Topic Title: Data Annotation Methods for Smartphone and Physiological Signal Based Emotion Recognition

Student Name: Andrew Benjamin Barnett **Student ID:** z3418876

A. **Problem statement**

Effective annotation of data with appropriate labels is an integral part of any supervised machine learning system. This study investigates the application of three different data annotation methods to emotion recognition using phone and physiological data in real world unconstrained contexts (RWUCs). The two most common annotation methods used in these contexts are 'on the spot' ratings of emotion and 'end of day' reviews. To date, no work has been done to experimentally compare these methods, or to combine the two, and as such these are the first issues this study will cover. Similarly, the use of pre-labelled stimuli to elicit emotions is another common method for annotation, however, as this approach does not appear to be directly applicable to real world contexts its potential use in such contexts has not been explored either, another issue this study will address.

B. **Objective**

Review relevant literature in the field of emotion recognition (ER) using smartphone or physiological data, particularly in regards to annotation methods.
Implement system to collect smartphone, phys, and annotation data of multiple forms.
Implement system for lab trained physiological classifier to assist ER in RWUCs
Experimentally evaluate effectiveness of the above mentioned systems and methods

C. **My solution**

Phone data and annotation data in 3 forms (ratings, reviews, and combined) collected using a specially developed Android app which used Android System APIs and EmotionSense lib.
Physiological data collected using Empatica E4 wristband.
Wide range of features computed using Java program for Phone data, Matlab for phys data.
Lab training using pre-annotated images from NAPS database.
SVM and KNN to evaluate performance of annotation types using Scikit Learn in Python.

D. **Contributions** (at most one per line, most important first)

Experimental comparison of data annotation types in real world unconstrained context
Proposed and demonstrated novel combined data annotation technique
Proposed and implemented system for physiological lab training
Implemented Android application for phone data and annotation data collection
Proposed and demonstrated novel approach for reminding users while doing reviews.
Higher time scale granularity than previously achieved in similar systems
Presented varied feature extraction systems for both phone and physiological data
Implemented classifier suite to investigate annotation methods in a range of settings.
Data set collected in a real world unconstrained setting for one individual and annotated with emotion labels for a period of around one month.

E. **Suggestions for future work**

Wider range of participants
Investigating other dimensions of emotion e.g. arousal or dominance
Investigating real world applications of these annotation methods

While I may have benefited from discussion with other people, I certify that this report is entirely my own work, except where appropriately documented acknowledgements are included.

Signature: A.B. Barnett Date: 27/11/2016

Pointers

List relevant page numbers in the column on the left. Be precise and selective: Don't list all pages of your report!

8-9	Problem Statement
9	Objective

Theory (up to 5 most relevant ideas)

12	Emotion models
12-13	Annotation methods
17-18	Feature extraction
18	Physiological lab trained models
20	Validation methods

Method of solution (up to 5 most relevant points)

22-23	System design overview
38-45	Data collection app
45-46, 53	Feature extraction
57-62	Lab trainer
62-64	Classification suite

Contributions (most important first)

74-79	Experimental comparison of annotation types in real world unconstrained context
48	Proposed and demonstrated novel combined data annotation technique
57-62	Proposed and implemented system for physiological lab training
38-45	Implemented Android application for phone data and annotation data collection
44-45	Proposed and demonstrated novel approach for reminding users doing reviews.
45-57	Presented varied feature extraction systems for both phone and physiological data

My work

23	System block diagrams/algorithms/equations solved
73-74	Description of assessment criteria used
36-38	Description of procedure (e.g. for experiments)

Results

74-80	Succinct presentation of results
81-85	Analysis
81-85	Significance of results

Conclusion

87	Statement of whether the outcomes met the objectives
85-86	Suggestions for future research

Literature: (up to 5 most important references)

13	3.	Constantine, L. and H. Hajj. 2012
11, 13, 15	2.	LiKamWa, R., et al. 2013
17, 15, 20	4.	Jaques, N., et al. 2015
18	8.	Haag, A., et al. 2004
17, 15	5.	Sano, A., et al. 2015

THE UNIVERSITY OF NEW SOUTH WALES

SCHOOL OF ELECTRICAL ENGINEERING AND TELECOMMUNICATION

Data Annotation Methods for Smartphone and Physiological Signal Based Emotion Recognition

By

Andrew Barnett

Thesis submitted as a requirement for the degree

Bachelor of Engineering (Electrical Engineering)

Submitted: 28 November 2016

Supervisor: Vidhyasaharan Sethu

Student ID: z3418876

Course Code: ELEC4121 – Thesis B

Abstract

Effective annotation of data with appropriate labels is an integral part of any supervised machine learning system. This study investigates the application of three different data annotation methods for emotion recognition using phone and physiological data in real world unconstrained contexts. The two most common annotation methods used in these contexts are 'on the spot' ratings of emotion and 'end of day' reviews. To date, no work has been done to experimentally compare these methods, or to combine the two, and as such these are the first issues this study will cover. Similarly, the use of pre-labelled stimuli to elicit emotions is another common method for annotation, however, as this approach does not appear to be directly applicable to real world contexts its potential use in such contexts has not been explored either, another issue this study will address. Firstly, in order to compare annotation methods a collection module was created to gather both phone and annotation data of various types, along with physiological data obtained using an existing device. Secondly a lab based module to pre-train a physiological model using pre-annotated stimuli was also developed. Lastly in order to evaluate the effectiveness of these modules a full machine learning system was implemented including further feature extraction and classification modules. The results of this system showed reviews to be the most successful at recording simple emotion, while more detailed emotion was best recorded with a combined method developed for this study. Lab trained models showed some potential for enhancement of prediction but more work is required to reach conclusive results. Using the findings of this study, future works can make a more informed choice of data annotation techniques to suit their desired annotation scale and the amount of input that can be expected from users.

Contents

1. Introduction 7
1.1. Background 7
1.2. Motivations 8
1.3. Aims 8
1.4. Outline 9
2. Literature Review 11
2.1. Emotion Theory 11
2.1.1. First and Second Order Emotions 11
2.1.2. Emotion and Mood 11
2.1.3. Emotion Induction Context 11
2.1.4. Emotion Models 12
2.2. Collection 12
2.2.1. Annotation Methods 12
2.2.2. Modalities 14
2.2.3. Collection Methods 16
2.3. Feature Extraction 17
2.4. Physiological Lab Trained Models 18
2.5. Classification 18
2.5.1. Feature Selection 19
2.5.2. Machine Learning Algorithms 19
2.5.3. Validation Methods 20
2.5.4. Evaluation Methods 21
3. System Design 22
3.1. Design goals 22
3.2. System Overview 22
3.3. Collection Module Design 24
3.3.1. Requirements 24
3.3.2. Inputs 25
3.3.3. Outputs 26
3.4. Feature Extraction Module Design 26
3.4.1. Requirements 28
3.4.2. Inputs 28
3.4.3. Outputs 29
3.5. Lab Training Module Design 29

3.5.1. Requirements 30
3.5.2. Inputs 31
3.5.3. Outputs 31
3.6. Classification Module Design 32
3.6.1. Requirements 33
3.6.2. Inputs 33
3.6.3. Outputs 33
3.7. Design Benefits 34
3.8. Design Limitations 34
4. System Implementation 36
4.1. Collection Module Implementation 36
4.1.1. Emotional Model 36
4.1.2. Participants 37
4.1.3. Collection of Physiological Signals 37
4.1.4. Smartphone Data and Annotation Collection 38
4.2. Feature Extraction Module Implementation 45
4.2.1. Smartphone Feature Extraction 45
4.2.2. Physiological Feature Extraction 53
4.2.1. Overview 53
4.2.2. Processing 54
4.2.3. Attributing Samples 54
4.2.4. Features 55
4.2.5. Output 56
4.2.6. Windowed Feature Extraction 56
4.3. Lab Training Module Implementation 57
4.3.1. Image Database Preparation 57
4.3.2. Lab Trainer 57
4.3.3. Physiological Data Preprocessor 61
4.3.4. Lab Classifier Trainer 61
4.4. Classification Module Implementation 62
4.4.1. Overview 62
4.4.2. Setting Selection and List Initialisations 63
4.4.3. Preparing Features 63
4.4.4. Training and Validation Set Separation 64
4.4.5. Machine Learning Algorithms and Hyper-parameters 65
4.4.6. Feature Selection and Transformation 66

4.4.7. Performance Metrics 66
4.5. System Benefits 67
4.6. System Challenges 68
4.7. System Limitations 70
5. Results 73
5.1. Annotation Type Results 74
5.1.1. Overall 74
5.1.2. Scale Type 76
5.1.3. Classifier Type 77
5.1.4. Feature Type 79
5.2. Feature Type Results 80
6. Discussion 81
6.1. Annotation Type Discussion 81
6.1.1. Overall Results 81
6.1.2. Scale Type 82
6.1.3. Classifier Type 83
6.1.4. Feature Type 84
6.2. Lab Training and Feature Type Discussion 84
6.3. Comparison with Previous Works 85
6.4. Suggestions for Future Work 85
7. Conclusions 87
8. Bibliography 88
Appendix A 90
Appendix B 91

Abbreviations

API	Application Programming Interface
OS	Operating System
UI	User Interface
MIT	Massachusetts Institute of Technology
CPU	Central Processing Unit
JSON	JavaScript Object Notation
CSV	Comma Separated Value
KNN	K Nearest Neighbours
SVM	Support Vector Machine
PCA	Principle Component Analysis
SFFS	Sequential Forward Floating Selection
MSE	Mean Squared Error
MAE	Mean Absolute Error
LOOCV	Leave One Out Cross Validation
KFCV	K Fold Cross Validation
HOV	Held Out Validation

1. Introduction

1.1. Background

Emotion recognition systems make use of machine learning algorithms to look for patterns in data in order to try and predict when a user is happy or sad, or when they are calm or stressed. Smartphone data, such as call logs or app usage statistics, and physiological bio-signal data, such as heart rate or skin temperature, both contain potentially useful information which we can use to predict emotion. However, in order for machine learning algorithms to do this prediction successfully we require a large amount of training data which is correctly labelled so that we know what kind of emotion each segment of the data we collect corresponds to.

Practical applications of emotion recognition such as emotion tracking for mental health patients, or adaptive services like mood based music players generally require ongoing recognition of emotions in a real world and unconstrained context. This is very different to a lab environment where many variables can be controlled or observed and data is simple to collect. Both smartphone and physiologically based data collection methods are well suited to this task [1]. Many users have their phone on their person, or at least close by, 24 hours a day, seven days a week. This constant proximity can allow such devices to collect a wealth of data about their owners and their owners' communications, habits, movements, and preferences. Similarly, many users are beginning to adopt wearable devices such as fitness tracking wristbands or smart watches which are capable of recording physiological signals. As a result both physiological and smartphone information can be collected from users without the need for any additional hardware and in a manner that is almost completely unobtrusive [2]. However, while data may be becoming easier to collect (and thus attractive as means of recognising emotion) this data is of little use for training a machine learning algorithm unless accurately labelled, a task which is particularly difficult in the sorts of unconstrained contexts we are talking about.

Finding an accurate way to label data with its 'true' value is often referred to as 'establishing a ground truth'[3] and is a recurring issue in studies around emotion recognition. The approaches typically used depend heavily on the context in which emotion is being studied. If a study is performed in a lab analysing emotions that are induced by a known stimulus, then the label applied to the data can simply be derived from the label given to the stimulus. However, if the study is being performed in an everyday natural setting, where the stimuli are uncontrolled, then labelling becomes a much less trivial task.

1.2. Motivations

Previous studies involving emotion recognition in the real world generally adopt one of two approaches to data annotation, both of which are a form of self-assessment. The first is the use of intermittent 'on the spot' ratings where a user is prompted at multiple points throughout the day to rate how he/she is currently feeling. This has been implemented in some phone based applications such as Moodscope, an iOS application for creating mood calendars [2]. The second approach is to ask the user to complete retrospective surveys at the end of their day to describe how they were feeling at various times. This is the approach that has been taken by various studies completed by teams at the MIT Affective Computing Lab to investigate both physiological and phone based recognition [1, 4-6]. Both of these commonly used annotation approaches have pros and cons, but there has been little work done to assess or compare their effectiveness experimentally. What's more, the time scale granularity of these labelling schemes is quite low, generally labelling data in day long blocks with the smallest time windows being around 4 hours [2].

Aside from self-assessment the other main annotation technique that is adopted in many emotion recognition systems is the use of pre-labelled stimuli to illicit certain emotions. This method has been used extensively in the lab environment with stimuli such as images [7, 8] and short film clips [9], however, little work has been done to try and apply this technique to help with real world classification. Clearly it is unhelpful to provide the user with a pre-labelled stimulus while they are going about their daily activities and then use the associated label to annotate phone data, because the resulting induced emotion would not be a product of their real-world context. However, there could be potential for the use of these pre-labelled stimuli to pre-train models in a lab setting which could then assist with classification in a more general everyday setting.

Accurate annotation of data is one of the foundations of machine learning and is just as important as the collection of the data itself. If the labels are wrong or inconsistent then a machine learning algorithm is unlikely to be successful in achieving accurate predictions [3]. Thus, investigation into the effectiveness of annotation methods is highly warranted.

