison to other patients within the study, in addition to offering motivational incentives that were rewarded when a user maintained sobriety and/or used the application. SoberDiary was effective in its goal of encouraging safer drinking habits; however the application both requires frequent user interaction and the possession of a portable Bluetooth-enabled breathalyzer.

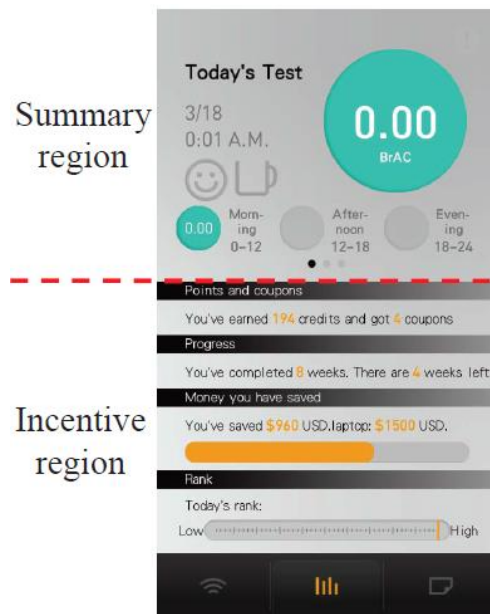


Figure 4: Statistical page within the SoberDiary application

## 2.6 First Version of AlcoGait Application

The first version of the AlcoGait application, created by Zachary Arnold and Danielle LaRose, can be seen in Figure 5 below. This application, like the current version of AlcoGait, also attempted to classify users' gait in real time; however this application only used accelerometer data rather than using both accelerometer and gyroscope data. In addition, this application tried to estimate the number of drinks a user had consumed rather than a BAC range. Finally, the application also did not attempt to normalize data before classification, and the use of normalization could have addressed individuals' differing walking patterns. The image below on the left shows the inference that the application made in regards to a user's intoxication level, and the image on the right shows the setup page from the application.

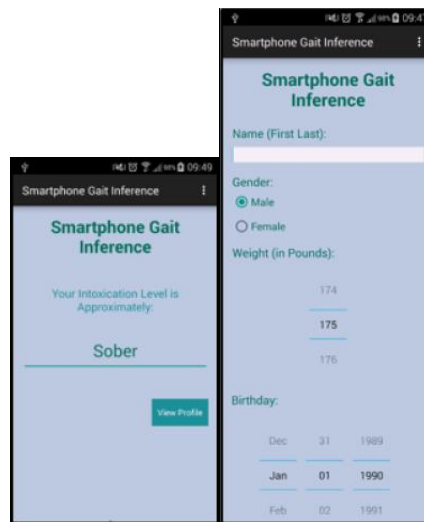


Figure 5: Screenshots of the first iteration of the AlcoGait Application

## 3 Methodology

In building upon the previous AlcoGait work, this thesis recruited 34 participants. To improve accuracy, we have included smartphone gyroscope features that are sensitive to alcohol consumption in addition to smartphone accelerometer data. We also explored feature normalization per participant and remove outliers to reduce inaccurate predictions. Finally, we allow the user to create a custom classification model using ensembling. These plans are detailed in Figure 6 below.

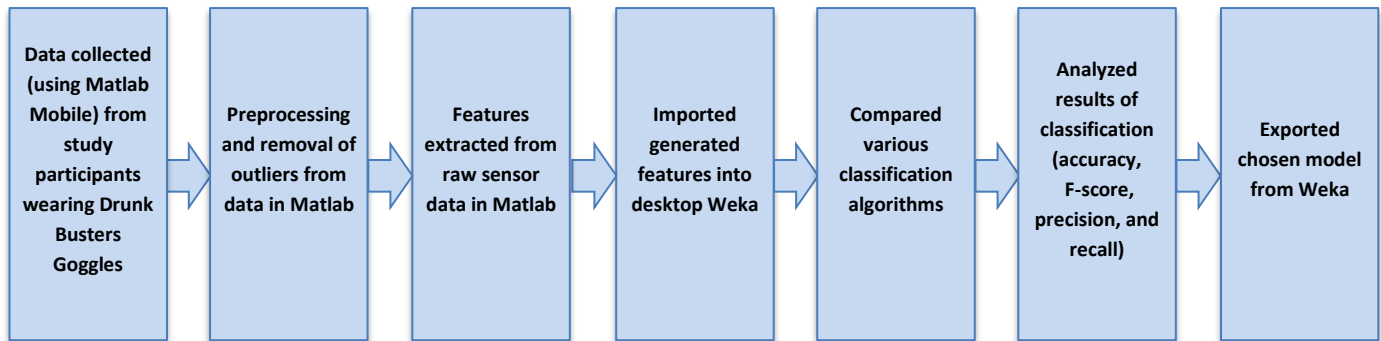


Figure 6: Flow diagram illustrating our entire data collection, feature extraction, and classifier-training process

### 3.1 Study Logistics and Procedure

Machine learning requires large amounts of data, and we believe that having a larger sample size will improve the application’s accuracy. We recruited 34 participants (14 male and 20 female) to improve the application’s classification accuracy. These participants were recruited via WPI’s Social Science Research Participation System, the SONA System, in addition to recruitment via email advertisements, social media advertisements, and word-of-mouth. The SONA system is a pool of psychology students who receive academic credit for participation in user studies.

#### 3.1.1 Sensor-Impairment Goggles

Students recruited via the SONA System were asked to wear pairs of sensor-impairment goggles while accelerometer and gyroscope sensor data was collected. These “Drunk Busters” goggles use vision distortion to “simulate the effects of alcohol consumption on the body,” [6]. Various Blood Alcohol Concentration (BAC) configurations simulate levels ranging from 0.04 to extremely high impairment at 0.35. The pairs we used were of the ranges [0.00-0.08), [0.08-0.15), [0.15-0.25), [0.25+). Wearers of these goggles are supposed to “experience the effects of reduced alertness, delayed reaction time, confusion, visual distortion, alteration of depth and distance perception, reduced peripheral vision, double vision, and lack of muscle coordination,” [6]. These goggles have been used by various groups to educate individuals regarding the effects of alcohol on one’s motor skills, however they have not been used in any published research studies. Regarding the scientific accuracy of these goggles, the creators of these goggles have said, “Our testing has been done in house, with nothing available to release to the public. It is instructive to note that each alcohol goggle simulates an approximate BAC range, there is no way anybody can design a goggle that simulates an exact BAC level, as 20 people at the exact BAC level would all behave a bit differently with varying test results of their physical behavior and such.”



Figure 7: Two pairs of Drunk Busters Goggles used in our data collection

The study procedure was as follows: Once participants signed the letter of consent, the investigator had the participants place an Android smartphone (provided by the investigator) in either the participants' front or back pants pocket. The participants then walked normally (no impairment) for around a minute and thirty seconds at whatever speed was comfortable/typical for the individual, and the application records smartphone's gyroscope and accelerometer data while the participant walks. Our Matlab application records five seconds of data in what is considered one "trial," and twelve trials (producing 60 seconds of data total) were completed. The participants were told that they did not need to walk in any specific patterns or at any particular speed, and this was to simulate a "real world" scenario where an individual would walk however he or she felt most comfortable rather than being told to walk at a specific pace or along a particular path.

The participants were then given various pairs of visual impairment goggles to wear. While wearing each pair of goggles one at a time, the participants were asked to walk again for one and a half minutes while gyroscope and accelerometer data is recorded. While the participants walked with the goggles, an investigator offered to walk with him or her to "spot" the individual if need be (be a guide, offer a steadying hand, etc.). If the participants felt too dizzy or about to fall or for any other reason, they were allowed stop the study by removing the goggles. Participants could also take breaks in between wearing various pairs of goggles. Participants were able to opt-out of the study at any point. If they did opt out, they were given the option to either allow their data, up to that point, to remain in the study or have their data permanently deleted and removed from the study. This data collection process can be seen in the flow diagram below.

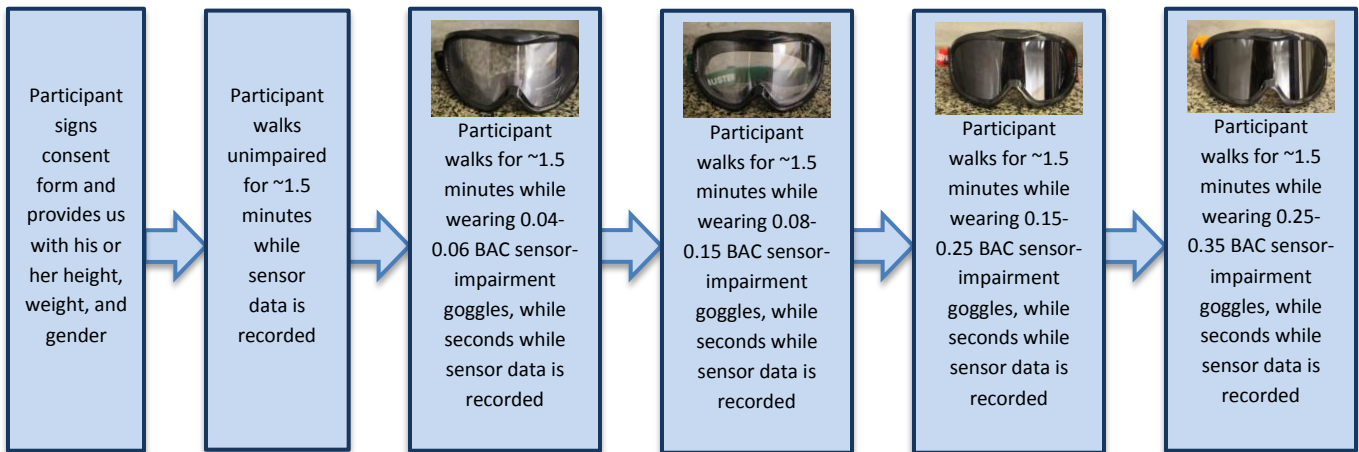


Figure 8: Flow diagram illustrating our data collection process

In addition to the aforementioned study procedure, we also investigated the possibility that requiring individuals to walk at relatively the same speed for every set of trials could provide more accurate results. This study procedure still involved recording data while the participant walked normally and while wearing each set of goggles, however we added the requirement that however quickly he or she walked while “sober” was the same speed that he or she must walk while “drunk.” This was achieved by having the participant walk for a set of trials while “sober” throughout the hallways in an academic building on WPI’s campus, and once the trials were completed a marker was placed on the ground in the hallway. The participants then had to walk through the hallways to that marker in the same amount of time while wearing each pair of sensor-impairment goggles.



Figure 9: Two of our participants wearing pairs of Drunk Busters Goggles

### 3.1.2 Intoxicated Participants

A study we constructed that involved working with intoxicated participants was presented to the Worcester Polytechnic Institute Institutional Review Board (IRB), and unfortunately a version of this study could not be

agreed upon in the given timeframe for this thesis. The experimenter planned to administer a breathalyzer test to each participant to acquire more accurate information related to BAC because not all individuals accurately know what constitutes one “drink,” and some individuals may inaccurately recall exactly how much they had consumed. While this study was not presently approved due to concerns regarding liability and student safety, the researchers hope this version of the study could be done in the future by other researchers for this project.

### 3.1.3 Detecting Intoxication via Gyroscope and Accelerometer Data

This thesis gathered gyroscope data and explores gyroscope features in addition to the accelerometer features previously explored in AlcoGait [15]. The accelerometer is a piece of hardware that measures the "acceleration force in  $m/s^2$  that is applied to a device on all three physical axes (x, y, and z), including the force of gravity," [4]. The gyroscope is another sensor component of Android smartphones, and this sensor “measures a device's rate of rotation in  $rad/s$  around each of the three physical axes (x, y, and z),” [4]. The gyroscope’s x, y, and z axes can be directly related to the three body axes, which are the mediolateral, anteroposterior, and superoinferior axes, respectively (Figure 5). Data was collected using an application developed in Matlab. The system allows users to collect new data and generate features based on previously-recorded accelerometer and gyroscope data. Data was collected in 5-second intervals and used to generate features.

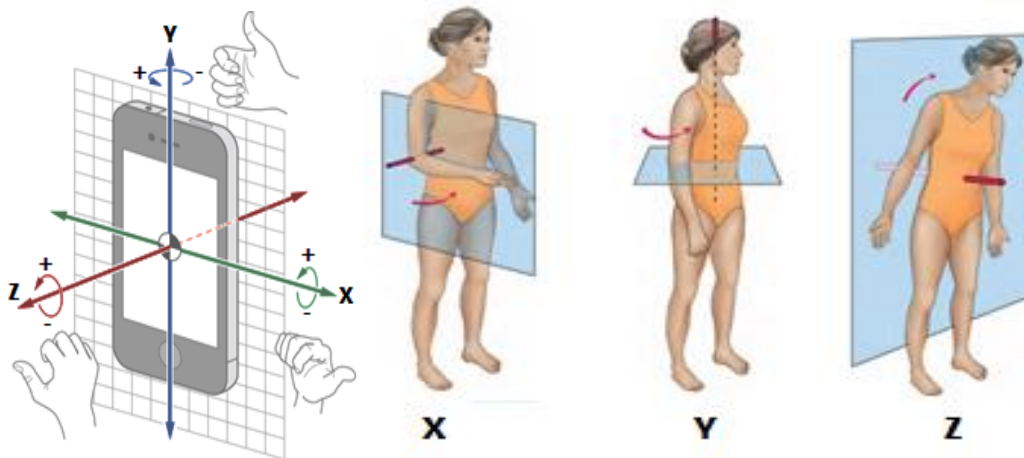


Figure 10: The three gyroscope axes in Android devices and how they correlate to the three axes of the body.

Nieschalk *et al.* [3] measured the Blood Alcohol Concentration (BAC) before test subjects were given various postural tasks to complete. A gyroscope feature called “sway area” (see figure 5) was then calculated by plotting points from two of the axes of the gyroscope. The researchers found that “sway area was the most sensitive parameter for detecting increased body sway after alcohol ingestion.” [3] This information influenced our decision to gather gyroscope data and investigate sway area in this thesis. The concept of sway volume (figure

6), a 3D extrapolation of sway area is our own contribution, which we hope will further improve our accuracy in detecting intoxication. Sway volume plots the (x, y, z) values gathered by the sensors and then determines the overall volume of those points when plotted.