1.3. Aims

In a broader sense the overall challenge of this study is to investigate the effectiveness of a range of data annotation methods for real world unconstrained settings where smartphone and physiological data is collected. Effectiveness will be used to refer to how good the annotation method is at labelling data in a way that allows for meaningful prediction. There are two ways that this will be

achieved. The first is a comparison of self-assessment annotation techniques including 'on the spot' ratings, retrospective reviews, and then a method of combining the two. This comparison will allow an investigation into which contexts the various techniques are most effective in. Secondly the study aims to explore the potential of using pre-annotated data to pre-train models in a constrained laboratory setting which can then be used to assist classification in a real-world context.

The hypothesis was that a continuous on the spot rating system would be able to achieve greater effectiveness than retrospective reviews due to the likelihood of users forgetting their exact feelings from throughout the day. However, it was expected that combining the two methods would result in an even more effective system due to the ability to label data more finely thus making each label less generalised and more relevant to its associated data. It was also expected that a classifier which makes use of a lab-trained model would be able to achieve more effective prediction than one without, due to the increased amount of information available to the classifier to make its predictions. As a result of these investigations it was hoped that is would be possible to increase the time scale granularity at which we are able to annotate while still achieving effective prediction.

1.4. Outline

In order to experimentally investigate the effectiveness of various data annotation methods it is necessary to create a full machine learning system to see how these methods perform in practice. As such, though the focus is on annotation methods, much of this study is also dedicated to producing this machine learning pipeline. The approach which will be explored can be broken down in to four stages as seen in Figure 1 below. The first of these is collection, referring to both data and labels. The second is feature extraction which refers to the extraction of useful measures and quantities from the collected data which can then be used as input to a machine learning classifier. Thirdly, in parallel to these first 2 stages is the physiological lab-training stage that attempts to train an extra classifier model that may be used to assist in the subsequent overall classification. The final stage is classification which takes in the features extracted in the second stage, along with a lab-trained classifier model, to create its own new classifier models which attempt to predict the emotions associated with each input feature vector. Many sections of this report will be logically grouped into these 4 stages.

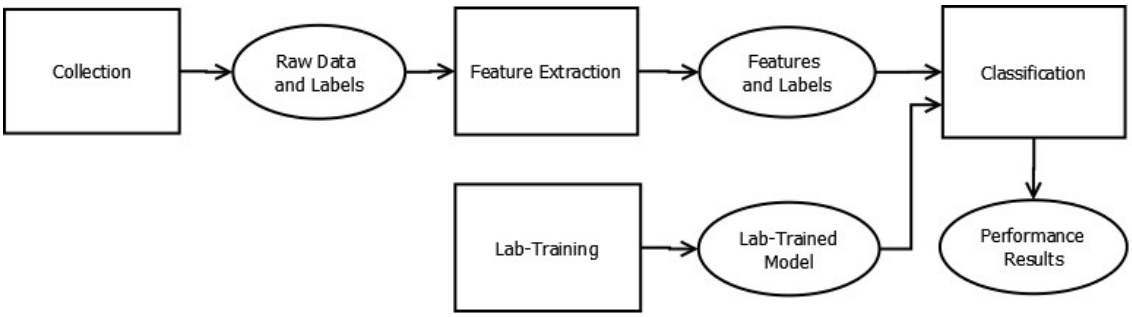

Figure 1 – Overview of Basic Stages

This study starts off by conducting a review of the relevant literature in our field. To ensure the reader has all the background knowledge required to understand the work that shall be outlined, we begin with a review of general emotion theory. Following this is a review of the literature relevant to each stage of the system, these stages again being collection, feature extraction, physiological lab-training, and classification. The report then goes on to outline the overall system design that has been chosen to allow for the most effective investigation of the various annotation methods, before delving in to the design of each stage of the system in more depth. A description of the methods employed to implement the designed system is then provided. Finally, the results collected from the system indicating the performance of the various annotation methods are outlined before being analysed in a discussion section.

From this work and analysis, we are able to draw conclusions about which of the contexts 'on the spot' ratings, retrospective reviews, or a combination of the two methods are most successful at predicting emotion. We are also able to demonstrate the usefulness of this combined method for the first time, as well as providing a full android application implementing these 3 annotation types (on a higher time scale granularity than previously achieved) and collecting smartphone data. As well as comparing these methods a system for the application of physiological lab-training to real world data is also designed and tested.

2. Literature Review

2.1. Emotion Theory

The basic concept of emotion requires little explanation, as it essentially refers to the instinctive or intuitive feelings that most humans experience. However, when we investigate emotion in a scientific context, as is being done in this study, there are a number of subtleties that are worth considering, and which will be explored in this section.

2.1.1. First and Second Order Emotions

First it is important to note that a distinction must be made between the first and second order emotional experience [10]. First order emotion refers to the underlying phenomenological aspect of our feelings which may not be something that a subject consciously understands. On the other hand, second order emotion refers to the feelings that we are consciously aware of. While the two are often closely linked, it must be remembered that they are also separate. Clearly if a subject is only aware of second order emotion it is very difficult to obtain ground truth data for first order emotions and as such this study will choose to focus its attention on the second order experience.

2.1.2. Emotion and Mood

Secondly, many studies make a distinction between emotion and mood and this is worth considering. Emotion is often considered as a more spontaneous reaction to stimuli and is frequently studied in relation to physiological signals such as in [7-9, 11, 12]. Mood on the other hand is the result of a cumulative sequence of events over a longer period of time and has been the focus of studies such as the Moodscope application [2]. While mood is generally considered less intense it is also considered longer lasting [2]. The distinction between the two concepts isn't clear cut and there is obviously an interplay between them. This study will attempt to discern the emotional response which lies almost in the middle ground between both concepts, with the intention of moving more towards the concept of emotion by increasing the time scale granularity. Nonetheless, both concepts can be classified and detected similarly so for the purposes of this study the term emotion will be used to refer to mood also.

2.1.3. Emotion Induction Context

Thirdly, it is important to consider the context in which an emotion is induced. Three main types of emotion induction strategies exist as is explained in [3]. These are acted, reacted and interacted. Acted, as the name suggests, involves subjects pretending or acting as if they feel a chosen emotion. Reacted, on the other hand, makes use of specially chosen stimuli to induce an emotion in a subject such as the International affective picture system [13] and the Geneva affective picture

database[14]. Finally, interacted refers to emotions that are elicited through interactive scenarios or everyday natural settings. It is worth noting that both reacted and interacted are the result of external stimuli and for this reason these are the two methods that this study will focus on.

2.1.4. Emotion Models

Finally, it is also important to define the way in which emotions are represented. There are two common representations (or models) used for emotions [3, 15] . The first is a set of discrete classes, such as Clynes' set of eight basic emotions. In this case the model classifies every sample into one of the categories in the set e.g. anger, fear, joy, or sadness. This model was used more in early research into emotion recognition such as [16]. The second representation is a continuous dimensional model often referred to as the circumplex model. This model consists of two continuous axes, valence (positive or negative) and arousal (level of excitement or "activeness") [17]. Each sample is classified by assigning it a value for each axis independently. This places all samples in a 2-dimensional plane. Regions of this plane can be chosen to represent specific emotions as can be seen in Figure 2 taken from [2]. Note that valence has been replaced with pleasure, and arousal with activeness. A third axis also exists in some studies called "dominance", however, it is seldom used.

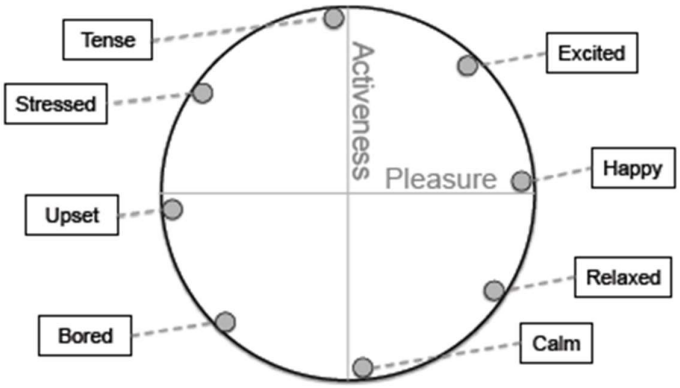

Figure 2 - The circumplex model [2]

2.2. Collection

2.2.1. Annotation Methods

The labelling of data with an accurate ground truth is a very important step in the machine learning process and one that we will focus much attention on in this study. There are a number of methods used, and the choice depends on the setting in which the study is taking place.

Studies carried out in the lab or constrained environments can quite easily make use of pre-annotated stimuli for the purpose of labelling. In such cases an emotion is induced by providing the

subject with a stimulus, such as a film clip, and then the data collected is given the same label as the stimulus which has been pre-defined. Examples include images such as those in the International Affective Picture System [13], audio such as that in the International Affective Digital Sounds [3], video clips or short movies [9, 18], or music [12]. It is also possible to use pre-classified stimuli in natural contexts by orchestrating specific emotion inducing events for a subject, for example arranging for a close friend to call them with good news [3]. This approach, however, could not be repeated frequently without arousing suspicion in the subject and also requires significant planning and organisation.

The most common approach for labelling data in natural settings, however, is through self-assessments performed by the test subject. These self-assessments generally come in two forms. The first is 'on the spot' ratings which ask the user at various points throughout the day how they are feeling at the current time. These could also come in the form of a push-notification on the subject's phone asking them to rate their mood on a simple scale [2]. The second approach is to ask the user to complete retrospective surveys at the end of their day to describe how they were feeling at a range of times [1, 4-6].

Most of the related studies undertaken to date have used one of these two methods, however they have limitations. Firstly in the case of 'on the spot' ratings it is possible that by interrupting the user to ask them to respond to questions on their current mood state we might be changing or affecting this state through the mere action of the interruption [10]. Similarly, users may forget to respond when they are busy and there is a limit to how regularly we can reasonably expect a user to answer such questions before our system loses the property of being unobtrusive. For these reasons using such a system for labelling data limits the temporal resolution that we can achieve (the minimum time interval that we can break a day up into). On the other hand, for end of day reviews, while the user is not at risk of being interrupted since they can choose to submit reviews at a time convenient for them, there is a chance that the subtleties of the emotions they experienced might be forgotten. A further option for self-assessment is for the model to ask for clarification only when it is unsure or predicts a change in emotion state [2].

Another method sometimes used for obtaining ground truth is the use of trained judges to decide on the emotion being displayed [3]. Clearly in a natural setting it becomes difficult to capture enough observational data for such a judge to review and this rules out the feasibility of this in our application.

2.2.2. Modalities

Modalities are the actual indicators upon which a machine learning algorithm can base its predictions. They represent the raw data that a system needs to collect. Modalities of various forms have been successfully used for the task of emotion recognition. The most commonly used are physiological signals, speech and audio based signals, image and video signals and more recently smartphone sensor signals.

While speech and image processing are both well-established techniques for emotion recognition this study has chosen not to focus much attention on them. Neither approach is particularly well suited to an ambulatory environment [16] due to the more obtrusive nature of sensors required such as microphones (with cords and added hardware) and cameras (which are difficult to place). As investigating annotation methods in continuous real world unconstrained environments is our primary aim, it was decided that these options were less appropriate. Of course, with more complex systems and specialised hardware these difficulties could be overcome.

2.2.2.1. Physiological Modalities

Studies that make use of physiological modalities [4, 5, 7, 9, 11, 12, 16, 19] tend to use similar sets of signals. The most common signals appear to be heart rate (often from blood volume pressure), skin conductance (or galvanic skin response), and skin temperature. Other signals which are often used are respiration measures such as breathing rate, electromyography signals (EMG), and electroencephalography signals (EEG), however the sensors required to collect these signals are not well suited to an ambulatory environment.

2.2.2.2. Smartphone Based Modalities

A list of potential smartphone modalities is provided in Table 1 below, along with descriptions of the kinds of useful information that such data can provide for emotion recognition. This information inspires the choices of features to be computed. Each sensor is also labelled with references to the studies in which they have been used previously.

Modality	Information
Screen On/Off [4, 5]	Frequency and duration of phone use First and last uses for the day (sleep indicator)
Location Services [1, 2, 4, 5, 20]	Current location Time spent at work vs home Time spent indoors vs outdoors Weather inferences Adherence to daily routine
Phone [2, 4, 5, 20]	Frequency/volume incoming/outgoing calls Most frequent contacts Call durations
Messages [2, 4, 5, 20]	Similar to phone
Emails[2, 5, 20]	Similar to phone
Social Media Usage	Messages as above Duration and frequency of use Number of posts and likes
Applications [2]	Usage of apps by category Music and Entertainment
Calendar	Number of events planned Proportion of day scheduled
Internet [2, 5]	Browsing patterns Search history
Sleep Monitoring [5]	Duration and quality of sleep
Microphone data [21]	Detection of conversations Noise levels
Accelerometer Motion [1, 4, 5]	Steps taken Activity levels - sitting/walking/running
Typing Behaviour	Typing speed Number of mistakes and backspaces used Use of special symbols and emoticons
Bluetooth[1, 20]	Proximity sensing of other people's devices

Table 1 - Summary of Smartphone Modalities

2.2.3. Collection Methods

Relevant studies have used a range of different devices for collecting both physiological signals and smartphone data. We present a summary of previously used and potential devices for collecting physiological signals in Table 2.

Device	Notes
Empatica E4 [22]	Complete set of most common sensors Easy to interface with good APIs - accessible raw data Bluetooth connectivity
Affectiva Q Sensor [4, 5]	Discontinued
Jawbone	Successor to Affectiva Q Sensor Consumer targeted, no raw data access
BodyMedia Sensewear [9]	Discontinued
Biopac System [11]	Requires electrodes and patches
Actigraph [22]	Accelerometer only Can add external bluetooth HR monitor
Generic HR Monitors	Potential limited access to raw data

Table 2 - Collection devices for physiological signals

While it is certainly possible to access smartphone data directly using the native operating system's APIs (i.e. android or iOS APIs), there are also a few pre-existing libraries that have been created to streamline this process. Some libraries that have been used in previous work are listed in Table 3.

Software Library	Notes
EmotionSense [1, 23]	Android based Good range of sensors Open source and easily extensible Outdated and not compatible with current android versions
Funf Open Sensing [5]	Similar to emotion sense Less sensors
MyExperience [1]	Not specifically emotion targeted Only for windows phones

Table 3 - Libraries for Smartphone Data Collection

Another aspect of the collection methodology that is worth considering is the choice of participants for the study. Some studies focus only on one participant and build a personalised model, while others use multiple participants. In the latter case further consideration must be given to the age,

gender, and cultural background of participants [3]. The length of time over which data is collected is also important, and whether or not collection is continuous or segmented.

2.3. Feature Extraction

Features are the meaningful values that we are able to extract by analysing raw data and their nature will vary depending on the form of this data. Feature extraction is simply the process of computing these values, but depending on the quality of the data or signal this can include further processes such as noise reduction [5].

Data that comes in the form of discrete records and logs such as text message metadata might be analysed to extract features such as number of messages received, number of messages sent, number of people contacted, or even percentage of people contacted who are in the most frequent contacts list, to give a few examples [4-6].

On the other hand, data that comes in the form of continuous signals such as skin conductance can be analysed by statistical means to extract features. Common statistics include mean, minimum, maximum, variance, range, standard deviation, area under the curve, multi-bin histograms and other forms of frequency analysis [8, 9, 16]. These statistics can also be extracted from normalised, differentiated, and first or second difference versions of the raw signals [4].

In a study done by MIT's Affective Computing lab [4] features were evaluated based on maximising information gain, which is equivalent to minimizing entropy, and this can be seen as the reduction in uncertainty about one variable after observing another [24].

Entropy is expressed in the following equation, where $P(x_i)$ is the probability of x_i being the value of random variable X:

$$H(X) = -\sum_{i} P(x_i)\log P(x_i)$$

The information gain can then be expressed as the change in entropy for X resulting from the observation of Y:

$$I(X,Y) = H(x) - H(X|Y)$$

By using this method, the study was able to identify which features provided the highest information gain. For phone base features the most successful features appeared to be screen state related (such as mean duration on, and total number of screen on events) and mobility features indicating how much time the user spent outdoors. For physiological features, it was shown that various skin conductance statistics (such as median, and standard deviation), as well as accelerometer statistics

(standard deviation), provided the highest information gains. Similarly, they found that features related to sleep were also effective.

In another study a similar process was followed again using entropy to choose the best features for classification [5]. In this case call, SMS and screen state features were all shown to be useful phone features, included in the best feature sets. Both skin conductance and accelerometer data were again shown to be effective features.

2.4. Physiological Lab Trained Models

Much of the work done to date on physiologically based emotion recognition has been in a laboratory setting, however, a variety of annotation methods have still been used. By far the most common method used in these settings is pre-labelled stimuli as is explained in section 2.2.1. on emotion annotation methods. Generally emotions are elicited using films, images or music, however, another method that was used in one study [8] was that of visualisation where the subject, specially trained in acting and visualisation techniques, was asked to sit in a quiet room and try to experience specific emotions with the help of personally chosen stimuli. Some studies have also made use of self-assessments by asking subjects to complete surveys after presenting them with stimuli [11].

Most studies opt for films as an emotion elicitation method following the work done by Gross and Levenson in 1995 [25] which suggested that films were the most emotionally engaging medium. However, a more recent study done specifically comparing films and images [26] suggests that pictures were actually more effective at producing a corresponding emotional state.

Even once an emotional stimulus type is chosen, the way in which this stimulus is presented must also be carefully considered. The order of presentation of stimuli is important because there is evidence to show that particularly arousing or disturbing stimuli can affect the emotions induced by subsequent stimuli [8]. As such studies adopt a method where stimuli are presented in groups of progressively increasing arousal level. These groups of stimuli are often separated by some sort of emotion normalising process [8]. As well as these considerations another study [16], adopting the Clynes protocol, engages physical expression by asking the user to press a button in an expressive way. This process provides what is called somatosensory feedback to the user and can strengthen their emotional response.

2.5. Classification

Classification approaches are widely varied in relevant studies, but it is not particularly meaningful to compare their performance across studies due to the varied nature of data collection methods that

these approaches are applied to and then the various combinations of feature selection and machine learning algorithms. None the less it is useful to summarise the approaches out there and to look at those which are frequently shown to be successful.

2.5.1. Feature Selection

A summary of the feature selection and transformation approaches used in the relevant literature is presented in Table 4. Of these the most commonly used types of feature selection seem to be Sequential Forward Selection (SFS) and Sequential Forward Floating Selection (SFFS), both very similar algorithms with SFFS just adding an extra "floating" step at each iteration (where it is checked if one of the currently selected features can be removed to increase performance).

Feature Selection	**Notes**
Sequential Forward Selection [2, 5, 12]	Most frequently used, very simple
Sequential Backward Selection	Very similar to SFS
ANOVA [7]	Analysis of variance method
Fisher Projection [12, 16]	Feature transformation rather than selection
Sequential Floating Forward Search [16]	Similar to SFS but at each step a step of SBS is also run
Principal Component Analysis [6, 12]	Doesn't consider class information

Table 4 - Feature selection approaches

2.5.2. Machine Learning Algorithms

A summary of frequently used machine learning algorithms for smartphone and physiologically based emotion recognition systems is given in Table 5. Among the most frequently used, which can be a useful indicator of an algorithm's strength, are K Nearest Neighbour (KNN) and Neural Networks, however Neural Networks do not appear to have been applied to smartphone data. Some studies have also compared different algorithms on the same data sets, and Support Vector Machines (SVMs) appear to have performed well against other classifiers in these cases.

Many of these algorithms also require hyper parameters to be provided when a classifier is created. The hyper parameters will obviously vary depending on the model but are often related to the algorithms learning rate, or penalty weightings to be assigned to incorrect classifications by the model (such as the SVMs C parameter).

Algorithm	Notes
K Nearest Neighbour [4, 9, 12, 16]	Most common, and very simple
Logistic Regression [2, 21]	Output can be continuous, weights meaningful
Neural Networks [4, 9, 11, 12]	Common, little information about features importance
Support Vector Machines [4, 5]	Shown to perform well against other methods
Random Forests [4]	Shown to perform well against other methods
Naïve Bayes [4, 16]	Simple, easy to add further evidence later

Table 5 - Machine learning algorithms

2.5.3. Validation Methods

The three common validation methods that are used in relevant studies are K Fold Cross Validation (KFCV), Leave One Out Cross Validation (LOOCV) and the use of Held Out Validation (HOV) sets. Often these methods are also used in combination [9].

The LOOCV [2, 9, 12, 16] approach is implemented by removing one training sample at a time from the training set, and training the model with the other remaining samples. The model is then tested with the single training sample that was left out, and its predicted value compared to the known value. This is repeated for each training sample in the set until each has had a turn at being 'left out', and the results are averaged. Because labelled emotion based training data is often costly to collect this approach is very popular because it optimises the size of the training set [3]. This approach does have issues, however, as the models trained at each iteration overlap significantly (as they differ by only one sample).

To ensure that there is no overlap between training and validation sets we can use an HOV set [4, 9]. In this scenario, a subsection of the data will be put aside (the validation set) for final testing and the remaining data (the training set) will be used to optimise parameters and train the model. Once parameters are decided and the model is trained, the HOV set is used to evaluate the classifier's performance. While this ensures that the training and validation sets have no overlap it reduces the size of the training set substantially.

KFCV [5, 21] works similarly to LOOCV but instead of removing individual samples, removes partitions of the data set. The first step is to split the data into K folds, and then a model is trained on K-1 of the folds and tested on the one left out fold. This approach is almost a trade-off between the previous two approaches causing less overlap between training sets than LOOCV but decreasing training set size by less than the held-out method.

2.5.4. Evaluation Methods

Many papers use accuracy as a scoring method for optimising and evaluating the performance of classifiers. When the problem involves normal multi-class classification then this is quite appropriate, however, when the scale that the study is classifying on has a natural ordering (i.e. is ordinal) then this measure does not tell the whole story [27]. Accuracy only tells us how many predictions were made exactly right, it doesn't take into account how far from the correct prediction any incorrect predictions were. For example, if the true value of a sample on a 5-point scale is 4, then a classifier which predicts 5 is much closer than another classifier which predicts 1. To account for this, two common scoring methods can be used. The first is Mean Absolute Error (MAE), which is the mean of the absolute value of the difference between true and predicted values and the second is Mean Squared Error (MSE), which is the mean of the square of the difference between true and predicted values [27, 28]. Of these two measures MSE places the greatest performance penalty on predictions that are very wrong.

Whether accuracy, MAE, or MSE is used as a scoring method, the score alone does not provide enough information about the performance of a classifier. A comparison should also be made with a trivial classifier as a benchmark [28]. Various trivial classifiers exist such as the Majority Class Classifier, Average Class Classifier, or classifiers that use ratings assigned to preceding samples [2].

3. System Design

This chapter will begin by revisiting the aims of the study, as outlined in section 1.3, in order to justify the design goals that the system has been designed to achieve. Following on from this is an introduction of the general structure of the solution in the system overview, in order to emphasise the key components of the system and how they fit together before going on to explain each of the components in more detail individually. For each component, we will explain the requirements, inputs, outputs and the internal structure. The final sections of this chapter highlight the benefits and limitations of the system's design.

3.1. Design goals

This study aims to investigate the effectiveness of a range of data annotation methods in real world unconstrained settings. Clearly this involves collecting and annotating data in a variety of ways, but in order to investigate the effectiveness of these methods experimentally this study will also require a full machine learning pipeline. As previously discussed this pipeline will need to be comprised of 4 stages – collection, feature extraction, lab training, and classification - as can be seen in Figure 1 in section 1.4.

As a whole the design goal is to design each of these stages to interface cleanly with each other so that when combined the system is able to achieve its aims of investigating data annotation techniques. To best investigate these data annotation techniques one of the key design goals will be to maximise the variety of approaches taken, data collected, features computed, and annotation schemes applied in order to create a wide range of scenarios in which we can test our hypotheses. Each stage has its own specific associated design goals and requirements which will be explored in more detail in the component's section.

3.2. System Overview

A basic overview of the system was first presented in Figure 1 of section 1.4, however, many details were omitted for the sake of simplicity in initial explanations. Having completed a review of the relevant background and theory it should now be appropriate to present a more detailed system overview as can be seen in Figure 3.

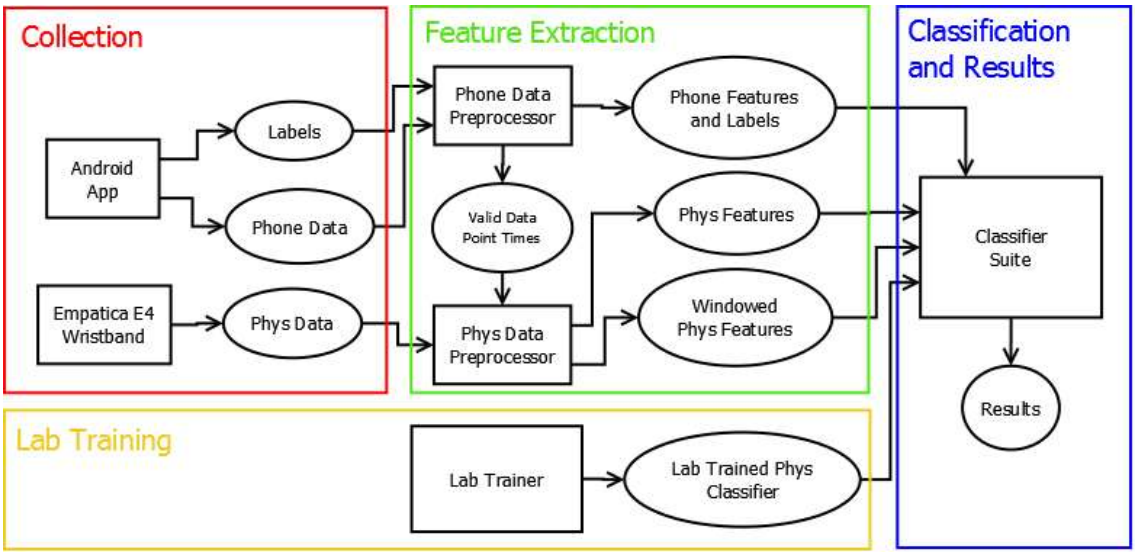

Figure 3 – Detailed System Overview Block Diagram

As has been explained, the system comprises of 4 stages or modules each with a relatively unique purpose. Each module is itself comprised of smaller sub components which carry out related tasks. The first stage in the system is clearly collection. The collection module is responsible for recording both phone and physiological data as well as asking the user to provide emotion labels in variety of forms to be attributed to this data. These data and labels are then downloaded from their respective devices and loaded into the second stage which we call the feature extraction module. After reading in all the labels provided to it (whether they be ratings or reviews) the feature extraction module associates all the phone and physiological data with its correct label. Once this association has been made the data is used to compute various features which are then passed to the classification module.