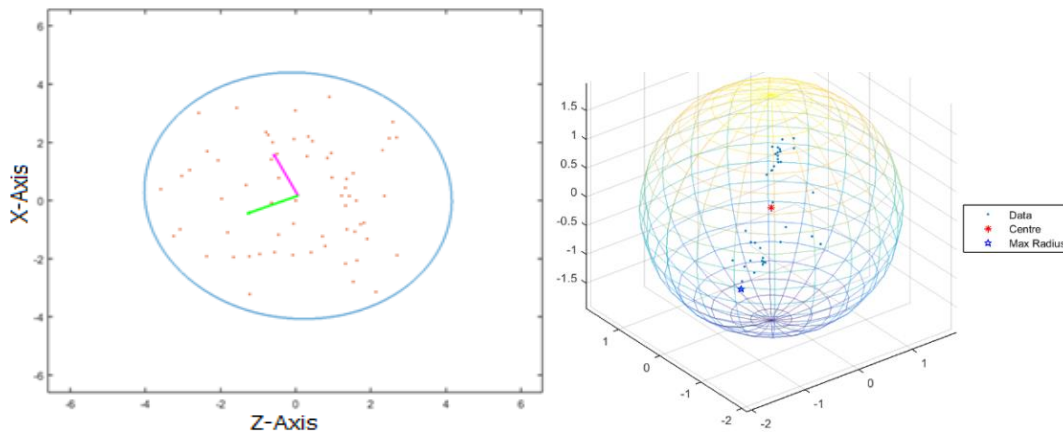


Figure 11: Sway Area and Volume Plots

Prior work by He and Agu [16] also found that combining both accelerometer and gyroscope features improves the accuracy of activity detection (jogging, walking, ascending or descending stairs, and sitting) on smartphones. In this work, we hope that combining the accelerometer and gyroscope will improve the estimation of user intoxication levels from their gait.

Once we gathered raw sensor data from participants who walked at various impairment levels, we generated features from that raw sensor data. The features we generated, and their formulas for doing so, are seen in the Table 1 and Table 2 below. The first column of each table contains the name of the feature. The second column has a description of what that feature is. The third column shows how we calculated this feature given the raw sensor data, and the fourth column cites where that feature was referenced from or if it was our own contribution.

The first table below details the four features that we generated from the gyroscope data, and the second table explains the accelerometer features we generated via work done by previous researchers Zach and Danielle when developing the first iteration of the AlcoGait application [15].

### Features Generated from Gyroscope Data

Feature Name	Feature Description	Formula	Reference
XZ Sway Area	Area of plotted gyroscope readings from Z (yaw) and X (pitch) axes	$XZ \text{ Sway Area} = \pi r^2$	[3]
YZ Sway Area	Area of plotted gyroscope readings from Z (yaw) and Y (roll) axes	$YZ \text{ Sway Area} = \pi r^2$	Own contribution
XY Sway Area	Area of plotted gyroscope readings from X (pitch) and Y (roll) axes	$XY \text{ Sway Area} = \pi r^2$	[13, 14]
Sway Volume	Volume of plotted gyroscope readings from all three axes (pitch, roll, and yaw)	$Sway \text{ Volume} = \frac{4}{3}\pi r^3$	Own contribution

**Table 1: Gyroscope features generated from our raw sensor data**



### Features Generated from Accelerometer Data

Feature Name	Feature Description	Calculation	Ref.
Steps	Number of steps taken	calculation of signal peaks above one standard deviation away from mean of gravity corrected magnitude of signal [15]	[15]
Cadence	Number of steps taken per minute	$cadence = \frac{\# \text{ steps}}{\text{minute}}$	[15]
Skew	Lack of symmetry in one's walking pattern	$skewness = \frac{\frac{1}{n} \sum (x_i - \mu_x)^3}{\left[ \frac{1}{n} \sum (x_i - \mu_x)^2 \right]^{3/2}}$ Where $x_i$ is the data sequence, and $\mu_x$ is the average of all $x_i$ [33]	[15]
Kurtosis	Measure of how outlier-prone a distribution is	$kurtosis = \frac{\frac{1}{n} \sum (x_i - \mu_x)^4}{\left[ \frac{1}{n} \sum (x_i - \mu_x)^2 \right]^2}$ Where $x_i$ is the data sequence, and $\mu_x$ is the average of all $x_i$ [33]	[15]
Average gait velocity	Average steps per second divided by average step length	average gait velocity = $\frac{(\text{average steps} / \text{sec})}{\text{step length}}$	[15]
Residual step length	Difference from the average in the length of each step	residual step length = $\frac{\text{distance}}{\# \text{ steps}}$	[15]
Ratio	Ratio of high and low frequencies	$harmonic \text{ ratio} = \frac{\sum_{l=1,3,5,\dots} V_l}{\sum_{j=2,4,6,\dots} V_j}$ Where $V_i$ is the amplitude of odd-ordered harmonic frequency and $V_j$ is the even-ordered harmonic frequency [33]	[15]
Residual step time	Difference in the time of each step	$residual \text{ step time} = \frac{\sqrt{\frac{1}{n} \sum (interval_i - \mu_{interval})^2}}{\mu_{interval}}$ Where $interval_i$ is a sequence of stride intervals and $\mu_{interval}$ is average of all $interval_i$ [33]	[15]
Bandpower	Average power in the input signal	bandpower = bandpower(x) Where x is a matrix of the magnitude, and bandpower calculates the average power in each column independently [34]	[15]
Signal to noise ratio	Estimated level of noise within the data	$snr = \frac{power_{signal}}{power_{noise}}$ [33]	[15]
Total harmonic distortion	"Determined from the fundamental frequency and the first five harmonics using a modified periodogram of the same length as the input signal" [30]	$THD_F = \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + V_5^2 + V_6^2}}{V_1}$ Where $V_1$ is energy contained within peak of PSD at the fundamental frequency and $V_i$ are the energy contained within the harmonics [15]	[15]

Table 2: Accelerometer features generated from our raw sensor data

### 3.2 Outlier Removal

If a user briefly trips while walking or bumps his/her phone while in his/her pocket, this will cause outliers in our data set, and these outliers could affect our classification accuracy. The goal of removing outliers was to improve the accuracy of our classification process. Our outlier-removal algorithm removes a specific percent of outliers from accelerometer and gyroscope data per plane of data. The algorithm takes in a matrix of data (as [x-plane, y-plane, z-plane] triples, with one reading per plane) and a percent to be removed from both the top and the bottom of the data (per plane). First the method calculates how many rows (we will call this “W” number of rows) should be removed from the top and bottom based on the size of the matrix and the percentage provided. Next, it sorts the data by the x-values and removes “W” rows from the top and bottom of the sorted list of triples. Once those triples with x-plane outliers have been removed, we repeat the process for the y-plane and then z-plane values. Figure 12 is a small example of this outlier-removal process. (Normally our data set is around 10,000 samples; however we are using a much smaller set in this example.)

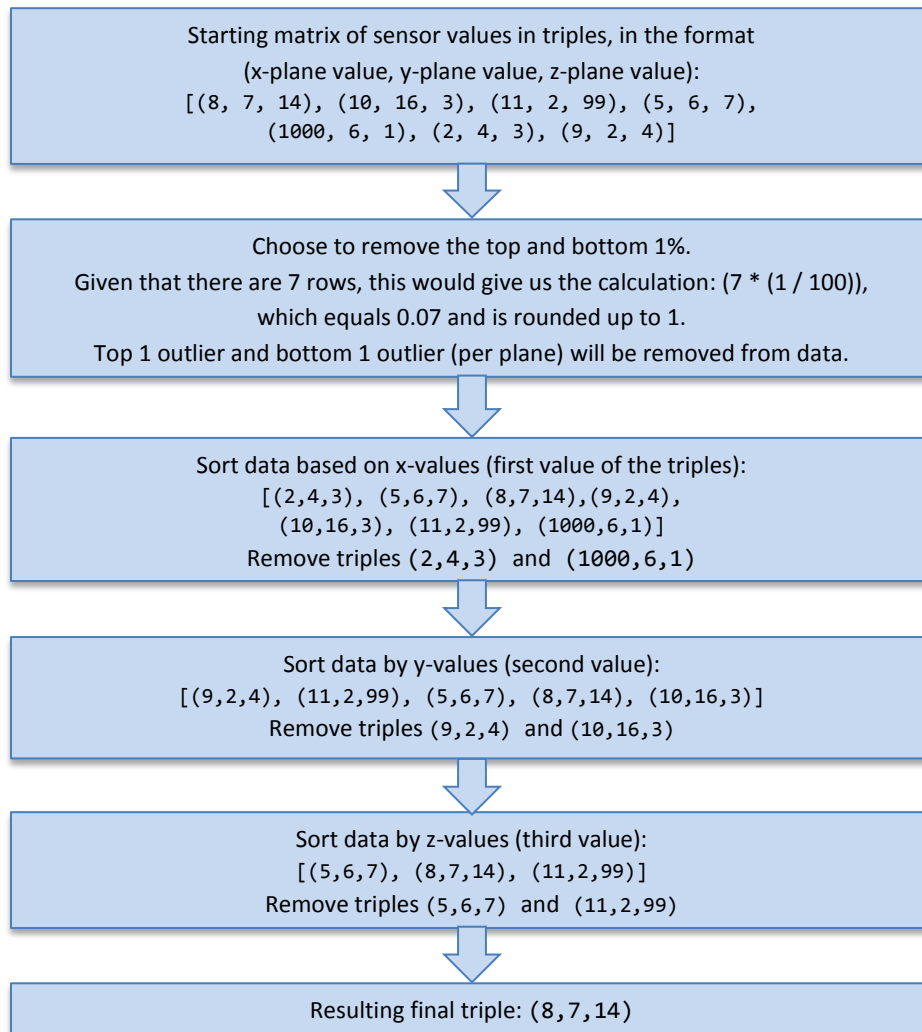


Figure 12: Flowchart explaining the outlier-removal process

### 3.3 Feature Normalization

No two individuals walk exactly the same, and each individual will likely not walk exactly the same way at the same intoxication level every time data is recorded. To minimize such inter-person differences in gait patterns, users' sway area and sway volume features were normalized per person. This task involves first calculating the average sway area by totaling and averaging the sway area data for every recording when a participant was sober (zero drinks). Once this baseline was created, each sway area calculated for a participant (at any BAC) was divided by the average sway area when a participant was sober to create a normalized sway area ratio. The formulas for average sway area and normalized sway area can be seen below.

$$\text{Average sway area} = \frac{\Sigma (\text{sober sway area calculations})}{\# \text{ sober readings}} \quad (\text{Equation 2})$$

$$\text{Normalized sway area} = \frac{(\text{current samples sway area})}{\text{average sway area}} \quad (\text{Equation 3})$$

### 3.4 User-Specific Classification Model Using Ensembling

A general machine learning model trained from data gathered from all users has been used as the default in the improved AlcoGait Android smartphone application. Per-user models have been found to be useful and this application will also allow the user to create a personalized model for himself or herself, rather than using the general model.

Ensembling allows the combination of the starter data with the individual user's data. Ensembling methods are "learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions" [11]. To make an ensemble of classifiers more accurate than one of its individual classifiers, the classifiers need to be both diverse and accurate. Two classifiers are considered diverse "if they make different errors on new data points" [11]. "An accurate classifier," Thomas G. Dietterich of Oregon State University writes, "is one that has an error rate of better than random guessing on new x values" [11]. Some options for constructing the classifiers are Bayesian Voting and manipulating the training examples with either a concept called bagging or using cross-validation to create a cross validated committee.

## 4 AlcoGait System Design and Implementation

Our AlcoGait system is comprised of three main parts: A desktop Matlab application that we designed for initial gait data collection, a server used to house the Matlab application to extract features from users' data in real-time, and an Android application that will use this server and our classification model to make real-time predictions of users' BAC. Figure 13 below depicts our development process of AlcoGait: Phase 1 included gather initial sensor raw data from study participants and extracting features from that data. We then used desktop Weka to select a classification algorithm and generate a model, and Phase 2, which involved the development of the AlcoGait server and the final Android application that would classify gait in real-time.