Parallel to this we have the lab training module which makes use of pre-labelled emotional stimuli to induce emotions in the user. While lab training is being run the user's physiological signals are recorded. Thus the labels associated with each of the presented stimuli are then applied to the corresponding segment of physiological data, and this data/label pairing is used to train a classifier model (after feature extraction of course).

Once this model is trained it is passed to the final classification module as well. The final classification module takes the various types of features and labels and combines these to create a range of different scenarios (e.g. one scenario could be phone data only with reviews as labels, another scenario could be physiological data with combined ratings and reviews as labels).

For each scenario the classification module creates a classifier whose settings and hyper parameters are optimised before evaluating the performance of this classifier on its associated scenario and outputting the evaluation results for analysis.

3.3. Collection Module Design

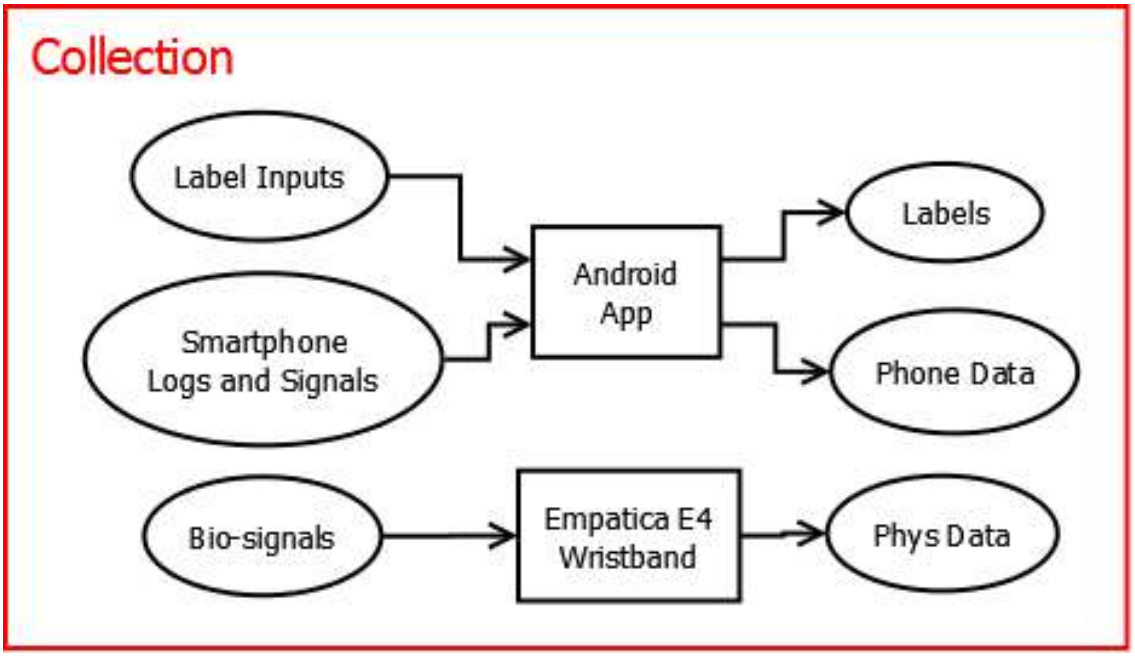

Figure 4 – Collection Module Block Diagram

The collection module is made up of two components, or collection tools. Two separate components are used so that a wider range of data types can be collected to achieve our primary design goal. The first component is a smartphone application which has two functions. The first function is to collect smartphone data in the background which can be used in the next module to create useful features. The second function is the collection of annotation data, denoted in the diagram as label inputs which is also used by the following feature selection module. The second component of this module is a wearable device which is able to collect a range of physiological bio-signals. Some small sections of this physiological (phys) data will also be input to the lab training module for the periods in time where lab training occurred. These two data types were chosen for their ability to capture continuous data in real world unconstrained contexts in a way that is mostly unobtrusive.

3.3.1. Requirements

The two primary goals of the collection system are first to collect data, and second to collect labels to associate with this data. Each of these goals can be broken down further.

Data should be collected continuously and reliably in a way that user error is unlikely to result in the erasure, corruption or non-recording of data. This data collection process should occur silently in the background such that it is as unobtrusive to the user as possible. A wide variety of data should be collected so as to best capture indicators of any emotion that may be experienced, and also to allow for the investigation of relationships between different annotation types and specific types of data. A large enough volume of data should be collected to allow for effective prediction, and to avoid issues such as overfitting.

Emotion labels similarly need to be collected continuously and reliably. Labels of both 'on the spot' rating and retrospective review style should be collected for the same data as this will allow for the fairest comparison of the effectiveness of each method. Both reviews and ratings should be convenient for the user and create the smallest interruption possible so as to still be deemed unobtrusive. At the same time, it is desirable to have annotations as frequently as is practical, in order to maximise the time scale granularity. The system should also have methods in place to ensure that annotations are made regularly and not forgotten.

Finally, both data and labels must be easily accessible for transfer to a PC for analysis. Neither collection process should be a drain on system resources such as battery life, CPU time, or mobile data allowances, again for the sake of being as unobtrusive as possible.

3.3.2. Inputs

As seen in Figure 4 there are 3 inputs (or groups of inputs) to this module.

- Label inputs – This is the annotation data provided by the user. This is collected through the app in the form of on the spot ratings and also end of day reviews. The user is prompted to submit these by the app.
- Smartphone logs and signals – Smartphone logs are records kept by the phone's operating system about phone activity such as calls, messages and app usage. Smartphone signals refer to the notifying signals sent by the operating system when specific events occur such as the screen being turned on or the phone becoming connected to WIFI. These are both collected by the smartphone app.
- Bio-signals – A range of bodily signals to try and capture the physiological response to emotions that are being experienced. These signals include indicators such as how much the user is sweating, their skin temperature, their movement and heart activity.

3.3.3. Outputs

As seen in Figure 4 there are 3 outputs (or groups of outputs) for this module.

- Labels – The annotation data provided by the user is recorded along with a timestamp of exactly when the annotation was provided. Ratings and reviews are saved in separate files.
- Smartphone data – Data from the various smartphone sensors is saved in separate files e.g. a file for call data, a file for app usage data and so on. This data is in a categorical or numerical format depending on the data type. Each piece of data is also labelled with the correct timestamp.
- Physiological (Phys) data – Physiological signals are sampled at varying rates and the samples are saved into a separate file for each physiological signal type e.g. a skin temperature file, and a skin conductance file. Each file is labelled with a timestamp signifying the beginning of collection, and a sampling rate.

3.4. Feature Extraction Module Design

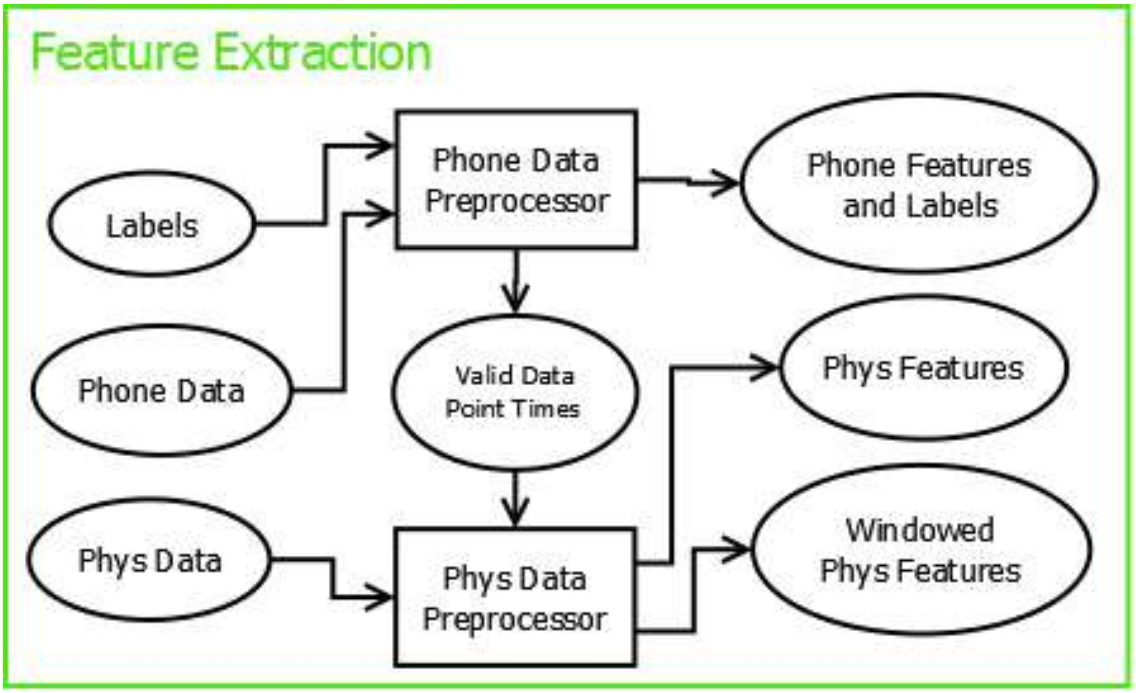

Figure 5 – Feature Extraction Module Block Diagram

The feature extraction module is comprised of two subcomponents. The reason for separating these components will become apparent in the implementation section and is simply a result of the different data types being more suited to being processed in different programming languages. The labels and phone data produced by the collection module are taken in by the phone data pre-

processor which first reads in each label and creates a data point to associate with it. Following this each phone data file is read in and the relevant segments of data are then stored with their corresponding label in the data points created in the previous step. Once all data has been read in, features are computed to create a phone based feature vector for each data point. These features are then output to a file, along with their emotion label, which is passed to the subsequent classification module.

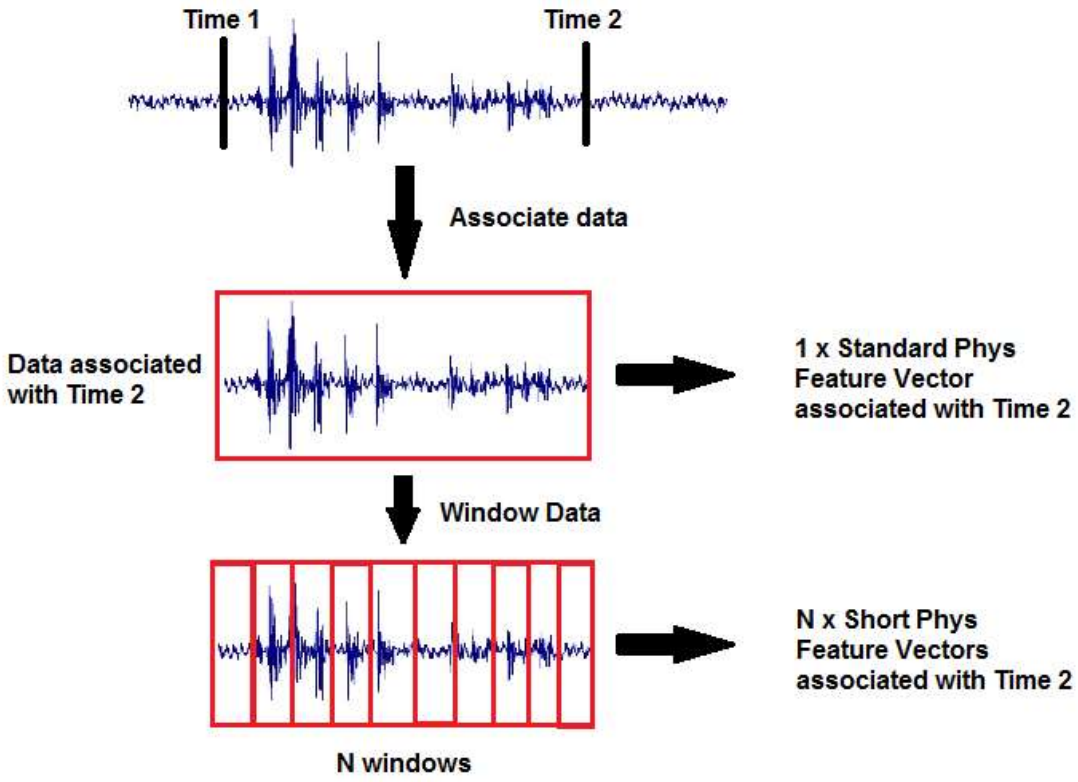

Figure 6 – Windowed Physiological Features

The phone data pre-processing component also outputs another file which is used by the physiological data pre-processor. This file contains a list of the timestamps of all the valid data points that were created by the phone data pre-processor. The reason this file is passed, instead of simply inputting the labels directly is that some data points may contain corrupted, missing or erroneous data which is removed by the phone data pre-processor and we want to ensure that the data points computed by the physiological pre-processor match those of the phone, to allow the two sets of feature vectors to be combined if desired.