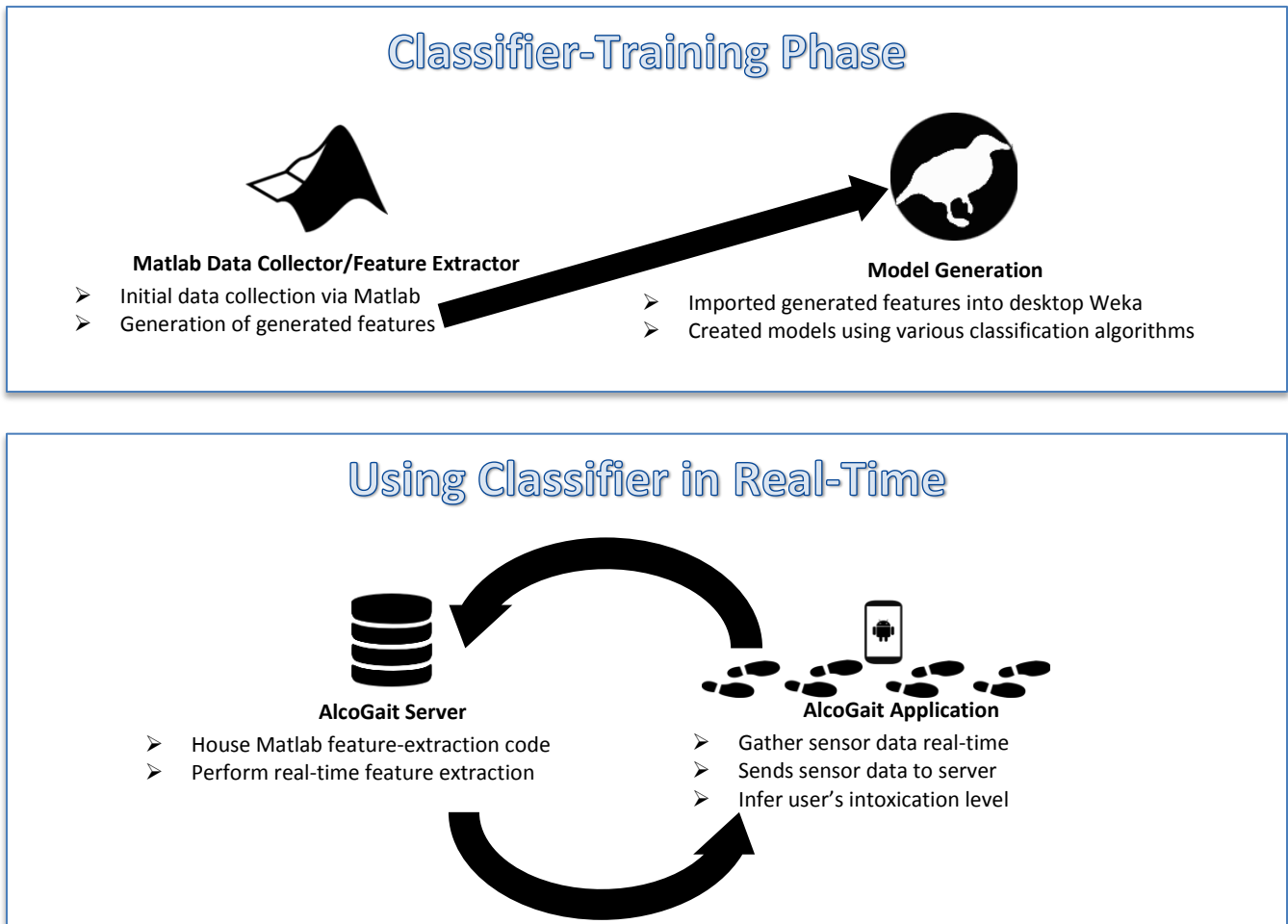
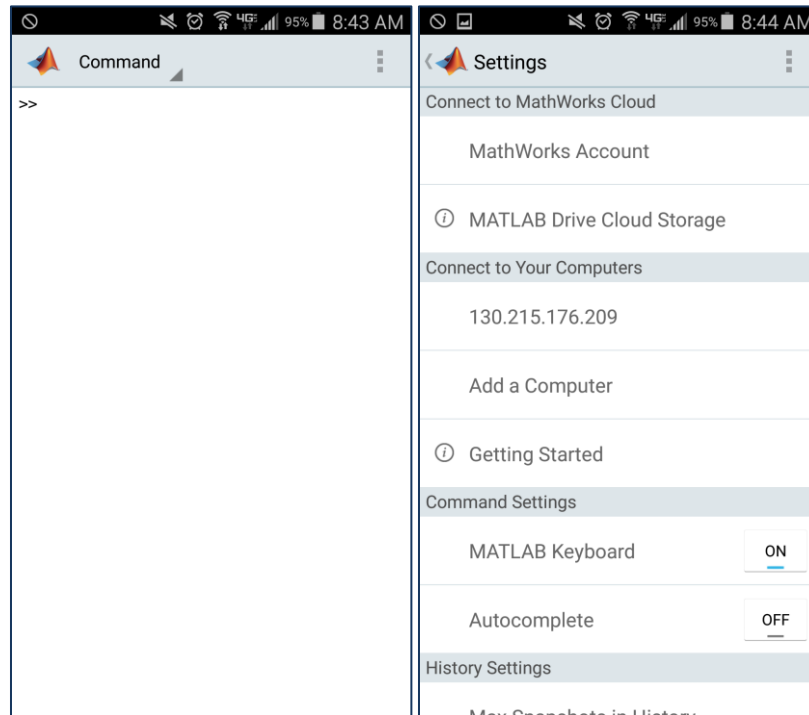


Figure 13: Diagrams showing AlcoGait project progression

### 4.1 “Data Collector/Feature Extractor:” A Desktop Matlab Application for Data Collection and Normalization

The “Data Collector/Feature Extractor” is a desktop Matlab program that was created to initially gather raw gyroscope and accelerometer gait data from participants by working in combination with Matlab Mobile for

Android. Matlab Mobile, installed on our Android device used for gathering data, can either run basic Matlab commands within the command line of the app, or it can communicate with a desktop Matlab application via WiFi by providing Matlab Mobile with the IP address of the computer that desktop Matlab is running on. In our research studies, we used Matlab Mobile to listen to the accelerometer and gyroscope sensors and send data back to our desktop Matlab application. Shown in Figure 1 below are 2 screenshots of Matlab Mobile.



**Figure 14: Left image shows the Matlab Mobile terminal; right image shows Matlab Mobile settings**

Once Matlab Mobile sent raw gyroscope and accelerometer sensor data to our Data Collector/Feature Extractor, features were then generated from this raw sensor data. The resulting generated feature data was recorded in a comma-separated value (CSV) file. This data then was used to train our classification model with Weka, and that model was finally exported from Weka and into our AlcoGait Android Application.

```
Command Window

Academic License

>> runAlcoGait
Are you:
  1) Collecting new data (type the number 1 below),
  2) Analyzing previously collected data from Funf or Matlab (type the number 2 below),
  3) Normalizing the data for a participant (type the number 3 below),
  4) Combining multiple CSV files for Weka classification (type the number 4 below)?
  5) Running a correlation and p-value analysis (type the number 5 below)?
fx |
```

Figure 15: Screenshot of the features offered by our Data Collector/Feature Extractor, our desktop Matlab application

#### 4.1.1 Initial Sensor Data Collection Used to Extract Features From

Initial “intoxicated user” and sober user data was collected via a desktop Matlab application initially developed by researchers Arnold and LaRose [15] and continued to be developed by us. We will refer to this throughout the paper as the “Matlab Data Collector/Feature Extractor.” The application initially creates a user profile for each participant, and each profile contains the participant’s name, height, weight, age, gender, and a randomly-generated ID number.

```
Command Window

Does this person need an ID #? Type Y to create one, or if the user already has an ID #, type that now.
Y
-----
Hello! Thank you for participating in this experiment. For each question, please type your answer below and then press the Enter key.

What is your first and last name? (Again, please type your answer and then press the Enter key.)
Christina Aiello
-----
What is your weight in lbs?
fx |
```

Figure 16: Screenshot showing some of the information that new study participants must enter

The application lets researchers enter the participant’s ID number, BAC, and a number of five-second trials to complete. The program receives raw gyroscope and accelerometer data from the Android Matlab application via Wi-Fi connection, and due to Wi-Fi’s unreliability, data is saved in five-second trials rather than waiting a full minute to save because if the Wi-Fi connection is lost at any point during a set of trials, any generated feature data that was previously saved has not been lost. The program was designed to attempt to restart itself automatically due to WiFi connectivity issues, and after ten restart attempts, the application displays a message to the researcher that it unsuccessfully tried to restart ten times.

Once the data-recording has been started, a voice says the word “begin” to the participant, alerting him or her to start walking. At the start of every five-second trial, a voice then says “Now recording” to let the researcher know that data is successfully being received and recorded. When five seconds of data has finished recording, a voice says, “Finished recording. Now analyzing and saving,” to indicate to the researcher that data is currently being saved. It is recommended to have the study participant continue to walk while the Matlab application generates features and saves data, rather than starting and stopping walking, to avoid making the participant repeatedly start and stop walking to keep the user walking more naturally.

#### 4.1.2 Feature Generation

Accelerometer features are generated with the Matlab Data Collector/Feature Extractor via the same methods created previously by researchers Arnold and LaRose [15], which are described above in section 3.1.3, ‘Detecting Intoxication via Gyroscope and Accelerometer Data.’ Gyroscope features are generated by collecting raw data from the gyroscope, removing outliers from the data, and then calculating sway area or sway volume. “XZ Sway Area” is created by plotting readings from X (pitch) and Z (yaw) axes and then plotting an ellipse that encompasses all of the plotted points. “YZ Sway Area” is calculated via readings from Z (yaw) and Y (roll) axes, and “XY Sway Area” is calculated via readings from X (pitch) and Y (roll) axes. “Sway Volume” is calculated using the readings from all three axes (pitch, roll, and yaw). The data for accelerometer and gyroscope features, in addition to the participant’s BAC at the time the data was generated, are saved to a CSV file on the researcher’s computer. Each participant has his or her own CSV file with one or multiple sets of generated feature data within them. Any values shown as “NaN” were unable to be computed by our Matlab functions and are just treated by Weka as missing data. The Matlab application then allows multiple CSV files to be combined into one file that can then be used with the Weka (Machine Learning) system. Sample data of all generated features can be seen in Table 3 below, where each row is a set of data we gathered and generated features from.

**Sample Generated Feature Data**

Steps	Cadence	Skew	Kurt	Gait Velocity	Residual Step Length	Ratio	Residual Step Time	Bandpower	SNR	THD	XZ Sway Area	YZ Sway Area	XY Sway Area	Sway Volume
7	0.41269	4.9785	34.971	0.058955	-9.2308	NaN	-14.0732	78977.845	-4.5713	-5.6597	21.613	8.0487	86.3641	4.73E+10
11	0.37983	1.3002	5.2246	0.03453	-3.6923	0.7664	-24.7818	4238.4543	-5.866	-9.5262	7.248	7.3589	12.7573	1250.322
19	0.45045	1.46	5.2562	0.023708	14.7692	0.246	-33.6214	5133.3959	-6.8341	-3.2764	7.2357	7.124	12.4601	1244.121

Table 3: Sample generated feature data from one of our participants, from walking done when participant was sober



### 4.1.3 Data Normalization

Our normalization process, which normalized a participant’s data (in comparison to his/her sober walk) before it became a part of our final model, was done due to individuals having differing walking patterns. When the set of trials has completed and sober data has been recorded (possibly in addition to other data, however sober data is required), the Data Collector/Feature Extractor normalizes the generated feature data, which can then be classified in Weka. This normalization involves first calculating the average sway area by averaging the sway area data for every recording when a participant was sober (zero drinks). Once this baseline has been created, each sway area calculated for a participant (at any BAC) will be divided by the average sway area when a participant is sober to create a normalized sway area ratio. Sample raw and normalized sway area data can be seen in Table 4 below. The middle column, “Raw Data,” shows the raw value we received for the given feature. The third column, “Normalized Data,” shows how the resulting value appears after going through the normalization process.

Feature	Raw Data	Normalized Data
XZ Sway Area	6.9945	0.903666
XZ Sway Area	6.9074	0.892413
XY Sway Area	7.6818	1.035159
XY Sway Area	7.4179	0.999597
YZ Sway Area	12.3547	0.486809
YZ Sway Area	11.688	0.460539

Table 4: Sample of raw generated feature data versus normalized data

## 4.2 Model Creation via Desktop Weka

A final gait-data model created with our generated feature data was used to then make real-time predictions of users’ BAC within our final Android application. Once data had been collected, features were generated from the data, and the data was normalized, we imported the data into the desktop Weka application in CSV form. Weka is a machine learning program containing various algorithms that can be trained with data provided by the user. Once our generated feature data was imported into Weka, we selected various combinations of machine learning algorithms such as Bayes Net, Random Forest, and Bagging, in combination with different training and testing methods, such as percent split and cross-validation. Training and testing methods are how data is given to the algorithm to train the model (teaching it about the data) and testing the model (seeing how accurate the

model is at making estimations about new data it hasn't seen before). A detailed explanation of this process can be seen in Appendix C, and a screenshot of the Weka application can be seen below in Figure 17.

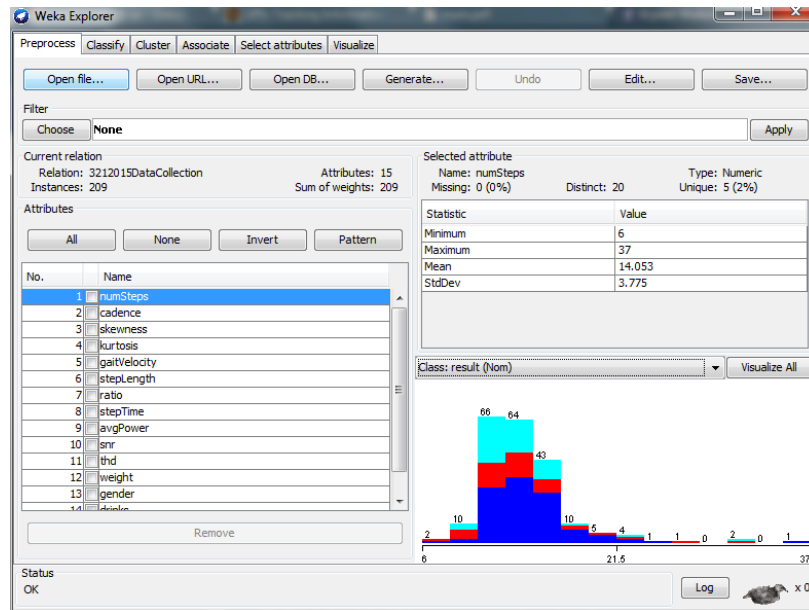


Figure 17: Screenshot of the Weka desktop application, with our data set loaded

### 4.3 AlcoGait Server

Our AlcoGait server currently hosts an executable version of part of our Matlab data collection code, which can be used to generate gait features real-time via a RESTful API call from our Android Application. The server is not making any modifications to our original gait data set; it is only generating gait features based on a user's smartphone's raw sensor data to be used immediately for classification within the Android Application.

### 4.4 Android Application

The third component of this the AlcoGait system is the final Android Application, which will classify a user's gait in real-time. Below we describe the initial setup process involved with the application and how our real-time classification algorithm is configured.

#### 4.4.1 Initial Application Setup

When users first install the application, they must specify their height, weight, gender, name, birthdate, and typical drinking times (with defaults set to 6pm to 4am). The height, weight, and gender information is used for classification. The typical drinking times are used by the application to determine when to analyze whether the user is walking or not (which then triggers the data collection process if the user is walking, a process we explain in detail below).