After reading in the valid data point timestamps the physiological pre-processor then reads in the physiological data provided by the collection module and associates the correct segments of this data with each data point. Features are then calculated and output in a set of feature vectors that is the same size, and corresponds to the set of phone features. As well as this the physiological data pre-processor also computes another (much larger) set of features. Each data point (which in general will cover a period of time of around 1 to 3 hours) is then split into 10s windows and the features associated with that window are computed and output, along with a label denoting which larger time period that window is associated with. Figure 6 above explains this process graphically.

The reason for doing this is to provide features that are appropriate for use as input to the lab trained classifier that we explore in the next section 3.5. As can be seen in Figure 6 the first step is to split up the raw data by associating it with the timestamps of the provided labels. In the diagram the segment of data in the red box has been associated with time 2. From this data, the standard physiological features for the label corresponding to time 2 can be calculated and these are output in one file. Then this segment of data is again split into n small 10s time windows and each of these small time windows has its physiological features computed. As a result there are now n feature vectors of small time periods all of which are associated with time 2, and these are output in a separate file to the standard feature vectors.

3.4.1. Requirements

The main goal of the feature extraction stage is to take in the various forms of collected data and labels and compute features to associate with each label. After taking in all labels this module will need to correctly associate the segments of data in time to each label, and be able to do this for a variety of annotation schemes. For each of these segments a good range of features should be calculated which capture any aspects of the data which might give insight into the user's emotional state. These features must then be output in a format that can be read by the classifier stage. The module will also need to gracefully handle corrupted, erroneous or missing data. Finally, the feature extraction stage will also need to output extra feature vectors for some physiological signals which are appropriate to use as input to the physiological lab-trained model. All this should be done in a way that is easily extensible to allow for the addition of new data types if desired.

3.4.2. Inputs

As seen in Figure 5 there are 3 inputs (or groups of inputs) to this module. Each of these inputs is received from the collection module.

- Labels – The annotation data provided by the user. Ratings and reviews are saved in separate files, and each is labelled with a timestamp.

- Smartphone data – Data from the various smartphone sensors is saved in separate files. Each piece of data is also labelled with the correct timestamp.
- Physiological (Phys) data – Physiological signals are sampled at varying rates and the samples are saved into a separate file for each physiological signal type. Each file contains a header with a timestamp signifying the beginning of collection, and a sampling rate.

3.4.3. Outputs

As seen in Figure 5 there are 3 outputs from this module, each of which is passed to the classification module.

- Phone features and labels – Each data point (corresponding to a label) has all its features output on one line with the final entry being the label itself.
- Physiological features – Each data point (corresponding to a label) has all its features output on one line. This should be the same number of lines as the phone features file.
- Windowed physiological features – For each data point in the physiological features file there will be many windowed physiological feature vectors. Each vector is output on its own line with the first element being the label time with which that window is associated.

3.5. Lab Training Module Design

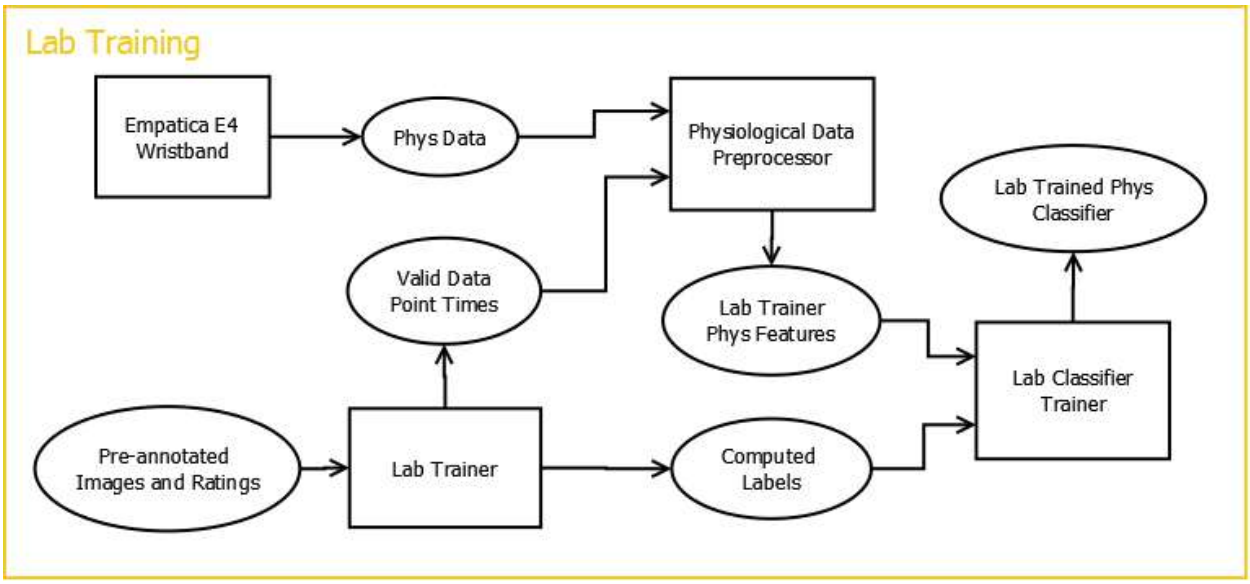

Figure 7 – Lab Training Module Block Diagram

Previously to this section the lab training module has been represented by a single block to avoid over complicating the overall design diagram. Here the details of its construction are presented as can be seen in Figure 7 above. The module is comprised of 3 main components but also receives

input (in the form of physiological data) from the Empatica wristband, one of the components of the collection module as seen in section 3.3.

The first of these components is the Lab Trainer itself. This program takes in a set of images along with their predetermined emotional ratings and then displays these for a length of time to the user. The time and duration over which each image is displayed is recorded and output as a set of valid data point times. This Lab Trainer component also takes the predetermined emotional ratings provided by the database and computes an equivalent annotation label value on the same scale as that which has been used in the overall system. These computed labels are then output (one for each valid data point time) to the Lab Classifier Trainer.

Before the classifier can be trained it is still necessary to compute features from the collected data. This is done by the second component in this module, the Physiological Data Pre-processor, using very similar methods to those of the feature extraction section. Physiological data is taken in from the collection module and the relevant segments are attributed to each valid data point provided by the Lab Trainer. The output of this component is a set of physiological feature vectors which correspond to the computed labels output by the Lab Trainer.

The final component, the Lab Classifier Trainer, takes in both the features and labels provided by the two other components and uses these to optimise and train a classifier which can then be used to classify other physiological feature vectors representing small time periods. This classifier model is then saved and passed to the final classification module.

3.5.1. Requirements

The lab training stage's primary goal is to produce a trained model which can be saved and passed to the classification stage to then be used to enhance predictions. This module must provide the user with stimuli to induce an emotional response which can then be recorded and obtained from the collection stage. This should be done in an automatic and user friendly manner. When a stimulus is provided, this module must also make sure to record which stimulus was provided and the time and duration it was provided for so as to be able to associate this with the corresponding time segment in the physiological data. The stimulus must be displayed in a way that allows it to be as effective as possible at inducing the intended emotion, but at the same time stimuli must not cause unreasonable distress for the user.

Once data is collected features must be extracted keeping in mind similar design goals to those in the previous section. As well as this, appropriate emotion ratings must be calculated from the provided database scores for each image. These should be on the same scale as is used in normal data collection for users reporting their emotional state. Finally, once these features and labels are obtained a model should be trained which can then be stored and provided to the classifier stage.

3.5.2. Inputs

As can be seen in Figure 7 there are 2 inputs to this module.

- Pre-annotated images and ratings – These images are sourced from an affective picture system which comes in the form of a set of image files and an accompanying ratings file containing details of the images.
- Physiological data – The physiological data required by this module is a very small subset of the data collected by the collection module. The data for each physiological signal is stored in its own file. The lab training module takes all the same input signals as the main feature extraction module except for accelerometer data which is ignored due to the assumption that the subject is stationary while using the lab training system.

3.5.3. Outputs

As can be seen in Figure 7 there is only one output from this module, which is passed to the classifier module.

- Lab Trained Physiological Classifier – This classifier is trained and optimised using the data collected in this module and as such is most relevant to physiological features computed over a shorter time period (like those which were used to train it).

3.6. Classification Module Design

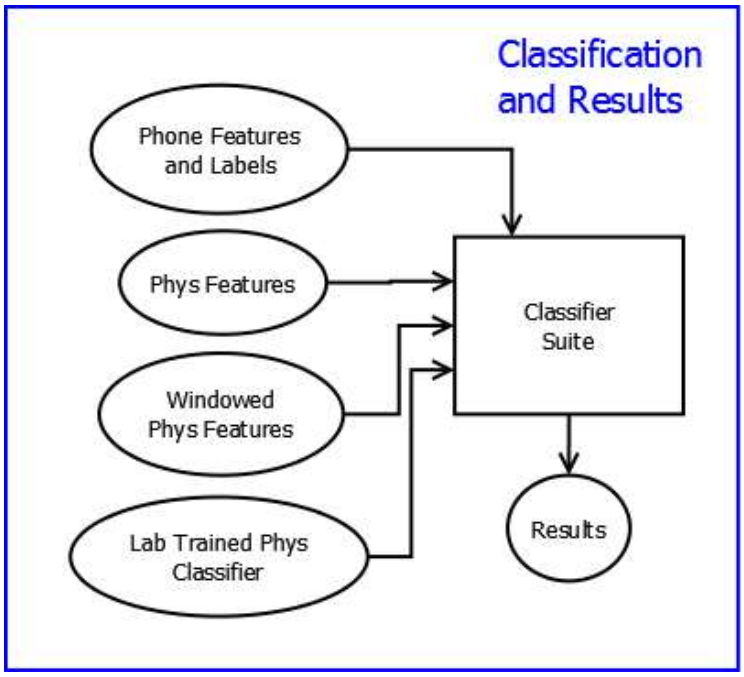

Figure 8 – Classification Module Block Diagram

The Classification Module consists of one core component, the classifier suite. This component receives inputs from both the feature extraction and the lab training modules. As a first step the classifier suite takes in the multiple different types of labels and features and generates a range of scenarios. These scenarios are characterised by different combinations of these feature and label types. Examples of different scenarios could be one with only phone features and only end of day reviews as labels, or another could contain only physiological features but with on the spot ratings as labels. This module also takes in the large set of windowed physiological features created by the feature extraction module and inputs each of these into the lab trained physiological classifier it receives as input from the lab training module. The lab classifier prediction results for all the windowed feature vectors which are associated with the same point in time are then combined, and from these results extra features are created.

A whole new set of features can then be created by concatenating these extra features and the standard physiological feature vectors. This new set can then be used to generate even more scenarios. Once this range of scenarios has been generated, a classifier can be created for each one which can be then trained on its associated data and labels. Each classifier's performance is then

evaluated by comparing with a trivial classifier on the same data set over a range of metrics. The results are printed out in a readable form and saved in a numerical format for ease of analysis.

3.6.1. Requirements

Once the system has created features of various types, and applied various annotation schemes to them the goal is to use these combinations to create a range of different scenarios. These scenarios should also apply more than one type of machine learning algorithm to create greater breadth in the types of scenarios created. For each scenario that is created a classifier should be built, and its hyper parameters and settings (such as feature selection/transformation) tuned to maximise its effectiveness at predicting the associated data. There will be many different scenarios so it should be possible to generate them, train optimal classifiers and evaluate the performance of each in a way that is automatic. The module should also provide the performance of a trivial classifier for each scenario against which performance of the trained classifier can be compared. Evaluation should also provide a range of metrics and scores for each classifier which are saved in both human readable and numerical formats.

3.6.2. Inputs

As can be seen in Figure 8 there are 4 inputs (or groups of inputs) to this module.

- Phone features and labels – The phone features computed by the feature extraction module and their associated labels of various types.
- Physiological features – The physiological features computed by the feature extraction module corresponding to the same labels as the phone features.
- Windowed Physiological features – A large set of physiological feature vectors that are computed over small windows, many of which correspond to the same label.
- Lab Trained Physiological Classifier – A classifier model which is trained on physiological responses to emotional stimuli in a lab setting. The features used to train this classifier are computed over small periods of time and thus it is only really appropriate for classification on similarly short time periods.

3.6.3. Outputs

As can be seen in Figure 8 there is only one output from this module.

- Results – Various performance metrics for the classifier in each scenario are output alongside metrics computed for a trivial classifier on the same data.