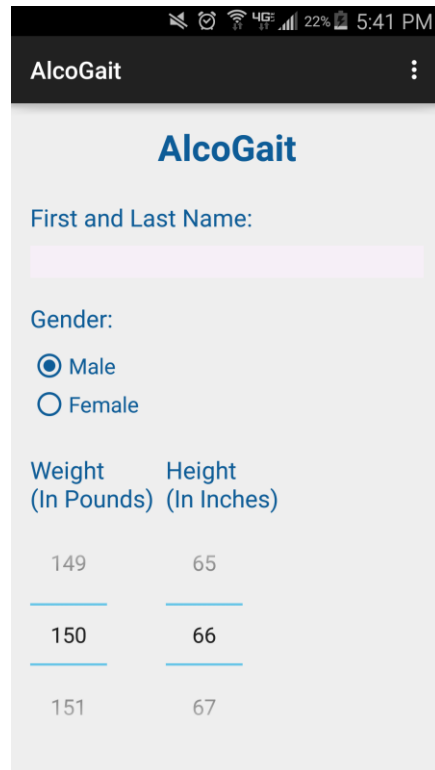


Figure 18: Screenshot of initial setup screen

Next, we ask new users to use a feature within our application to record one minute of sober walking data, data that we extract features from to use as a baseline for normalization. The user is given instructions on the screen describing how the process works: The user presses the “START” button and then has 5 seconds to put the phone in his or her front or back pants pocket. The phone will then sound an alarm signaling the user to start walking. When a minute has passed the user hears this alarm again, indicating that he or she can stop walking. The user then must click the “SAVE” button to save this data. If he or she wants to re-record this data, the user can do so by pressing “START” again. Once we have acquired this baseline data, the classification process can begin. Figure 19 below is a screenshot of our baseline-data-collection screen.

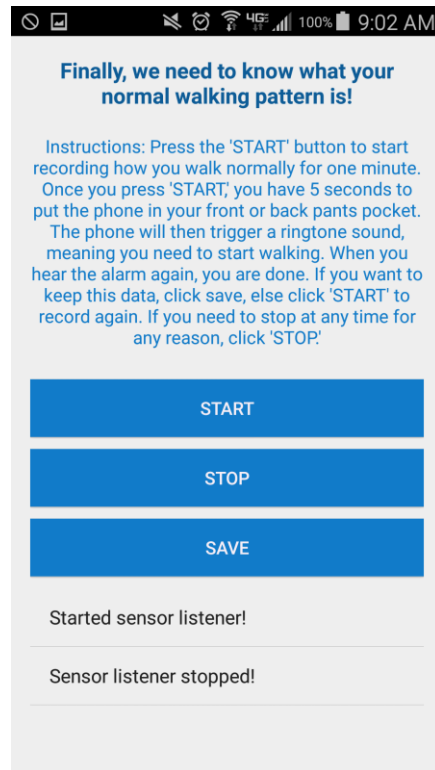


Figure 19: Screenshot of screen where sober walking data is gathered

#### 4.4.2 Real-Time In-App Classification

If the current time is between the user's chosen starting and ending "typical" drinking times, the application listens to the user's actions to detect if the user is walking via the Google Activity Detection API, which we explain more below. If the user is walking, the application starts listening to and recording raw accelerometer and gyroscope sensor data. The application either continues collecting sensor data if one minute has passed, or the application stops listening to the sensors if the user has stopped walking. Once a minute of raw sensor data has been collected successfully, the raw sensor data is transformed into JSON and is sent via RESTful API call to our AlcoGait server. The server contains a version of our Matlab Data Collector/Feature Extractor in an executable file that extracts features from our raw sensor data and then sends the generated feature data back to the Android application, a process that takes approximately 4 to 5 seconds. The application then normalizes this data, and lastly the application leverages the Java library for Weka to classify our generated feature data once it has been normalized. The screenshot in Figure 20 below shows how the result of this classification is displayed to the user: An estimated BAC range, in addition to a timestamp of when this estimation was made, is shown on the main screen of the application.

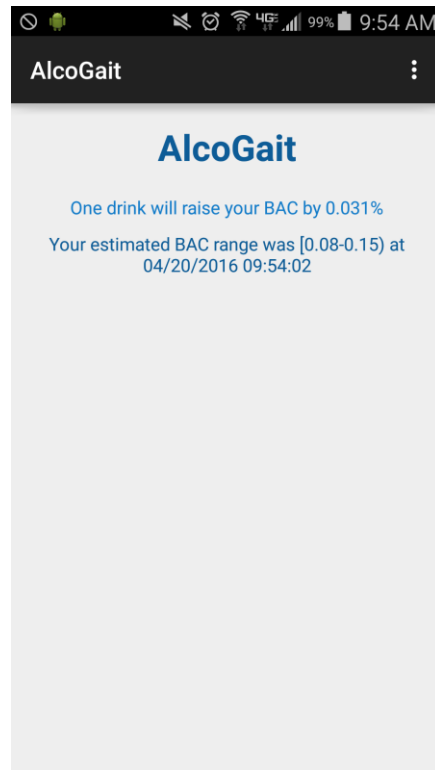


Figure 20: Main screen displaying the results of our feature extraction and classification.

#### 4.4.2.1 Detecting Walking Action

This application requires having the ability to detect when the user is walking, to accurately classify one's walking patterns. First the application detects the user's motion. Once walking has been detected, sensor data gathering is started (gathering raw accelerometer and gyroscope data). When walking has ceased, the data is either saved (if one minute has passed) or deleted (if not enough time has passed) and the process restarts. If data is successfully collected for one minute and saved, that data is then sent via a REST request to our server to have features generated from it. Once those generated feature values are sent back from the server to the phone, the Weka model is used to classify the data and make a prediction about the user's estimated BAC range. This process is detailed in Figure 21 below.

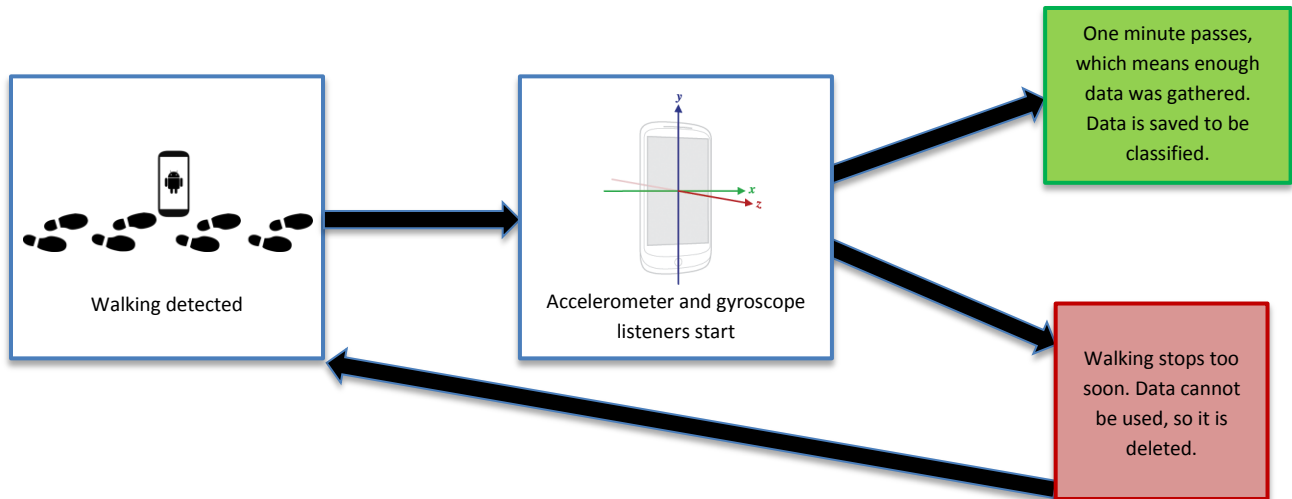


Figure 21: Flow diagram explaining the process of detecting walking and gathering sensor data

Motion detection was specifically done via the *DetectedActivity* class, which is a part of the Google Activity Detection API. This class can detect if a user is in a vehicle, on a bicycle, on foot, running, still, tilting, or walking. This *DetectedActivity* class has a confidence value from 0-100, where a larger value indicates a higher confidence level, and a value of  $\leq 50$  means another activity may be just as likely. A broadcast receiver was created in our AlcoGait project to listen for changes in the user's activity and assess the confidence level of the activity reading.

In investigating power consumption of this API on a Samsung Galaxy S4 Android device via a power-analysis application called *Trepn Power Profiler* [35], checking the user's activity every second for 30 minutes (total test time was 30 minutes) brought the battery percentage from 26% to 22% and used  $\sim 28$  watts of battery power. Checking activity every 5 seconds for 30 minutes caused battery percentage to decrease from 22% to 20% using  $\sim 30$  watts. We concluded that the consumption would be improved if classification is only attempted once every 15 minutes, meaning that activity detection would also only be run every 15 minutes. The results of this analysis can be seen in Figure 22 below.

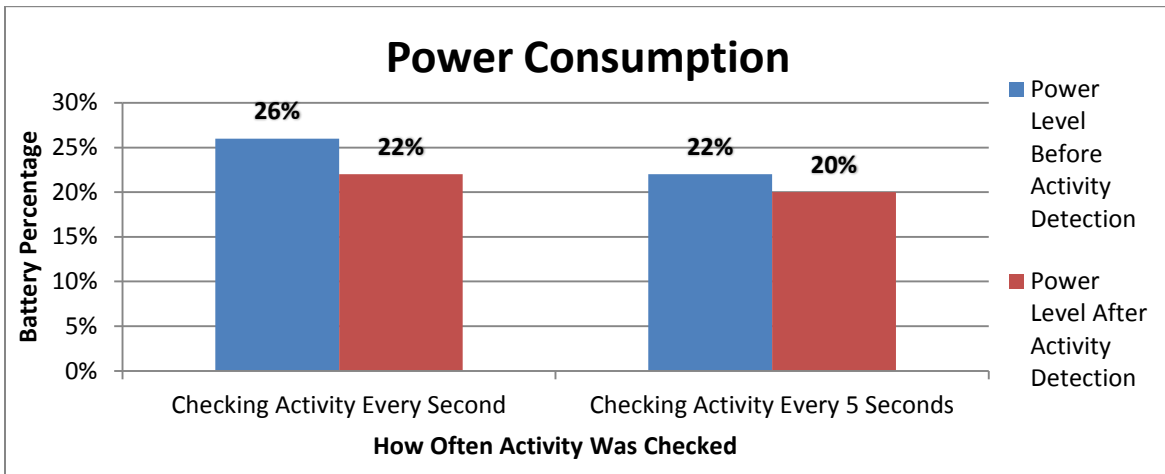


Figure 22: How activity detection affected battery life

#### 4.4.2.2 Integrating Matlab Code into Android Application

We explored various options for integrating our Matlab code with our smartphone application, eventually making use of a server to house an executable of our Matlab code. This process is described below.

Numerous options were explored regarding integrating our Matlab desktop application into the final product, the Android Smartphone Application. We first explored creating a library (JAR file) using the Matlab Compiler within desktop Matlab. This was unfortunately unsuccessful due to the inability to put the Matlab Compiler Runtime, which is needed to run code in any Matlab-compiled libraries, on an Android Smartphone. We then explored the concept of converting the Matlab code to C code using Matlab's C Compiler, which would convert our Matlab code to code in the C programming language, however our Matlab code was not able to be fully converted to C code, causing a loss of most of our generated features. Simulink, a Matlab product that gives the ability to run Matlab libraries on select Android Devices, was explored next. Unfortunately we were unable to access Simulink due to a lack of access to Matlab 2016, however this is an option we may pursue in the future.

We finally moved on to the concept of creating a standalone Matlab executable from our Data Collector/Feature Generator Matlab program and putting that executable on a server. The Android Application would then make a REST request (which stands for "Representational State Transfer") to the server via Internet connection, sending over raw sensor data in JSON format. The server uses a service written in the PHP language to store this sensor data in a comma-separated values (CSV) file within the server. Next the server runs the Matlab executable, which accesses the raw data in the CSV file. The Matlab code generates features and stores them in JSON format in a second CSV file on the server, and the JSON from that file is then sent back to the Android device. We initially explored the idea of using an Ubuntu server for these transactions; however the Matlab executable would not properly run on an Ubuntu server due to the limited options that Matlab code could be exported as.

We then explored the idea of using a Windows server created via Internet Information Services (IIS), a component of Windows 7. This Windows server was successful in running our Matlab executable via a REST call, and this can be seen in Figure 13 earlier in this thesis.



## 5 Analysis and Results

### 5.1 Initial Data Collection for Training the BAC Detection Model

Initial data collected via Matlab was classified using the desktop version of Weka. We gathered gait data from 34 participants, 20 female and 14 male. Participants' heights ranged from 150cm to 200cm (with a mean of 172cm with a standard deviation of 10.22cm), weights ranged from 100lbs to 250lbs (with a mean of 155lbs with a standard deviation of 31.96lbs), and ages ranged from 18 to 22 (with a mean of 20 years with a standard deviation of 1.32 years). A visual breakdown of participant demographics can be seen in the four graphs below, and Appendix B contains our complete data set of participant information.