3.7. Design Benefits

The system which has been designed accomplishes a wide variety of tasks and as such it is beneficial that the overall design is quite modular. This allows alterations to be made to individual modules relatively easily without needing to make extensive alterations at all stages of the system. Adding new features, data types, annotation methods, or even classifier types, is thus relatively simple.

Another benefit of the system is the diversity of data, features, and annotation types that are used. This is made possible by the modular design which makes it easy to add new aspects to the system. It is also added to by the range of techniques applied not only in collection and feature extraction but also in the creation of extra feature types from the lab training module. Due to this diversity, the system is better equipped to meet the aims of the study by investigating the different annotation types in a range of scenarios.

The diversity of data, however, does mean that there are a lot of different scenarios that not only need to be generated but also tested. It is beneficial then that the system has been designed to create these combinations, optimise the relevant parameters and evaluate the performance automatically. This comes back to the flexibility described above, and means that adding new variations to the system is not discouraged by a difficult process of integration.

3.8. Design Limitations

While the increased diversity of the system is definitely a benefit, to accommodate for this there have been some extra inputs and outputs added. When these extra signals are within a bound of a module this adds to the module's complexity, and when they go between modules this decreases the overall modularity of the system.

The current design of the overall system could not be run in real time. While alterations to allow this would certainly be possible, at present the modules have been designed to operate on data in batches. Seen as the aims of the study are to evaluate annotation methods, a real-time system is not really necessary and would add unnecessary challenges. Perhaps if additional annotation methods were implemented, such as a scheme where the user was only asked for an input when the classifier had low confidence in its predictions, then this could become necessary.

Similarly, the system does not provide any feedback to the user or make any predictions that are not purely for the sake of evaluation. While this does not limit the study in terms of addressing its aims, subjects may be more inclined to take part in a study where they feel like they are at least getting

something out of their participation such as some sort of mood journal, or the novelty of seeing how well the system is able to predict their emotional state.

4. System Implementation

Now that the overall design and requirements of the system have been established this section will present the details of the implementation. To achieve this, the descriptions of implementation are grouped into the modules that were introduced in the section above.

First we will look at the collection module where physiological data was collected using a multi-sensor wristband along with smartphone data and emotion labels using an android application. Then we shift our attention to the feature extraction module to look at how the raw data was processed and which features were created. Following this we explore the implementation of the lab training module which pre-trains a classifier model in parallel to the first two modules. Finally, we explore the classification module to look at how different classifier scenarios were generated, how classifiers were trained, and then of course how they were evaluated. As the various design choices are presented, the reasons behind these decisions and how they help to achieve our aims of investigating data annotation techniques in a real world unconstrained context will also be explained.

4.1. Collection Module Implementation

As was outlined in the design section, the collection module is made up of two important components: a wearable device for collecting physiological signals and an android app for collecting smartphone and annotation data. However, before giving a summary of the work done on these two components we will first explain the emotional scale upon which ratings will be made and make a note on the participants used in the study.

4.1.1. Emotional Model

As explained in section 2.1 when it comes to emotional models there are generally two options. The first is the discrete model where emotions are classified into one class in a set. The seconds is the circumplex model which is a 2-dimensional continuous model where emotions are classified according to both a valence and an arousal score. This study has used a simplified version of the circumplex model by investigating valence. Essentially the valence only model projects the circumplex model into one dimension. Therefore when users are asked to rate how they are feeling they provided a valence rating on a scale of 1 to 5.

The primary reason for choosing only one axis was to make ratings as easy as possible for a user. Less scales to report on (i.e. just valence and ignoring arousal) means that ratings are quicker and easier to submit and create less of a distraction. The reason that valence was chosen as the axis (as oppose to arousal) was firstly because it is a less ambiguous concept for users but secondly because

as is stated in [10] “many investigators consider the valence of emotions to be the most important dimension of affective experience”.

4.1.2. Participants

It is worth noting that data collected in this study comes from a single participant. While data from more participants is desired it was not feasible within the scope of the project and with the time and resources available. It could also be important to note that the participant was also the researcher, an Australian male university student in his early 20s. To ensure that a large enough volume of data was collected, data collection was carried out over a continuous period of a month (with some days omitted when collection devices were out of power, or when the devices could not be used).

4.1.3. Collection of Physiological Signals

The device chosen for collection of physiological data was the Empatica E4 wristband. This device was chosen primarily because of its complete and varied set of sensors that all come bundled together in the one device. The 4 sensors which record skin conductance, skin temperature, blood volume pulse and 3 axis accelerometer data cover all the most commonly used modalities in the relevant literature. The device is also coupled with a very user friendly software package.

Downloading data off the device is as simple as connecting the device to a PC via USB and once downloaded all the data is all stored in an online account which allows the researcher to view sessions in graphical forms and also to download raw data in readily accessible CSV formats for analysis. Furthermore, the device has good battery life of up to 72 hours and a small form, both factors which make the device less obtrusive to the user who has to wear it. An example file header can be seen below for a file containing the skin conductance data.

```
1475911978.000000
4.000000
0.000000
0.052519
0.081980
... continued ...
```

The first line of the header is the time after the epoch in milliseconds when the session was started, the second line contains the sample rate and then subsequent lines contain data values.

4.1.3.1. Testing Collection of Physiological Signals

In order to test that physiological signals were being collected as expected and to see how the wristband behaved under non-standard circumstances a manual test plan was developed. A range of activities such as normal use, intense exercise, sleeping, removing the wristband and leaving it on a Table, or removing the wrist band and leaving it in a bag were all completed and logged manually in

a work book. The resulting signals were then inspected and examined to get an idea of how signals looked in the non-standard contexts, compared to normal use, so that these could be detected in the subsequent feature extraction module. This was also a useful way of checking that the physiological data collection system was working as expected.

4.1.4. Smartphone Data and Annotation Collection

The implementation of smartphone data and annotation collection was by far the most substantial volume of work for the collection module (and possibly the entire study). This was done using a smartphone application as this was seen to be the simplest and least obtrusive method for collecting this information. Almost all users have smartphones so it means that further hardware is not required. Most users also take their smartphone with them everywhere so this solution also makes it unlikely that a user will forget or be unable to make annotations so long as they are prompted. It was required to create a new application because in order to compare the effectiveness of annotation types fairly the different annotations must be collected simultaneously for the same set of raw data. No study has ever collected both types of data before and so no pre-existing application was available. What's more there was no application already available for the continuous collection of the desired set of smartphone features. There were however libraries available to help, and the one which was chosen to assist in this study was the EmotionSense library, the use of which is explained in more detail in the following sections. The android operating system was chosen as the platform to develop for, firstly because it is supported by more devices (which would be desirable if more participants are used in future work) but secondly because the provided system APIs provide a much wider range of functionalities and access than other common smartphone OSes like iOS.

To explain the implementation of the app this section will first provide a structural overview and then detail the way in which smartphone data is collected. Following this we explain some of the decisions relating to the appearance of the application, before going on to describe the details involved in the code design for both the rating and review activities.

4.1.4.1. App Overview

The app for collecting smartphone and annotation data is comprised of 2 main modules as can be seen in Figure 9 below. The first module on the left, made up of 4 components, is responsible for collecting the various types of smartphone data and runs entirely in the background, never disturbing the user. The second module on the right, made up of 6 components, is responsible for collecting the annotation data and as such requires an interface with the user. While the main activity is included in the annotation module, due to the fact that it has a UI associated with it which acts as the main menu, it also sets and runs the services required to collect other data. The code for all these components is made available through the link provided in Appendix A.

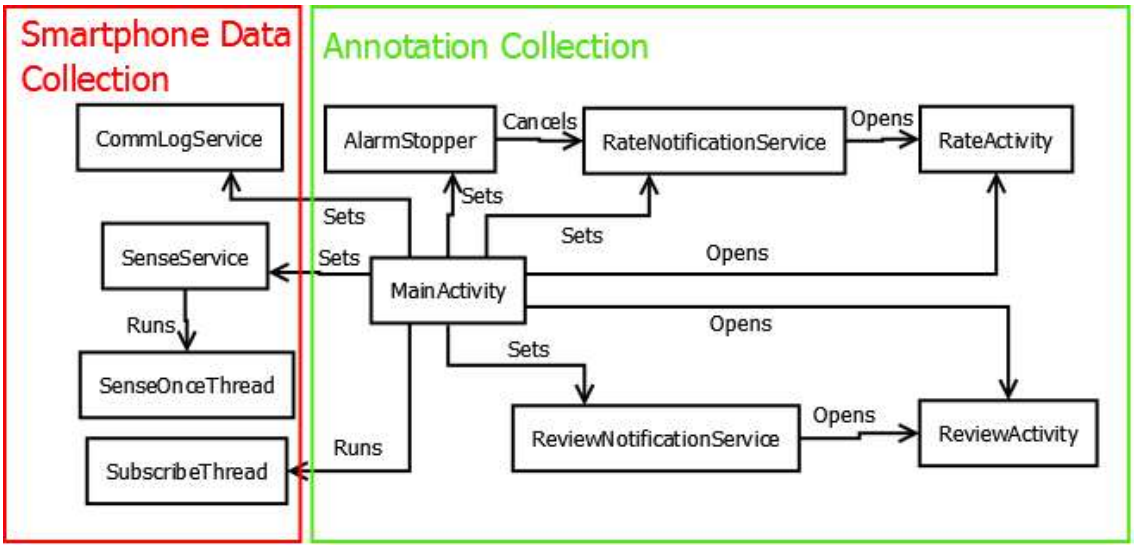

Figure 9 – Android Application Components

4.1.4.2. Smartphone Data Collection

The smartphone data collection module is made up of 4 components, all of which run silently in the background so that the user is never disturbed. These components together collect a wide range of smartphone modalities. Those chosen for collection in this study were:

- App Usage – This provides information about what the user is doing on their phone and specifically what activities they are carrying out. This data is also used to gather information about social media usage.
- Calls – This provides information about who is being called, for how long and how often. This clearly sheds light on who the users communications which can clearly be linked to emotion.
- Connection State – This modality signals whether the user is connected to WIFI, cellular data, or neither. This is used as an indicator as to how much time the user is spending outside as WIFI is assumed to generally be available only in buildings. As previously discussed this has been shown to be a useful emotion indicator.
- GPS – This provides the users location every 2 minutes. This can also be used to identify time spent in a user's favourite places, or how far they have travelled.
- Screen On/Off – This provides all the timestamps of when the phones screen has been turned on or off. This is a great indicator of overall phone usage patterns but can also be used to estimate information such as the time the user got up and the time they went to sleep.
- SMS – Similar to call information this can provide useful data about communication patterns.

The first data collection service is the CommLogService. This service, similarly to all the other services in this app, is scheduled by the MainActivity with an alarm. The code for this is shown here and can be applied similarly to all services.

```
//Set alarm to log call, app and message communication logs
Intent comLogIntent = new Intent(getApplicationContext(),
ComLogService.class);
comLogOperation = PendingIntent.getService(getApplicationContext(),
comLogIntentCode, comLogIntent, PendingIntent.FLAG_UPDATE_CURRENT);
alarmManager.cancel(comLogOperation); // in case it is already running
alarmManager.setRepeating(AlarmManager.RTC_WAKEUP,
System.currentTimeMillis(), comLogInterval, comLogOperation);
```

As the name suggests this service is responsible for recording the data that is stored in the Android communication logs. These logs are accessed directly through the android APIs and include the UsageStatsManager (which provides app usage data), the CallLog (which provides call data), and the MessageLog (which provides message data). The CommLogService is set to run every 4 hours at which point it collects all this data from the relevant logs since the last time that it was called. The method used to access each log is slightly different, and requires knowledge of android programming, but can be seen in the code base provided at the link in the appendices. While it was initially hoped that these 3 modalities could be collected using the Emotionsense library this was not possible in the end for a number of reasons. Firstly, the Emotionsense implementation of App Usage was deprecated as it relied on an old paradigm no longer supported in recent Android versions. Outdated features was a common issue with the Emotionsense library and Secondly while the call and smartphone data collection libraries worked, they did not provide all the necessary information and were also only capable of reading in ALL calls or messages in the phones entire history at once, which was not the functionality desired in this application. All the data collected by this service is stored in a simple CSV format.

For app usage, this file gives the timestamp, the name of the app package, and whether that app was brought to the foreground (opened) or sent to the background e.g.

```
2016/10/18 13:19:55,com.facebook.katana,background
2016/10/18 13:19:55,com.google.android.googlequicksearchbox,
     foreground
2016/10/18 13:31:33,com.google.android.youtube,foreground
```

For the call data the output has the form of a timestamp, followed by the contact number, call duration and call direction (incoming, outgoing or missed) e.g. (Note: phone numbers altered)

```
2016/10/12 10:34:07,0412345678,1223,2
2016/10/12 11:52:32,0487654321,8,2
2016/10/12 11:39:55,0311112222,600,1
```

For the message data a similar format is held with the timestamp followed by the SMS direction, the number and finally the length e.g.

```
2016/10/11 23:10:10,incoming,+61412345678,85
2016/10/12 01:08:16,incoming,+61412345678,345
2016/10/12 16:16:29,incoming,+6142222333,236
```

The second data collection service is provided by the SenseService class, and this has been adapted from an EmotionSense demo application. This service is run roughly every 2 minutes thanks to an alarm similar to that of the CommLogService above. When run, this service starts a new thread by running the SenseOnceThread class which is able to poll the phone's GPS to get the user's location. This is then saved in a similar format to the above files where a timestamp is followed by a latitude and longitude e.g.

```
2016/10/03 17:41:39,-33.9178246,151.2370512
2016/10/03 17:45:28,-33.9178763,151.2370068
2016/10/03 17:48:52,-33.9177612,151.2369756
```

The final two modalities, screen on/off and connection state, are collected by the subscribe thread. This is another class which has been appropriated from an EmotionSense demo application. This thread runs continuously in the background and implements the SensorDataListener interface which allows it to pick up any events related to screen or connection state as they happen and log them accordingly in the onDataSensed() callback method. The default format for data sensed in this way is JSON and as such this format has been kept when the data is saved. The connection state data has the following form:

```
"userid":"test-user-id","deviceid":"test-device-
id","senseStartTime":"16:07:15:910 03 10 2016 +1100
GMT+11:00","senseStartTimeMillis":1475471235910,"dataType":"Co
nnectionState","connected":true,"connecting":true,"available":
true,"networkType":"WIFI","roaming":"NOT_ROAMING","ssid":"\"AS
IO SURVEILANCE VAN\""
```

Screen state data is similar with its JSON format:

```
"userid":"test-user-id","deviceid":"test-device-
id","senseStartTime":"16:07:23:717 03 10 2016 +1100
GMT+11:00","senseStartTimeMillis":1475471243717,"dataType":"Sc
reen", "status":"SCREEN_OFF"
```

Finally, the testing of this section was quite a difficult task. As the data collected is real world data and simulations and tests cannot be generated automatically this requires the creation and implementation of manual test plans. Specific phone interactions were orchestrated such as test SMS conversations, test phone calls, and test app usages. These were noted down manually as they

occurred and then these results were compared with the results collected and stored in the data files. Similar tests were executed for all the data types. For both GPS and connection state this required leaving the lab to go and walk outside.

4.1.4.3. Annotation Data Collection

The annotation module of the app is made up of 6 components as can be seen in the block diagram in Figure 9 of section 4.1.4.1. Three of these components are android activities which essentially represent the different UI screens (and associated background implementation) of the annotation tasks performed by the app. The central activity is the MainActivity whose UI component is the app's main menu screen. This activity is also responsible for scheduling all the background services that need to be run (both for the aforementioned smartphone data collection module and the annotation module). The 3 other components (which are not activities) are all services which means that they are designed to be run in the background at a scheduled time, and as such are all scheduled by the MainActivity. The RateNotificationService and ReviewNotificationService, as the names suggest, are responsible for creating notifications which are sent to the user's phone to remind them to submit either a rating or a review. When the user taps the notification, the corresponding activity is started for the user to submit either a rating or review. The code to do this is in depth and requires android knowledge, but can be found by following the link to the code base in Appendix A. The AlarmStopper is a service which exists to stop the RateNotificationService from being scheduled after the user's chosen bed time. The two other activities, the RateActivity and the ReviewActivity, are the activities which are run when the above mentioned notifications are tapped. These activities are the components which display the UI screens with which a user can interact to submit emotion annotations, and can also be accessed from the main menu screen at any time. Both of these activities are explained in more detail in the sections 4.1.4.5 and 4.1.4.6 below.

4.1.4.4. Appearance

The general approach with the appearance of the app was to keep everything as plain and simple as possible. Obviously, this makes the app intuitive and easier to use but more importantly it means that the app is less likely to have an effect on the user's emotions subconsciously. For example, if bright and warm colours were used throughout the app then there may be a subtle bias towards more positive ratings, and the same could be said for dark or dull colours in relation to negative ratings. An example of this simple and plain design can be seen in the screen shot below in Figure 10 showing the UI component of the MainActivity.

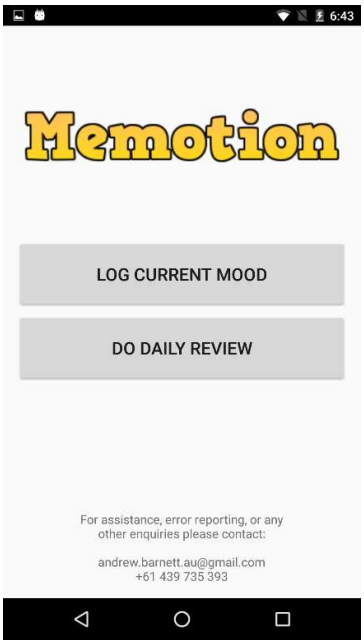

Figure 10 – MainActivity UI Component

4.1.4.5. Ratings

As explained in the overview above the ability for the user to make 'on the spot' ratings of their current emotion is provided by the RateActivity. This component can be opened either by way of a push notification, or from the main menu of the app. The UI associated with this component can be seen in Figure 11 below. Each of the 5 points on the valence scale are represented by a face taken from the universal list of "emojis". For a user to submit their current emotion they simply tap the face corresponding to how they are feeling. This allows ratings to be made easily in a matter of seconds, causing minimal interruption to the user. If the user submits their rating incorrectly then they can submit again and this will override any rating they have already made in the last 5 minutes. To ensure that the user does not forget to submit ratings a push notification is sent to the user asking them to submit a rating every 2 hours. Users may also submit ratings at other times simply by accessing the rate activity from the main menu. Ratings are saved in a csv file in rating/timestamp pairs.

Testing that ratings were recorded correctly was a simple procedure of manually entering ratings and checking that the correct ratings appeared with the correct timestamp. On the other hand, testing that the notification creation was working correctly required longer tests. For initial tests the notification was set to appear at minutely intervals, then once this was verified the interval was set to the correct length. Similarly, the AlarmStopper service had to be tested in this way to ensure that it worked correctly to stop rating notifications being generated. This was first done by setting

notifications to stop after 3 minutes and only once this smaller timescale was working was the correct time set.

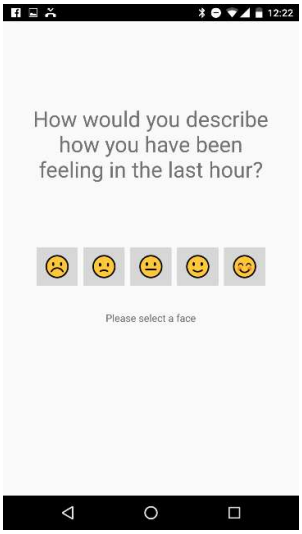

Figure 11 – RateActivity UI Component

4.1.4.6. Reviews

Sometimes users may be busy at many points throughout the day and will not have been able to submit ratings. If this is the case, reviews completed at the end of the day can be used as an annotation scheme instead or to complement the already submitted ratings. This ability is provided by the ReviewActivity. One review is requested from the user for every hour between 9am and 11pm. While the ReviewActivity may look simple, as can be seen below in Figure 12, there is a lot going on behind the scenes. For each hour of the day the review screen shows the time and date to be reviewed, the current progress that the user has made through the rating process, and the rating scale for the user to choose from. On top of this the user is also provided with information about what they were doing at the time. This information includes their location (as shown on a google maps fragment) any messages they received in the hour, any calls, and finally any apps used. The google maps fragment makes use of Google's MapView API and retrieves the GPS coordinates from those already collected by the app. All communications and app information is retrieved directly from the android system logs. This information is provided to the user to help remind them of how they were feeling at each hour throughout the day in an attempt to collect more informed and effective ratings. Once the users have viewed all the info they can select a rating for the given time period and the ReviewActivity ends by creates a new ReviewActivity for the next hour of the day. Similar to ratings, the collected review values are saved in a csv file in rating/timestamp pairs.

Testing for this module was carried out in a very similar way to the RateActivity testing as explained above, using mostly manually executed test plans, and doing initial testing by reducing the timescale over which notifications are given and data is collected.

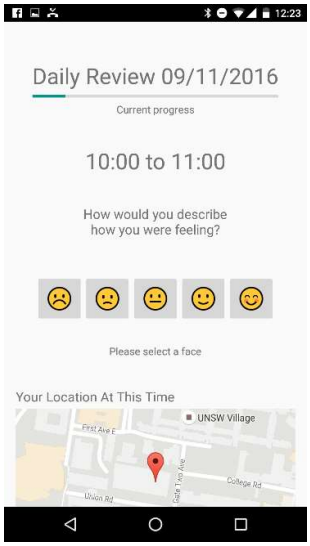

Figure 12 – ReviewActivity UI Component

4.2. Feature Extraction Module Implementation

The feature extraction module is made up of 2 subcomponents as was seen in section 3.4. The first of these subcomponents that will be explained is the smartphone feature extraction which takes in the smartphone data and collected labels to compute features and also output a list of the valid labels and their times. The second component that will be discussed is the physiological signal pre-processor which reads these valid label times in and uses them to divide the physiological data into segments. For each of these segments physiological features are calculated and output. Finally, a third program is explained, which similarly computes physiological features, but does so by first dividing the standard segments into 10s windows.

4.2.1. Smartphone Feature Extraction

This section begins with an overview of the smartphone feature extraction module and how its various classes interact. The section then proceeds to explain the role of each of the classes in more detail finishing up with a closer look at how the various forms of sensor data are represented in this data model and the features which are calculated for each.

4.2.1.1. Overview

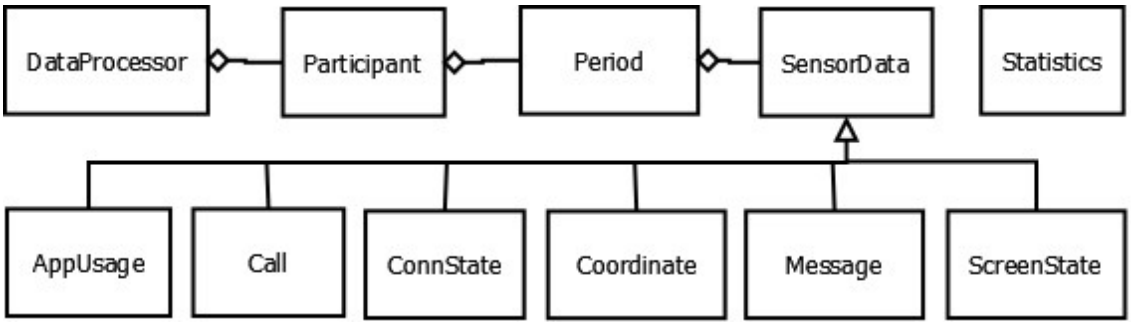

Figure 13 – Smartphone Feature Extraction UML Diagram

The smartphone feature extraction system has been implemented in the java programming language. The first reason that this language choice was made specifically for this component was due to the wide range of data types (dates, numbers, strings, etc.) that were being dealt with because java works well with a variety of types and provides all the basic data structures to store these types in convenient collections (arrays, lists, hash tables etc.) Secondly because many of these data types can be defined by a set of variables and attributes, but also still share some commonalities, an object-oriented approach is the obvious choice. The overall structure can be seen in the UML (unified modelling language) diagram in Figure 13 above. The main procedure for this program is housed in the root DataProcessor class. This class, in theory, is capable of holding a collection of Participant objects so that if there is more than one participant in the study then their data can be logically separated. The DataProcessor reads in all the data from the correct files and sends each piece of data to the correct participant. The Participant class stores a collection of Period objects each of which represents a period of time with an emotion label. All the data associated with a Participant is stored in the correct Period associated with the time at which that data was recorded. Finally, each Period maintains collections of the various types of SensorData objects that are associated with that given period in time. As can be seen in Figure 13 the SensorData class (which is actually an abstract class) is inherited by the 6 different data type classes, and these are where the individual pieces of data are stored. The final class in the diagram, the Statistics class, is essentially a collection of statistical functions and is only ever referenced statically.

4.2.1.2. Data Processor Class

The DataProcessor class is the root class of the system containing the main function. This is where all the input files are named and the class contains methods for loading all the data from these files and then outputting the computed features. This functionality is clearly demonstrated by the main function snippet below:

```
//Create a new data processor object
DataProcessor processor = new DataProcessor();

//Load the label data
processor.loadRatings(ratingsFile);
processor.loadReviews(reviewsFile);

//Load the smartphone data
processor.loadCalls(callsFile);
processor.loadMessages(messagesFile);
processor.loadApps(appFile);
processor.loadCoords(gpsFile);
processor.loadConnState(connStateFile);
processor.loadScreen(screenFile);

//Output the required data
String outputFile = "emotionFeatures.csv";
String headerFile = "emotionHeader.csv";
processor.outputData(outputFile);
processor.outputHeader(headerFile);
processor.outputRatingTimes("rateTimes.csv");
```

In the first two methods shown above both the ratings and reviews are loaded. The DataProcessor is responsible for parsing the ratings and reviews and creating a new Period object for each, which is then passed to the Participant object. The role of ordering and combining these Periods is done by the Participant class, and will be explained in the next section.