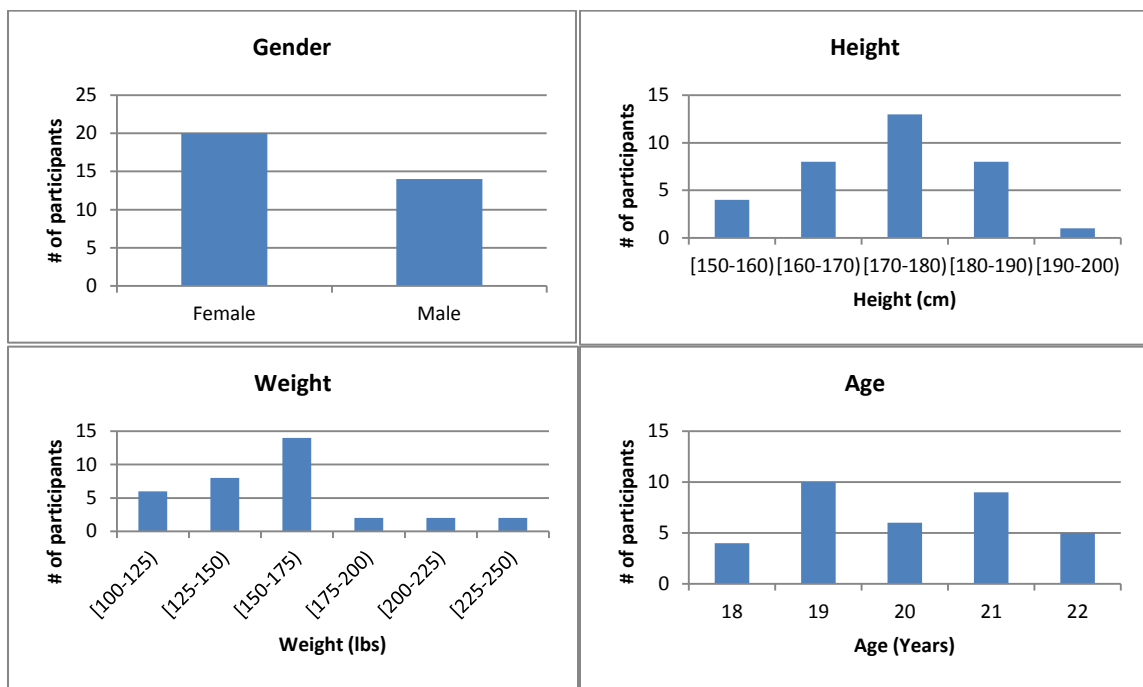


Figure 23: These four graphs show the breakdown in various attributes of our participants.

### 5.2 Data Classification

Classification algorithms that we investigated were J48, JRip, Bayes Net, Random Forest, Random Tree, and Bagging. We investigated classification accuracy of normalized vs. not normalized data, in addition to removing certain generated features (all gyroscope features, all accelerometer features, certain gyroscope features, and certain attributes describing the participant such as height, weight, and gender). Not including gyroscope features within one set of classification testing, in addition to not including accelerometer features, let us assess whether using features from solely one sensor would be superior to using both sensors. To determine the

successfulness of a classifier, we examined the percent accuracy, precision (proportion of instances that are actually of a class divided by the total instances classified as that class), recall (proportion of instances classified as a certain class divided by the actual total in that class), f-measure (a combined measure for precision and recall calculated as  $2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall})$ ), and ROC area, which is the area under the curve when plotting true positives versus false positives [15]. Figure 24 below shows the highlights of our classification results when classifying our data using the desktop version of Weka, and more detailed information regarding our classification results can be seen in section 5.2.3, “Comparing Accuracy of Various Classifiers.” The biggest takeaways from classification were:

- I. When classifying all generated features within desktop Weka from both sensors (with sway areas and volume normalized), ID, height, weight, and gender, the J48 classifier using percentage split, 99% train and 1% test had the highest accuracy of 89.45% (highest in comparison to our other classifiers used with this data set) and an ROC area of 0.916. In addition, this was our best classifier overall in comparison to classifiers trained on different combinations of the available features.
- II. When classifying all generated features within from both sensors (with sway areas and volume not normalized), ID, height, weight, and gender, the J48 classifier using percentage split, 99% train and 1% test had the highest accuracy of 88.89% (highest in comparison to our other classifiers used with this data set).
- III. When classifying all generated features from just the accelerometer, Random Forest using cross-validation, 10 folds had the highest accuracy of 64.56% (highest in comparison to our other classifiers used with this data set) and an ROC area of 0.851.
- IV. When classifying all generated features from just the gyroscope, Random Forest using cross-validation, 10 folds had the highest accuracy of 75.79% (highest in comparison to our other classifiers used with this data set) and an ROC area of 0.919.

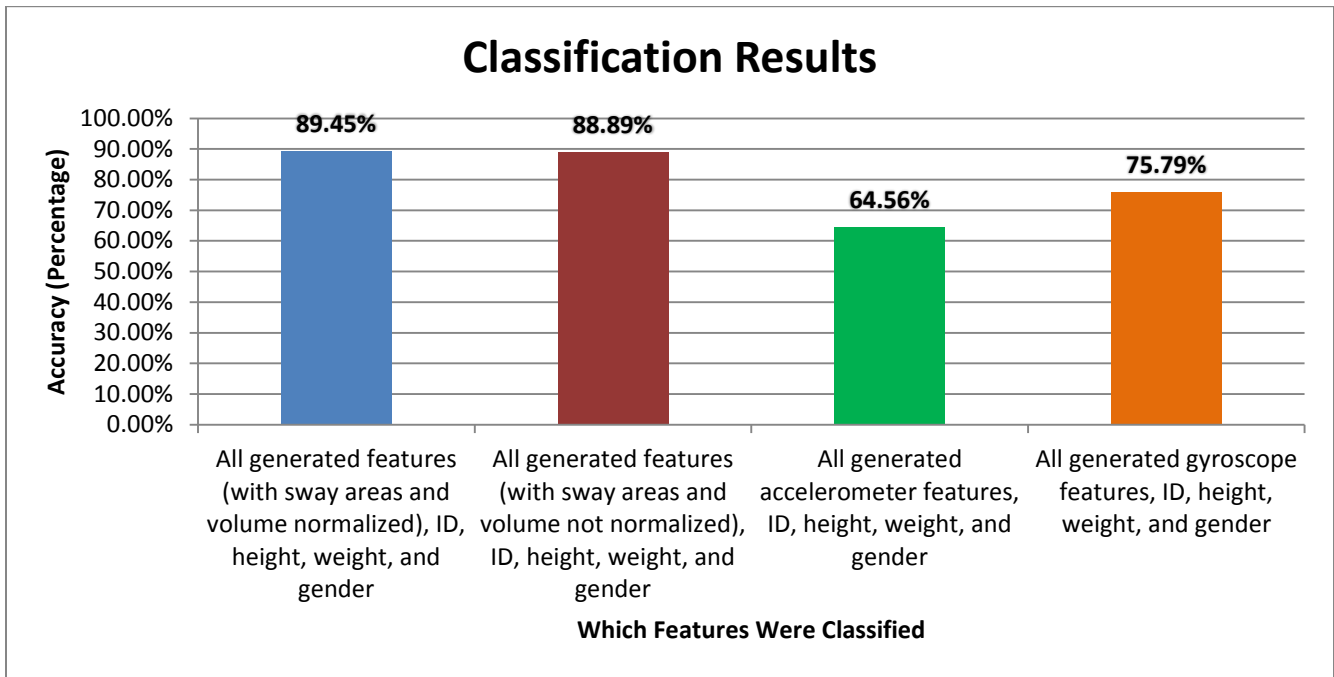


Figure 24: Graph showing the accuracies of the best-performing classifiers

### 5.2.1 Exploring Value of and Removal of Features

We explored the removal of features to see if removing various features increased or decreased accuracy. In some instances, classifiers trained on just the gyroscope data were more accurate than classifiers with gyroscope and accelerometer data. In other cases, classifiers trained with data from both sensors were superior. An overall trend was that the classifiers trained on just accelerometer data were (in most instances) less accurate than the classifiers with data from both sensors and the classifiers with just gyroscope data. Again, this data can be seen in Appendix A. Shown below in Table 5 are the p-values and correlation coefficients of each feature used for classification. Features with p-values less than or equal to 0.05, meaning they are statistically significant, have been highlighted in green.

P-Value	Correlation Coefficient	Feature
0.21537	-0.02881	Steps
0.21441	-0.02887	Cadence
4.26E-12	-0.1601	Skew
3.71E-10	-0.14496	Kurt
0.002902	-0.06918	Gait Velocity
0.2366	-0.02752	Residual Step Length
NaN	NaN	Ratio
8.22E-06	0.10344	Residual Step Time
2.38E-18	-0.20114	Bandpower
NaN	NaN	SNR
NaN	NaN	THD
8.49E-18	-0.19788	XZ Sway
6.71E-13	-0.16596	XY Sway
2.87E-24	-0.23314	YZ Sway
3.59E-08	-0.12763	Sway Volume
0.95949	0.001181	Weight
0.71924	-0.00836	Gender
0.69053	0.00926	Height
0.29829	0.024188	Participant ID

Table 5: P-Values and correlation coefficients for each classification feature

### 5.2.2 Including Additional Physical Attributes of Participants

We initially classified the generated features data along with the participant’s weight, following the procedure originally created by the previous researchers. Once this was completed, we classified generated features again and included both the participant’s gender and weight. We then classified the features in addition to the participant’s weight and height. Finally we classified the generated features data in addition to a combination of the participant’s gender, height, and weight, which was the most accurate. For example, when classifying the normalized data using Random Forest with percentage split (99% train, 1% test), including gender, height, and weight was 5.3% more accurate than solely using generated features and weight.

### 5.2.3 Comparing Accuracy of Various Classifiers

We explored various classifiers available within Weka, such as J48, Random Forest, Bayes Net, JRip, and Bagging. The results of our classification exploration can be seen in Table 6 below.

Classification Configuration			Results					
Attributes Classified	Classifier	Test Set	Accuracy When Normalized	Precision	Recall	F-Measure	ROC Area	Accuracy When Not Normalized
Accelerometer features, gyroscope features, ID, height, weight, gender	J48	Cross-validation, 10 folds	69.53%	0.695	0.695	0.695	0.817	69.26%
	J48	Percentage split, 66% train 33% test	63.28%	0.63	0.633	0.631	0.786	65.74%
	J48	Percentage split, 95% train 5% test	73.12%	0.735	0.731	0.731	0.835	72.22%
	<b>J48</b>	<b>Percentage split, 99% train 1% test</b>	<b>89.45%</b>	<b>0.912</b>	<b>0.895</b>	<b>0.895</b>	<b>0.916</b>	<b>88.89%</b>
	Random Forest	Percentage split, 66% train 33% test	72.66%	0.723	0.727	0.721	0.892	69.18%
	Random Forest	Percentage split, 95% train 5% test	81.72%	0.816	0.817	0.809	0.924	75.56%
	Random Forest	Percentage split, 99% train 1% test	73.68%	667	0.737	0.674	0.946	77.78%
	Random Forest	Cross-validation, 10 folds	73.74%	0.735	0.737	0.731	0.91	74.79%
	Random Tree	Cross-validation, 10 folds	67.69%	0.68	0.68	0.68	0.782	66.26%
	Random Tree	Percentage split, 66% train 33% test	66.77%	0.672	0.668	0.669	0.771	63.44%
	JRip	Cross-validation, 10 folds	50.29%	0.503	0.503	0.435	0.616	50.31%
	Bayes Net	Cross-validation, 10 folds	43.60%	0.405	0.436	0.41	0.696	44.34%
	Bayes Net	Percentage split, 66% train 33% test	45.47%	0.348	0.455	0.386	0.691	42.46%
	Bagging	Cross-validation, 10 folds	67.53%	0.673	0.675	0.674	0.792	70.94%
	Bagging	Percentage split, 66% train 33% test	60.25%	0.606	0.603	0.604	0.761	65.25%
All of the above except gyroscope features	Random Forest	Cross-validation, 10 folds	64.56%	0.637	0.646	0.637	0.851	
	Random Forest	Percentage split, 66% train 33% test	60.73%	0.597	0.607	0.597	0.815	
	J48	Cross-validation, 10 folds	63.59%	0.634	0.636	0.635	0.783	
	J48	Percentage split, 66% train 33% test	56.92%	0.569	0.569	0.568	0.74	
All of the above except accelerometer features	Random Forest	Cross-validation, 10 folds	75.79%	0.755	0.758	0.755	0.919	
	Random Forest	Percentage split, 66% train 33% test	75.20%	0.754	0.752	0.749	0.909	
	J48	Cross-validation, 10 folds	69.91%	0.697	0.699	0.698	0.838	
	J48	Percentage split, 66% train 33% test	70.75%	0.708	0.707	0.707	0.837	

Table 6: Results of training various classifiers with our dataset

### 5.2.3.1 Machine Learning Classifiers Tested

*J48*: an open source version of C4.5, an algorithm used to generate a decision tree. This algorithm, developed by Ross Quinlan, builds decision trees from a set of training data using the idea of information entropy. At each

node of tree, the algorithm picks the attribute that best splits its set of samples into subsets in one class or the other: The attribute with highest information gain is chosen to make decision [22].

*Random Forest*: an algorithm that uses the ensemble approach. "Ensemble" means to divide and conquer to improve performance. The algorithm creates decision trees using random subsets of attributes from the data, and it then finds a variable (and value at that variable) to make a binary split of the data [23].

*Bayes Net*: a probabilistic graphical model, which is a type of statistical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG) [24].

*JRip*: a propositional rule learner, described as, "Repeated Incremental Pruning to Produce Error Reduction (RIPPER)." This algorithm uses incremental reduced error pruning. It grows rule sets one at a time, building full rule set and then pruning it, and finally simplifying each rule as soon as it's built [24].

*Bagging*: generates a number of new training data sets, and each new training data set picks a sample of observations with replacement (bootstrap sample) from original data set [24].

### 5.2.3.2 Overall Classification Accuracy

The J48 and Random Forest classifiers were superior in classification compared to the other classifiers explored, which were Random Tree, Bagging, JRip, and Bayes Net. The results showed that J48 and Random Forest were 21% to 39% more accurate than the other classifiers we investigated.

### 5.2.3.3 Best Classifiers for the Various Bins

In addition to investigating which classifiers produced the overall highest accuracy, we examined which classifiers were the most successful at classifying data of certain bins. Bin "a," a BAC of [0.00-0.08), was best classified via Random Forest, using percent split (99% of the data was used in training the model and 1% of the data was used for testing the model). Bin "b," a BAC of [0.08-0.15), was best classified via J48 using percent split 99% train 1% test & Random Forest using percent split with 99% train 1% test. Bin "c," a BAC of [0.15-0.25), and bin "d," a BAC of [0.25+), both had the best success with J48 using percent split 99% train 1% test. The confusion matrices for these classifiers can be seen in Figure 25 below. The first row of letters (a, b, c, d) is what the sample was classified as (where 'a,' 'b,' 'c,' and 'd' correspond to our bins [0.00-0.08), [0.08-0.15), [0.15-0.25), [0.25+), respectively), and the letters in the final column (which again are a, b, c, and d) are the actual bins of the samples (again corresponding to our bins [0.00-0.08), [0.08-0.15), [0.15-0.25), [0.25+), respectively).

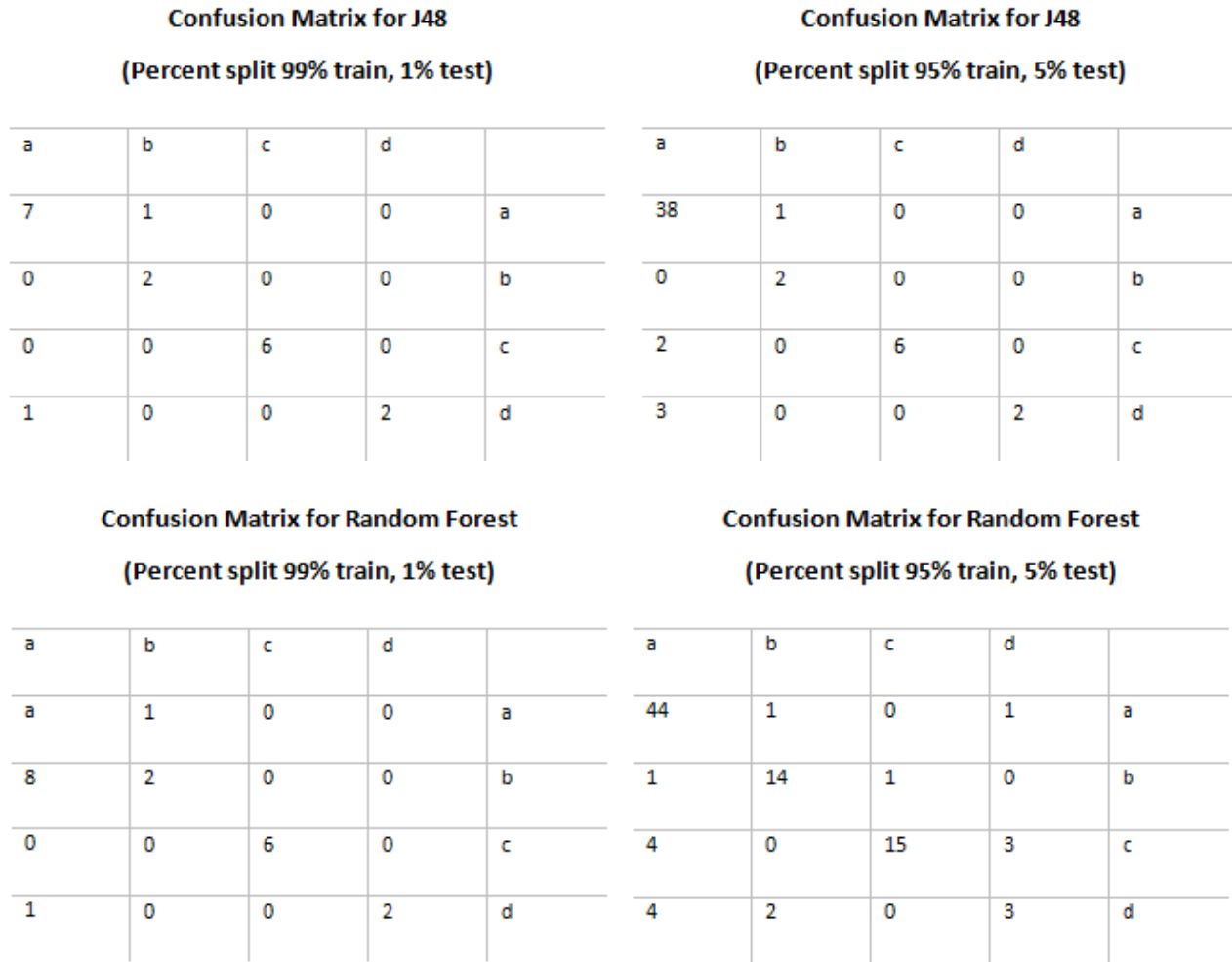


Figure 25: The four confusion matrices for our four best-performing classifiers

### 5.2.4 Analysis of Sway Area and Sway Volume

Our results showed that sway area and sway volume generally increased as a participant became more “intoxicated.” Shown in the box plots in Figure 26 below are sway areas for one participant, and these plots display the increasing sway area at each intoxication level. The first set of box plots shows the data after normalization, and the second set shows the data before the normalization process.

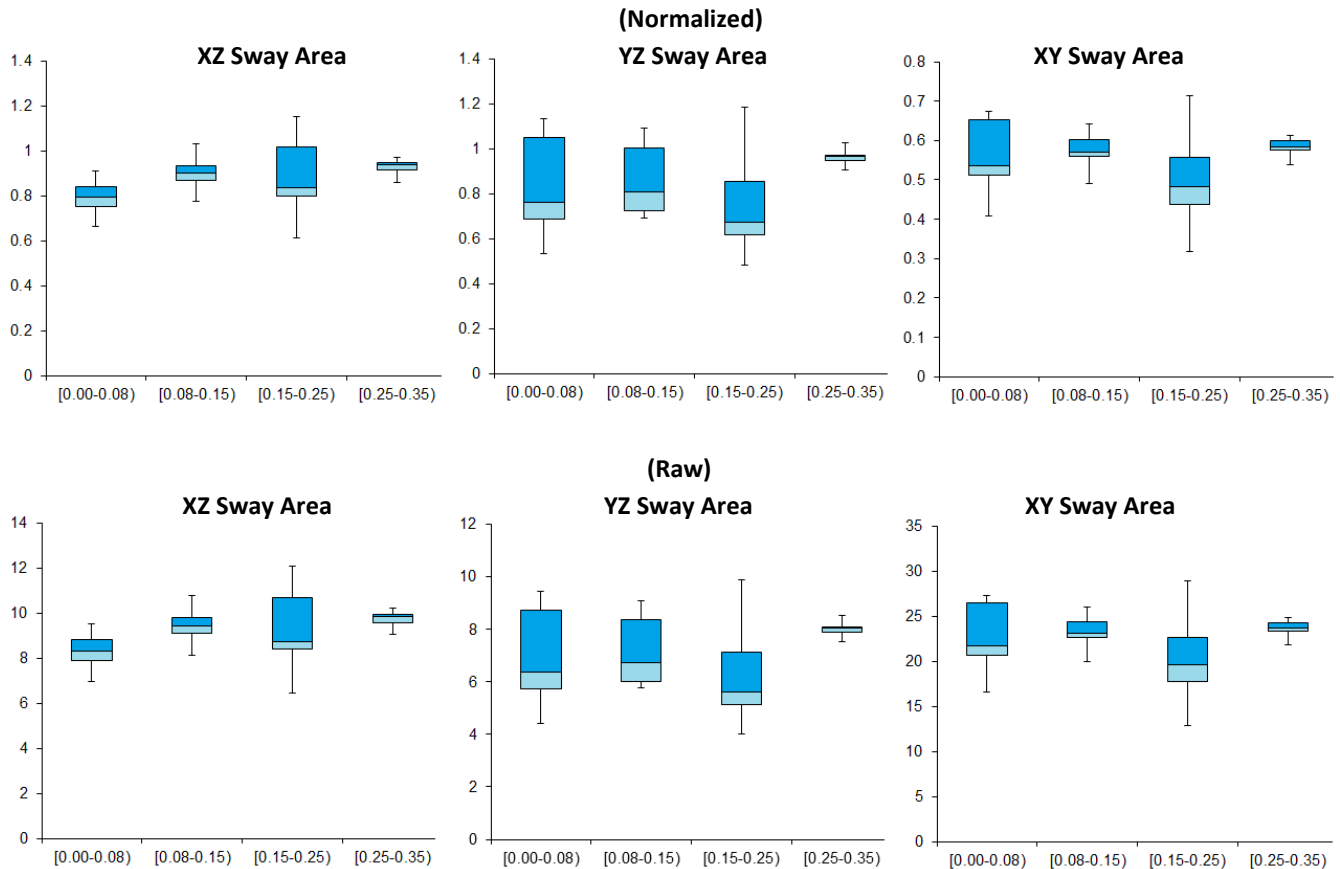


Figure 26: Boxplots showing the increase in some of our gyroscope sway areas as a participant became more "intoxicated"

Regarding classification, normalization of the three sway areas and sway volume had varying effects on classification accuracy. When using the J48 classifier, normalization made the classifier -4.1% to 6.16% more accurate, average of 0.75%. When using the Random Forest classifier, normalization made classifier -5% to 3.33% more accurate, average of -0.35%.

### 5.3 Exploring the Concept of Maintaining One's Walking Speed

We wondered whether if participants were walking too slowly or holding onto a wall when walking, the generated features from accelerometer data may be lost (or may become less accurate). We decided to explore the idea of hour-long studies, studies during which participants were asked to walk for one minute with no impairment and then mark where they stopped after walking for one minute. That walking pattern then had to be repeated with each pair of impairment goggles, meaning the participant had to walk at the same speed that he or she walked at before. We conducted these studies with four participants.

Overall the J48 classifier was the most accurate in our testing. Once data was collected and gait features were generated, we used Desktop Weka to create classifiers for this data. When using the J48 classifier and cross-



validation 10 folds, the model was 63% accurate. Using percentage split (66% of data being used for training and 33% used for testing), rather than cross-validation, brought the accuracy down to 56%. Using Random Forest with cross-validation produced a classifier with 67% accuracy, and using percentage split rather than cross-validation, produced a classifier with an accuracy of 60%.

#### 5.4 Investigating Personalization

We investigated the idea of personalization, which involved creating a model based solely on a singular user’s gait data that we had collected. Unfortunately our results were inconclusive: personalization improved some results (compared to our 89.45% accuracy with our overall model) and worsened others. Shown in Table 7 below are the results of our personalization exploration, with rows highlighted in green if personalization brought the accuracy above our general model’s accuracy. We also have the additional concern regarding personalization that users would have to self-report their BAC when constructing a personalized model, which requires the user to have repeated access to a breathalyzer.

Participant ID	Classifier	Configurations	Accuracy	ROC Area
219171	J48	Cross-validation 10 folds	76.70%	0.843
219171	J48	Percent Split 66% train, 33% test	75%	0.768
219171	J48	Percent Split 99% train, 1% test	0%	0.000
219171	Random Forest	Cross-validation 10 folds	80%	0.967
219171	Random Forest	Percent Split 66% train, 33% test	75%	0.915
1506627	J48	Cross-validation 10 folds	93.30%	0.976
1506627	J48	Percent Split 66% train, 33% test	75%	0.773
1506627	J48	Percent Split 99% train, 1% test	100%	0.000
1506627	Random Forest	Cross-validation 10 folds	97%	0.998
1506627	Random Forest	Percent Split 66% train, 33% test	75%	0.964
1520109	J48	Cross-validation 10 folds	76%	0.86
1520109	J48	Percent Split 66% train, 33% test	80%	0.888
1520109	J48	Percent Split 99% train, 1% test	100%	0.000
1520109	Random Forest	Cross-validation 10 folds	90%	0.942
1520109	Random Forest	Percent Split 66% train, 33% test	85%	0.972

Table 7: Results of exploring personalization

## 6 Conclusions and Future Work

This thesis researched and developed the AlcoGait smartphone application that was 89.45% successful in classifying a user's intoxication level via the J48 algorithm based on features generated from raw gyroscope and accelerometer sensor data from an Android smartphone. The AlcoGait application successfully leverages our trained classification model in real-time by first making a RESTful API call to our AlcoGait server running our Matlab executable, then using the returned results to classify our user's gait against our classification model, all while only requiring setup information from the user (and no other interaction).

Future work could include gathering data from actual intoxicated participants rather than simulating intoxication via Drunk Busters Goggles. In addition, we would like to find a method of integrating our Matlab feature-extraction code with our Android application on the device rather than by making RESTful API calls to our server. The analysis of other sensor data, such as the GPS, could indicate to us whether a user is in a location (such as a bar or restaurant) that sells alcoholic beverages. Finally, this application could be integrated into a health care system. Users struggling with alcohol addiction could use this application to track their intoxication levels, and that data could then be sent to a physician for analysis and for making recommendations.

## References

1. Muscolino, J., & Muscolino, J. (2013). Workbook for Know the body: Muscle, bone, and palpation essentials. St. Louis, Mo.: Elsevier Mosby.
2. Event Handling Guide for iOS. (n.d.). Retrieved Dec 1, 2015, from [https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/motion\\_event\\_basics/motion\\_event\\_basics.html](https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/motion_event_basics/motion_event_basics.html)
3. Nieschalk, M., Ortmann, C., West, A., Schmä, F., Stoll, W., & Fechner, G. (1999). Effects of alcohol on body-sway patterns in human subjects. *Int'l Journal of Legal Medicine*, 253-260.
4. Sensors Overview. (n.d.). Retrieved Dec 1, 2015, from [http://developer.android.com/guide/topics/sensors/sensors\\_overview.html](http://developer.android.com/guide/topics/sensors/sensors_overview.html)
5. Intoxichck® Phone App - Buzzed Driving App | Impaired Driving App | Impairment Assessment. (n.d.). Retrieved Dec 1, 2015, from <http://fatalvision.com/intoxichck.html>
6. Lifeloc Technologies - Impairment Goggles. (n.d.). Retrieved Dec 1, 2015, from <https://lifeloc.com/c-60-impairment-goggles.aspx>
7. Caruana, R. (2006). Performance Measures for Machine Learning, CS 678 Spring 2006. Retrieved Dec 1, 2015, from [http://cs.cornell.edu/courses/cs678/2006sp/performance\\_measures.4up.pdf](http://cs.cornell.edu/courses/cs678/2006sp/performance_measures.4up.pdf)
8. Coalition against Drug Abuse, "History of Drug Abuse," 2014. [Online]. Available: <http://drugabuse.com/library/history-of-drug-abuse/>. [Accessed 5 November 2015].
9. H.-L. Kao, B.-J. Ho, A. C. Lin and H.-H. Chu, "Phone-based Gait Analysis to Detect Alcohol Usage," ACM, Pittsburgh, USA, 2012.
10. K.-C. Wang, Y.-H. Hsieh, C.-H. Yen, C.-W. You, M.-C. Huang, C.-H. Lee, S.-Y. Lau, H.-L. Kao, H.-H. Chu and M.-S. Chen, "SoberDiary: A Phone-based Support System for Assisting Recovery from Alcohol Dependence," in Proc ACM Ubicomp 2013, Seattle, WA, 2013.
11. Dietterich, Thomas. Ensemble Methods in Machine Learning, (1-5). Retrieved Dec 4, 2015, from <http://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf>
12. McKay, J. R., Franklin, T. R., Patapis, N., and Lynch, K. G. Conceptual, Methodological, and Analytical Issues in the Study of Relapse. *Clin. Psychol. Rev.* 26, 2 (Mar 2006), 109–127.
13. Verhoeff, L., Horlings, C., Janssen, L., Bridenbaugh, S., & Allum, J. (n.d.). Effects of biofeedback on trunk sway during dual tasking in the healthy young and elderly. *Gait & Posture*, 76-81.
14. Davis, J., Carpenter, M., Tschanz, R., Meyes, S., Debrunner, D., Burger, J., & Allum, J. (2010). Trunk sway reductions in young and older adults using multi-modal biofeedback. *Gait & Posture*, 465-472.
15. Arnold, Zach and LaRose, Danielle. "Smartphone Gait Inference." WPI Major Qualifying Project, April 2015. <https://wpi.edu/Pubs/E-project/Available/E-project-043015-100131/unrestricted/SmartphoneGaitInference.pdf>
16. He, Qian and Agu, E. "Investigating the Use of the Smartphone Gyroscope to Improve Activity Recognition and to Detect Phone Position." Unpublished manuscript
17. Nocera, M., & Ward, R. M. (2013). The relationship between Thursday drinking, BAL, and reported number of drinks consumed. *Proceedings of the National Conference on Undergraduate Research*, 26, 741-745 [https://ncurdb.cur.org/ncur2014/archive/Display\\_NCUR.aspx?id=72226](https://ncurdb.cur.org/ncur2014/archive/Display_NCUR.aspx?id=72226)
18. Chambers, M., Liu, M., Moore, C.: Drunk driving by the numbers. United States Department of Education.

19. Fact Sheets - Alcohol Use and Your Health. (2015, December 17). Retrieved January 8, 2016, from <http://www.cdc.gov/alcohol/fact-sheets/alcohol-use.htm>
20. Impaired Driving: Get the Facts. (2015, November 24). Retrieved January 8, 2016, from [http://www.cdc.gov/motorvehiclesafety/impaired\\_driving/impaired-driv\\_factsheet.html](http://www.cdc.gov/motorvehiclesafety/impaired_driving/impaired-driv_factsheet.html)
21. Tjondronegoro, D.; Drennan, J.; Kavanagh, D.J.; Zhao, E.J.; White, A.M.; Previte, J.; Connor, J.P.; Fry, M.-L., "Designing a Mobile Social Tool that Moderates Drinking," in *Pervasive Computing, IEEE* , vol.14, no.3, pp.62-69, July-Sept. 2015 doi: 10.1109/MPRV.2015.62
22. University of Minnesota. (n.d.). Classification Methods. Retrieved February 10, 2016, from <http://www.d.umn.edu/~padhy005/Chapter5.html>
23. Wiener, M., & Liaw, A. (2002, December). Classification and Regression by randomForest. Retrieved February 10, 2016, from [ftp://131.252.97.79/Transfer/Treg/WFRE\\_Articles/Liaw\\_02\\_Classification and regression by randomForest.pdf](ftp://131.252.97.79/Transfer/Treg/WFRE_Articles/Liaw_02_Classification and regression by randomForest.pdf)
24. R, S. (2015). Performance Analysis of Different Classification Methods in Data Mining for Diabetes Dataset Using WEKA Tool. *International Journal on Recent and Innovation Trends in Computing and Communication IJRITCC*, 3(3), 1168-1173. doi:10.17762/ijritcc2321-8169.150361
25. S. Demura and M. Uchiyama, "Influence of moderate alcohol ingestion on gait," *Sport Sci Health*, no. 4, pp. 21-26, 2008.
26. Smartphone users worldwide 2014-2019 | Statistic. (n.d.). Retrieved April 20, 2016, from <http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>
27. National Highway Traffic Safety Administration. (n.d.). The ABCs of BAC A Guide to Understanding Blood Alcohol Concentration and Alcohol Impairment. Retrieved April 20, 2016, from <http://www.nhtsa.gov/links/sid/ABCsBACWeb/page2.htm>
28. Wang, K., Huang, M., Hsieh, Y., Lau, S., Yen, C., Kao, H. (., . . . Chen, Y. (2014). SoberDiary. Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct. doi:10.1145/2638728.2638847
29. Mayo Clinic. (n.d.). Alcohol. Retrieved April 20, 2016, from <http://www.mayomedicallaboratories.com/test-info/drug-book/alcohol.html>
30. Mathworks. (n.d.). Total Harmonic Distortion. Retrieved April 20, 2016, from <http://www.mathworks.com/help/signal/ref/thd.html>
31. Greenbaum, P.E.; Del Boca, F.K.; Darkes, J.; et al. Variation in the drinking trajectories of freshmen college students. *Journal of Consulting and Clinical Psychology* 73:229–238, 2005. Graph in NIAAA College Bulletin, NIH Publication No. 07–5010. Printed November 2007. Online at [http://www.collegedrinkingprevention.gov/1College\\_Bulletin-508\\_361C4E.pdf](http://www.collegedrinkingprevention.gov/1College_Bulletin-508_361C4E.pdf)
32. Substance Abuse and Mental Health Services Administration (SAMHSA) National Survey on Drug Use and Health. Underage Alcohol Use among Full-Time College Students. Issue 31, 2006. Online at <http://www.oas.samhsa.gov/2k6/college/collegeunderage.htm>
33. Qi, Muxi. "A Comprehensive Performance Comparison of Signal Processing Features in Detecting Alcohol Consumption from Gait Data." WPI Graduate Thesis, April 2016.
34. Mathworks. (n.d.). Bandpower. Retrieved April 20, 2016, from <http://www.mathworks.com/help/signal/ref/bandpower.html>

35. Trepn Power Profiler. (n.d.). Retrieved April 25, 2016, from <https://developer.qualcomm.com/software/trepn-power-profiler>

## APPENDIX A: Study Description / Consent Form for Data Collection



The University of  
Science and Technology.  
And Life.™

### Informed Consent Agreement for Participation in a Research Study

**Investigators:** Christina Aiello

**Contact Information:** [alcogait@wpi.edu](mailto:alcogait@wpi.edu)

**Title of Research Study:** Smartphone Gait Inference

#### Introduction:

You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

#### Purpose of the study:

In this experiment, we will gather data to investigate using a smartphone app to measure its owner's intoxication levels based on his or her walking patterns.

#### Procedures to be followed:

You will be given an Android smartphone, and you will be asked to place the smartphone in either your front or back pants pocket. I will then ask you to walk normally (no impairment) across a padded portion of the WPI Sports and Recreation Center floor while the application records the smartphone's gyroscope and accelerometer data. You will then be given various pairs of visual impairment goggles, goggles meant to distort your vision and alter your walking patterns in a fashion similar to alcohol. While wearing each pair of goggles (one at a time), you will be asked to walk for one minute across the WPI Sports and Recreation Center floor while gyroscope and accelerometer data is recorded. While you walk with the goggles, an investigator will walk with you to "spot" you if need be (guide you, offer a steadying hand, etc.).

**If you are feeling too dizzy or about to fall or for any other reason, you may stop the study by removing the goggles.**

#### Risks to study participants:

You may become slightly dizzy, unbalanced or uncomfortable wearing the vision-impairment goggles, goggles that will affect your vision and potentially your walking patterns. If at any time you are not comfortable participating in this study, you may cease participation. You do not give up any of your legal rights by signing this statement.

#### Benefits to research participants and others:

The overall results of this experiment will provide more accurate information to users on their intoxication levels without the hassle of manually recording their alcohol intake or activities.

**Record keeping and confidentiality:**

Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to confidential data that identifies you by name. Any publication or presentation of the data will not identify you.

**Compensation or treatment in the event of injury:**

This project has not budgeted for compensation to subjects in the event of injury. Subjects are encouraged to inform the investigator in the event of injury and to seek professional medical attention if symptoms are severe or/and persist.

**Cost/Payment:** n/a

**For more information about this research or about the rights of research participants, or in case of research-related injury, contact:**

WPI Institutional Review Board Chair: Professor Kent Rissmiller, Tel. 508-831-5019, Email: kjr@wpi.edu

WPI's University Compliance Officer: Michael J. Curley, Tel. 508-831-6919, Email: mjcurley@wpi.edu

Primary Investigator: Professor Emmanuel Agu, Email: emmanuel@cs.wpi.edu

Student Investigators: Christina Aiello, Email: cjaiello@wpi.edu

**Your participation in this research is voluntary.** Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. If you choose to do so, you will be given the option to erase all previous data and to have it not used in the study. There will not be any repercussions from the university, including grades or academic standing. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

**By signing below,** you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

\_\_\_\_\_ Date: \_\_\_\_\_

Study Participant Signature

\_\_\_\_\_

Study Participant Name (Please print)

\_\_\_\_\_ Date: \_\_\_\_\_

Signature of Person who explained this study

## APPENDIX B: Demographic Information for Study Participants

User_ID	Weight	Gender (0 for female, 1 for male)	Age	Height
1448713	118	0	20	152.4
3137562	170	1	22	185.42
5817268	179	1	19	185.42
2030631	173	1	19	190.5
4048271	150	1	19	177.8
1015115	170	0	21	165.1
3424592	250	1	21	182.88
814834	175	0	21	157.48
708358	115	0	22	172.72
3330820	125	1	19	170.18
655673	152	0	21	167.64
843814	145	1	20	170.18
452182	142	0	21	154.94
2305555	110	0	18	172.72
3802493	160	0	20	160.02
1329047	145	0	19	165.1
2243516	150	1	19	175.26
3221270	185	1	19	177.8
2142625	150	0	21	182.88
1848768	119	0	18	168.91
1828585	230	1	21	187.96
2016057	205	0	18	170.18
1808421	120	0	19	160.02
1748862	143	0	18	175.26
1637144	130	0	22	162.56
1905299	150	0	22	167.64
1506627	155	1	20	185.42
1520109	200	1	20	177.8
219171	128	0	21	172.72
5700722	145	1	20	180.34
3500284	150	0	22	175.26
3534537	165	1	22	180.34
2116907	150	0	19	152.4
1134647	120	0	19	170.18



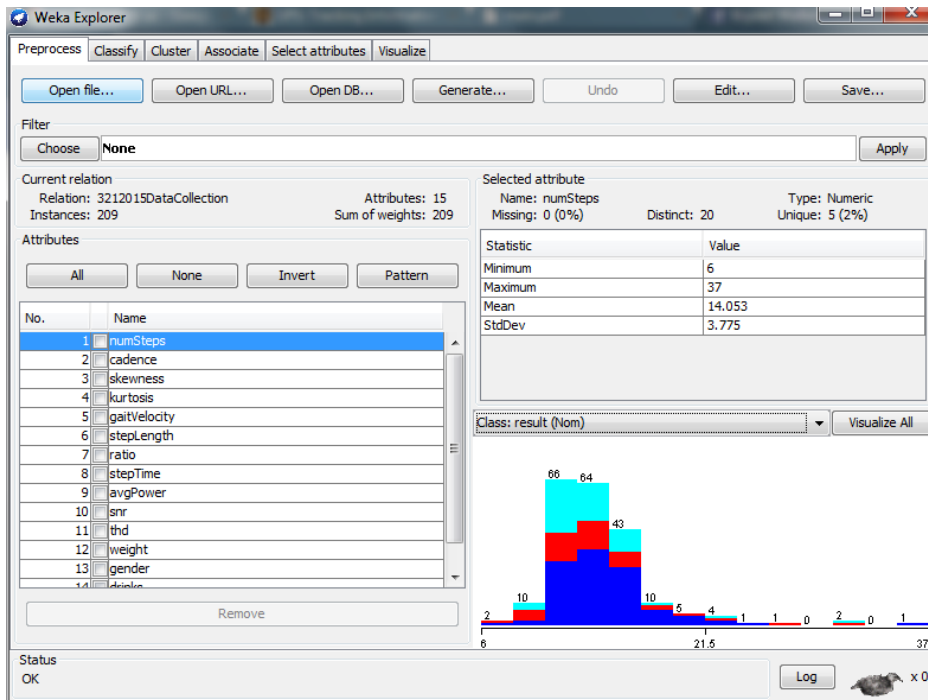
## APPENDIX C: Using Weka for Classification

### Classifying Data Using Weka

Now that you have your data in a CSV file, it needs to be classified! You can manually classify it using Weka.

**Note:** The result category in the CSV file is the one you want to try to predict. 'a' is the first possibility (least amount of drinks), 'b' is the second possibility, and 'c' is the third.

1. On the "Preprocess" tab, click "open file" and select your CSV file.
2. Choose "result" as your target. Your target MUST be nominal (not a number) for this to work properly. The graph should look like this, with three different colors, one for each possible result:



3. Next, go to the "Classify" tab. Choose a classifier with the "Choose" button. (Random Forest is a good one to use.)
4. Now choose test options. (Cross-validation with 10 folds is a good option.)
5. Be sure that "result" shows up on the dropdown above the "Start" button.
6. Click the "Start" button.

7. The output should be something like this:

```
=== Run information ===

Scheme:      weka.classifiers.trees.RandomForest -I 10 -K 0 -S 1 -num-slots 1
Relation:    3212015DataCollection
Instances:   209
Attributes:  15
             numSteps
             cadence
             skewness
             kurtosis
             gaitVelocity
             stepLength
             ratio
             stepTime
             avgPower
             snr
             thd
             weight
             gender
             drinks
             result
Test mode:   10-fold cross-validation

=== Classifier model (full training set) ===

Random forest of 10 trees, each constructed while considering 4 random features.
Out of bag error: 0.0625

Time taken to build model: 0.06 seconds

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

Correctly Classified Instances      203          97.1292 %
Incorrectly Classified Instances     6           2.8708 %
Kappa statistic                    0.9533
Mean absolute error                 0.1277
Root mean squared error             0.1935
Relative absolute error             31.0004 %
Root relative squared error         42.6522 %
Coverage of cases (0.95 level)     100          %
Mean rel. region size (0.95 level) 67.9426 %
Total Number of Instances          209

=== Detailed Accuracy By Class ===

             TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
0.991  0.029  0.972   0.991  0.981   0.962  0.999  0.999    a
0.933  0.018  0.933   0.933  0.933   0.915  0.994  0.975    b
0.965  0.000  1.000   0.965  0.982   0.976  0.997  0.991    c
Weighted Avg.  0.971  0.019  0.972   0.971  0.971   0.956  0.998  0.992

=== Confusion Matrix ===

  a  b  c  <-- classified as
106  1  0 | a = a
  3 42  0 | b = b
  0  2 55 | c = c
```

8. Look at the confusion matrix. Things in the (a,a) spot were correctly classified as 'a.' Anything else in the 'a' column was classified incorrectly. Same with (b,b) and (c,c).
9. In the SGI application, there is a jar (link below) that is Weka ported to Android.

## APPENDIX D: Setting Up Windows IIS

On Windows 7, go to "Start," "Control Panel," "Programs and Features," "Turn Windows features on or off," and enable everything in Internet Information Services and Internet Information Services Hostable Web Core (be SURE to be serving static content to show web pages, and be sure to enable other settings so you can use executables on the server)

Configurations:

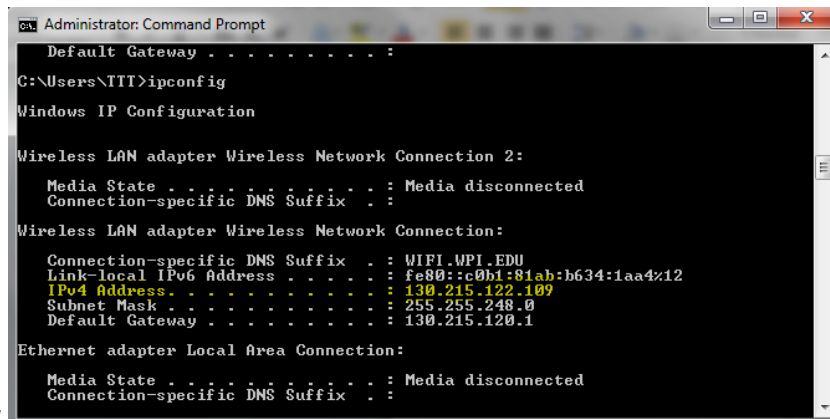
- Project location: Use C:\inetpub\wwwroot\alcogait
  - Put the aspnet\_client folder from wwwroot into this folder
- Use Web Platform Installer (Google it to download it) to install PHP on the server
- Default Document: Set "Default Document" to some index page, like index.html or index.php (So with the above link, it would go in the alcogait folder)
- Bindings:
  - http {blank} 80 130.215.28.180 (whatever comes up in the dropdown during setup)
  - http www.alcogait.com 80 130.215.28.180
  - http alcogait.com 80 130.215.28.180
- Disabling Firewall to Access Website from Outside PC
  - Go to "Windows Firewall with Advanced Security"
  - Click "New rule"
  - Say connections on all ports are fine
  - Name and save rule
- Getting URL Rewrite to Work
  - Translate htaccess file to web.config
  - Install URL Rewrite add on: <https://www.microsoft.com/web/gallery/install.aspx?appid=urlrewrite2>

## APPENDIX E: Instructions to Run AlcoGait Thesis Project

(By Christina Aiello)

### How to Gather Data Via Matlab

1. Close all CSV files that the program will use:
  - i. User profile (IDsAndParticipants) file
  - ii. Raw data files
  - iii. Generated features files
2. (If off campus) Get VPN running on both computer and mobile device
  - a. On computer, use network connect with:
    - i. *vpn.wpi.edu*
    - ii. Your *WPI email address (including @wpi.edu at the end)*
    - iii. *CCC password*
  - b. On phone, use Pulse Secure with:
    - i. *vpn.wpi.edu*
    - ii. Your *WPI email address (including @wpi.edu at the end)*
    - iii. *CCC password*
3. In Matlab on computer, type command “connector on” into terminal
4. On phone in Matlab app, go to “Connect to Your Computers” and type in IP address. Then click “Connect”
  - a. If on campus, open up the command prompt and type “ipconfig” and look for the wireless IP address (Wireless LAN adapter Wireless Network Connection, IPv4). See text highlighted in



```
Administrator: Command Prompt
Default Gateway . . . . . :
C:\Users\TIT>ipconfig
Windows IP Configuration

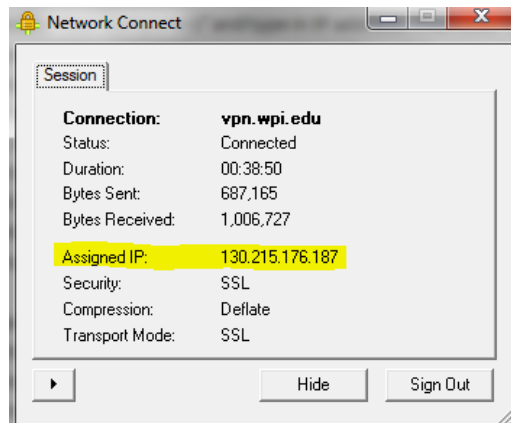
Wireless LAN adapter Wireless Network Connection 2:
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection:
    Connection-specific DNS Suffix  . : WIFI.WPI.EDU
    Link-local IPv6 Address . . . . . : fe80::e0b1:81ab:b634:1aa4%12
    IPv4 Address. . . . . : 130.215.122.109
    Subnet Mask . . . . . : 255.255.248.0
    Default Gateway . . . . . : 130.215.120.1

Ethernet adapter Local Area Connection:
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :
```

yellow below.

- b. If off campus and using VPN, go to the Basic View (right click VPN icon in bottom right menu) of the VPN service to see the IP address. See text highlighted in yellow below:



5. Note: Matlab app must be the active app on the phone. Screen can be locked or unlocked.
6. Either type "runAlcoGait" into the command line or open the runAlcoGait file and click run
  - a. Click "add to PATH" button if need be (if you get a message on your screen about needing to add runAlcoGait to the PATH)
7. Type 1 for first prompt to collect new data
8. Participant ID is next.
  - a. If this is a new participant, type Y to create a profile for this individual. The profile will include:
    - i. The participant's name
    - ii. The participant's gender
    - iii. The participant's height in feet and inches
    - iv. The participant's weight in pounds
    - v. The participant's age
  - b. If this is an old participant, type that participant's ID now.
9. Type the participant's BAC
10. Type the number of five-second trials that you would like to do
11. Wait for the "Begin!" and once you hear that, start.
12. Wait for the four "ding!" sounds to know that the trials are done.

### Data Locations:

(These are all hard-coded into the main runAlcoGait() method, so you can change them there to whatever you want them to be.)

### User profiles:

- File path is written in *getData.m* file
  - Currently hard-coded

### Raw data:

- Should be two files per set of trials, one for accelerometer data and one for gyroscope data

- File path is written in *createEllipse.m* file
  - Currently hard-coded

#### Generated features:

- Should be one csv file per participant, based on UUID
- File path is written in *analyzeAndLogData.m* file
  - Currently hard-coded

#### Sway area images:

- Should be 3 images per trial (not per set of trials, but per trial)
- File path is written in *dataCollection.m* file
  - Currently hard-coded

#### Steps to Normalize Data

1. In Matlab, either type “getData” into the command line or open the getData file and click run.
2. Type “3” to normalize data.
3. Give a new name for this file. This will be saved into the appropriate directory.
4. Give the full file path for the not-normalized generated features.

## Steps to Classify Data (in Desktop Weka)

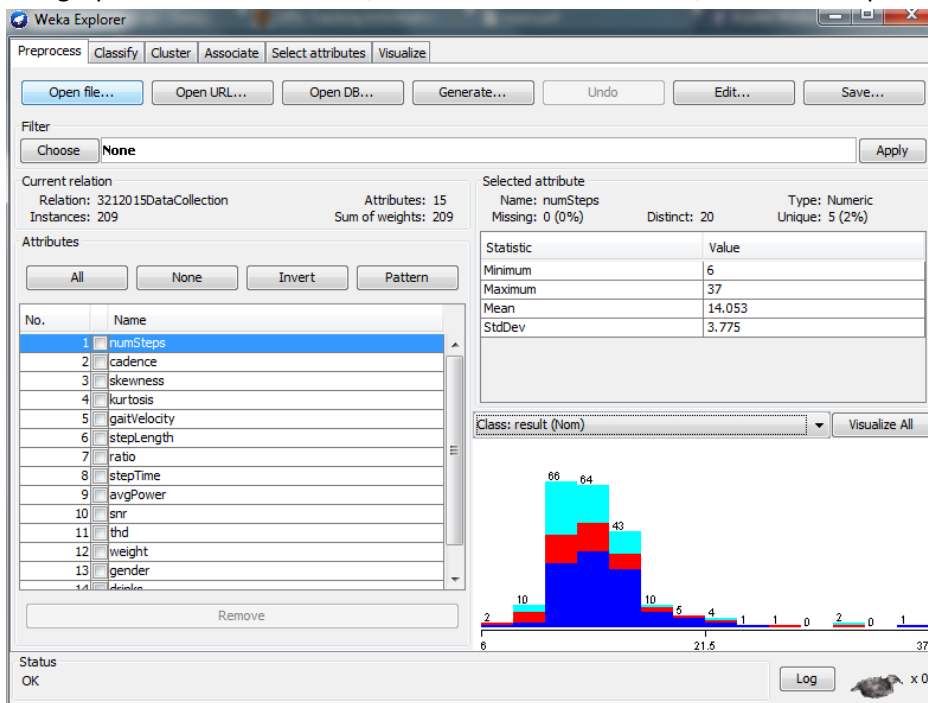
### First, in Matlab:

1. In Matlab, either type “getData” into the command line or open the getData file and click run.
2. Type “4” to combine all of your data into one file.
3. Give a new name for this file.
4. Give the full file path for the not-normalized generated features.
5. If any of the data says “#NAME?” or “=-Inf” you need to replace these with NaN

Now that you have your data in a CSV file, it needs to be classified! You can manually classify it using Weka.

**Note:** The result category in the CSV file is the one you want to try to predict. ‘a’ is the first possibility (least amount of drinks), ‘b’ is the second possibility, ‘c’ is the third, and ‘d’ is the fourth (and it’s the highest BAC range).

10. On the “Preprocess” tab, click “open file” and select your CSV file.
  11. Choose “result” as your target. Your target MUST be nominal (not a number) for this to work properly.
- The graph should look like this, with three different colors, one for each possible result:



12. Next, go to the “Classify” tab. Choose a classifier with the “Choose” button. (Random Forest is a good one to use.)
13. Now choose test options. (Cross-validation with 10 folds is a good option.)
14. Be sure that “result” shows up on the dropdown above the “Start” button.
15. Click the “Start” button.

16. The output should be something like this:

```

=== Run information ===

Scheme:      weka.classifiers.trees.RandomForest -I 10 -K 0 -S 1 -num-slots 1
Relation:    3212015DataCollection
Instances:   209
Attributes:  15
             numSteps
             cadence
             skewness
             kurtosis
             gaitVelocity
             stepLength
             ratio
             stepTime
             avgPower
             snr
             thd
             weight
             gender
             drinks
             result
Test mode:   10-fold cross-validation

=== Classifier model (full training set) ===

Random forest of 10 trees, each constructed while considering 4 random features.
Out of bag error: 0.0625

Time taken to build model: 0.06 seconds

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

Correctly Classified Instances      203          97.1292 %
Incorrectly Classified Instances     6           2.8708 %
Kappa statistic                    0.9533
Mean absolute error                 0.1277
Root mean squared error            0.1935
Relative absolute error             31.0004 %
Root relative squared error        42.6522 %
Coverage of cases (0.95 level)    100          %
Mean rel. region size (0.95 level) 67.9426 %
Total Number of Instances          209

=== Detailed Accuracy By Class ===

              TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
              0.991   0.029   0.972     0.991   0.981     0.962   0.999    0.999    a
              0.933   0.018   0.933     0.933   0.933     0.915   0.994    0.975    b
              0.965   0.000   1.000     0.965   0.982     0.976   0.997    0.991    c
Weighted Avg.  0.971   0.019   0.972     0.971   0.971     0.956   0.998    0.992

=== Confusion Matrix ===

  a  b  c  <-- classified as
106  1  0 | a = a
  3 42  0 | b = b
  0  2 55 | c = c

```

17. Look at the confusion matrix. Things in the (a,a) spot were correctly classified as 'a.' Anything else in the 'a' column was classified incorrectly. Same with (b,b) and (c,c).

## How To Get Model (.model file) From Desktop Weka to Put Into Java Application

- When on the “Classify” tab (in Explorer), click “More Options”
- Check the “Output source code” box
  - And copy & paste that code into Android Studio
- Also make sure “Output Model” (at the top of the box) is checked



- After clicking “OK” and “Start,” right click the result in the left pane and click “Save Model.”
- Then you need to copy and paste that .model file into Android Studio