In the subsequent 6 methods smartphone data is loaded, the input is parsed and a SensorData object of the corresponding type to the data is created and then handed to the Participant object. An example of how this is done for a line of call data is demonstrated here, and the process for other data types is very similar:

```
String[] fields = record.split(",");

if(fields[0].equals("")) {
	continue;
}


String timeStamp = fields[0];
DateFormat dateFormatter = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
Date callTime = dateFormatter.parse(timeStamp);

String phoneNumber = fields[1];
int duration = Integer.parseInt(fields[2]);
String inOrOut = fields[3];

Call currentCall = new Call(callTime, duration, phoneNumber, inOrOut);

participant.addSensorData(currentCall);
```

The final 3 methods are associated with output. The outputData() method calls upon the Participant object to compute all the features for each of its Periods and these are then output to a CSV file. The

outputHeader() method is similar but instead of computing features, outputs a single line CSV with the names of every smartphone feature created. This functionality was added so that in the final classification stage of the overall system the names of selected features could be output. The final outputRatingTimes() method calls on the Participant object for a list of the times associated with each Period in its list.

In the example shown above both the ratings and reviews were loaded, however, in order to create feature sets that use only one type of data annotation it suffices to remove the call to the method which reads in the annotation type that is no longer needed. So if it was desired to only load ratings then one would simply remove the call to loadReviews().

Similarly, to change the labels to use a 3 point scale (negative, neutral, positive) the boolean "threePoint" can be set to true and the following code in the methods for loading annotation data takes care of the translation:

```
if(threePoint) {
	if(rating == 1 || rating == 2) {
		rating = 1;
	} else if(rating == 3) {
		rating = 2;
	} else {
		rating = 3;
	}
}
```

4.2.1.3. Participant Class

The Participant class is supposed to represent an individual participant in the study. Due to time and resource constraints only one participant was able to be used in this study but this was not known at the time of the feature extraction system's design. Of course, the ability to deal with multiple participants could also be useful if this program were to be used in future applications. Regardless, the Participant class is responsible for storing and maintaining a list of Period objects associated with the participant. Periods are added by the DataProcessor class using a different function depending on whether the period is based on a label provided by a review or rating. The reason for this distinction is the way in which ratings and reviews are combined in the case that both systems want to be used for annotation simultaneously. The list of Periods is maintained in ascending order at all times. In order to add a rating based Period the method addPeriod() is called which simply adds the new period into the list at the correct location to maintain sorted order. If there is another period in the list within 15 minutes of this period however, then the first of the two periods is discarded. In order to add a review based Period, however, the process is slightly more in depth. Review based periods are added with the method addReviewPeriod(), which has the precondition that all rating based Periods must have already been loaded. This is because the system gives precedence to

ratings when the two methods are combined and only uses reviews to fill in the gaps when ratings are not available. Ratings were prioritised over reviews as they were shown to have better success when used individually. The process for adding reviews is outlined in the pseudocode below:

```
Find period[i], the first period in the list which is after
the newPeriod

If period[i] after is more than 45 minutes away

      If the period[i-1], i.e. the previous period in the list
      is more than 45 minutes before

            Add the newPeriod

      Else

            newPeriod's time is set to period[i-1]'s time plus
            45 minutes

            Add the newPeriod
```

This process ensures that no reviews are added if they are within 45 minutes of a rating which was provided as more frequent ratings are not deemed necessary and this would reduce the window associated with the adjacent ratings.

Once all periods have been added, the DataProcessor class can begin to add sensor data. The method used to accomplish this is shown below as it is quite simple for the reader to understand in plain code.

```
public void addSensorData(SensorData data) {

      //Find the correct period i.e. the first period
      //in the list whose time is after this data's time
      for(int i = 0; i < periods.size(); i++) {

            if(periods.get(i).getDate().after(data.getDate())) {

                  //If the time between the data and a period is less
                  than MAX_DATA_AGE, add it
                  if(data.getDate().getTime() -
                  periods.get(i).getDate().getTime() < MAX_DATA_AGE) {
                        periods.get(i).addSensorData(data);
                  }

                  break;
            }

      }

}
```

Once all sensor data is added to the correct periods then features can be calculated with the outputFeatures() method shown below which simply calls upon each Period in the Participant's period list to return it's associated feature vector. This feature vector is then written to the output file.

```
public void outputFeatures(BufferedWriter writer) {
	Date prevPeriodDate = null;
	for(Period period: periods) {
		try {
			String featureVector =
			period.outputFeatures(prevPeriodDate);
			writer.write(featureVector);
			prevPeriodDate = period.getDate();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

}
```

Similar methods are also provided to output the times associated with each Period (to create the valid label time file that is required by the physiological feature extraction), and to output the feature names which are required for the smartphone data header.

4.2.1.4. Period Class

A Period object represents the data point associated with a label provided by either a rating or a review. This object stores the label value but also all the data of the various sensor types associated with that label. The data structures used to store the various data types are of the following form:

```
private Date time;
private int rating;
private HashMap<String, List<Call>> calls;
private HashMap<String, List<Message>> messages;
private HashMap<String, List<AppUsage>> appUsages;
private List<ConnState> connStates;
private List<Coordinate> coordinates;
private List<ScreenState> screenOns;
```

When new data is added by the Participant method as seen in the previous section, the addSensorData() method is called, a shortened version of which is shown below. Note only the implementation for app usage data is shown as an example.

```
public void addSensorData(SensorData data) {
	if(data instanceof AppUsage) {
		AppUsage currentUsage = (AppUsage) data;
		if(appUsages.containsKey(currentUsage.getName())) {

			appUsages.get(currentUsage.getName()).add(currentUsage)

		} else {
```

```
                        List<AppUsage> usagesForNewApp =
                              new ArrayList<AppUsage>();
                        usagesForNewApp.add(currentUsage);
                        appUsages.put(currentUsage.getName(), usagesForNewApp);

                  }
            } else if(data instanceof Call) {
                  ...
            } else if(data instanceof ConnState) {
                  ...
            } else if(data instanceof Coordinate) {
                  ...
            } else if(data instanceof Message) {
                  ...
            } else if(data instanceof ScreenState) {
                  ...
            } else {
                  ...
            }
      }
```

Once all data has been added in order to generate the Period's feature vector the outputFeatures() method calls the computeFeatures() method from each of the data types that it holds. Each computeFeatures() method returns a string of features which the Period's method concatenates together and returns as seen below. The hour of the day associated with this period is also added as an extra feature, and finally the rating value is provided as the last element.

```
return Call.computeFeatures(calls, periodLength) + "," +
      Message.computeFeatures(messages, periodLength) + "," +
      AppUsage.computeFeatures(appUsages, periodLength) + "," +
      ConnState.computeFeatures(connStates, periodLength) + "," +
      Coordinate.computeFeatures(coordinates, periodLength) + "," +
      ScreenState.computeFeatures(screenOns, periodLength) + "," +
      calendar.get(Calendar.HOUR_OF_DAY) + "," +
      rating + "\n";
```

A similar method also exists which calls the methods from each data type's class to return a list of the feature names as strings which are again concatenated and returned for use in the header.

4.2.1.5. Sensor Data Classes

As can be seen in the code snippets of the previous section, and Figure 13 above, there are 6 different types of sensor data. Each of these inherit from the abstract class SensorData which simply ensures that each class has a Date field and also provides a getDate() function. Aside from this the general structure of each sensor data class is still quite similar. Each SensorData class generally consists of:

- Fields to hold any relevant info about the piece of data e.g. phone number, duration, direction etc.

- A constructor to initialise these fields.
- “Get” methods for each of these fields.
- A public computeFeatures() method which can be called by the Period class.
- Private helper methods which compute groups of similar features for the computeFeatures() method.
- A public featureNames() method which returns a string of all the names of the features associated with this data type.
- Private helper methods for the featureNames() method.

The features that are computed vary between data types depending on their nature. As an example the private helper methods used by the featureNames() method for the AppUsage class are shown below. It is interesting to note that the AppData class has calculated special features relating to the use of social media apps e.g. “smDurationSum” which is the sum of the durations of time for which social media apps were used during the given period. For reference, the full list of phone features by data type are provided in Appendix B. For the calculation details of these features please see the java files provided by following the link provided in Appendix A.

```
private static String socialMediaStatsLabels() {
	return		"smDurationSum" + "," +
			"smNumUses" + "," +
			"smDurationMean" + "," +
			"smDurationSD" + "," +
			"smDurationMedian" + "," +
			"percentNumUsesSM" + "," +
			"percentDurationSM";
}

private static String appStatsLabels() {
	return		"totalNumberAppsUsed" + "," +
			"maxNumUsesOneApp";
}

private static String durationStatsLabels() {
	return		"appDurationSum" + "," +
			"appDurationMean" + "," +
			"appDurationSD" + "," +
			"appDurationMedian" + "," +
			"maxDurationOneApp" + "," +
			"percentTimeMaxApp";
}

private static String timeStatsLabels() {
	return		"overallAppTimeSD" + "," +
			"meanIndivAppTimeSDs";
}

private static String numberLabels() {
	return "totalAppUsages";
 }
```

4.2.1.6. Testing

Testing of the features computed in this program was done manually. Each SensorData class has the ability to print debug information where feature names are output along with feature values to make verification easy. A series of particularly eventful (but also uneventful) Periods were chosen and all the features were calculated with the debug option turned on. Each value was then checked against the expected to ensure that output was logical.

4.2.2. Physiological Feature Extraction

Although the physiological feature extraction component has been represented as an individual block, in reality it takes the form of two Matlab files. The first is responsible for creating standard features and the second creates the windowed feature vectors. The overall function of the standard feature extraction is explained first by walking through pseudocode in an overview. Following this we will look at the most important steps in the pseudocode in more detail, such as initial processing, matching of data with labels, and calculation of features, providing small code snippets where appropriate. Following this the windowed feature extraction module, which uses a very similar process, is explained.

4.2.1. Overview

The general functionality and structure of the Physiological Data Pre-processor is demonstrated in the following pseudocode:

```
Read in rating timestamps from the file provided by the
smartphone feature extraction program.

For each recorded session (i.e. set of data files):
     For each data file type:

          Read in the data
          Run a median filter on the data

          For each timestamp:

               If the current data file overlaps with this
               timestamp :
                    Take all the data relating to this
                    timestamp
                    and compute the associated features.
                    Store features in results matrix

Print results matrix out as CSV file
```

Running the code represented by the above pseudocode will result in a feature file being output relating to one type of annotation data, this should match the type of annotation data that was used

in the smartphone feature extraction process which created the file of valid timestamps which was taken as input.

4.2.2. Processing

Once the raw data has been read in, a small amount of processing must be carried out. If the data type is blood volume pulse then the absolute value of the signal is taken. Similarly, if the data comes from the accelerometer then each reading is made up of an x, y, and z component and we choose to combine these in quadrature to give an acceleration magnitude i.e.

```
data(i) = sqrt(accData(i,1)^2 + accData(i,2)^2 + accData(i,3)^2);
```

Then regardless of data type all raw data signals are filtered using a median filter with order 9 in order to remove any noisy spikes.

4.2.3. Attributing Samples

In order to attribute samples to the correct emotion labels, a sample index is maintained as a data file is looped through. By knowing the start time of the file and the sampling rate it is possible to use the sample index to easily compute the time that a sample was collected. The system attributes all data in the 3 hours proceeding a label to that label. The process of updating this sample index and determining the start and finish indexes for the current segment of data can be seen in the code snippet below.

```
%Keep track of which sample we are up to
sampleCount = 1;
%Go through each labelled segment
for i = 1:1:size(ratingTimes,1)

    %Only include data that is up to 3 hours before the
        ratingTime
    MAX_DATA_AGE = 3*60*60;
    while(startTime + sampleCount/sampleRate + MAX_DATA_AGE
        < ratingTimes(i))
        sampleCount = sampleCount + 1;
    end

    %The current segment starts at and includes this index
    startIndex = sampleCount;

    while(startTime + sampleCount/sampleRate <
        ratingTimes(i))
        sampleCount = sampleCount + 1;
    end

    %The current segment finishes at and includes this index
    finishIndex = sampleCount - 1;

    %Make sure that we haven't run out of samples
```

```
if(finishIndex < startIndex || startIndex >
    size(filtData,1))

    %We have no samples in this segment

else

    %Compute features
    %startIndex and finishIndex can now be used
    %to access the correct segment of the data array
    ...

end
```

4.2.4. Features

A range of features are computed for each signal so as to maximise the information that can be provided to the final classification module. The signals that are provided to the feature extraction module are all of those made available by the Empatica E4 wristband:

- Skin Temperature
- Skin Conductance
- Blood Volume Pulse
- Heart Rate
- Accelerometer Data

For each of these signals, not only are features calculated for the raw data signal but the system also transforms the raw signal to create a normalised signal and a derivative signal. The normalised signal is calculated as:

```
normData(x) = (rawData(x) - rawMean)/(rawMax - rawMin);
```

And the derivative signal (which is really an approximation) is calculated by taking the difference of adjacent elements.

```
diffData(x) = rawData(x+1) - rawData(x);
```

For these 3 signals we then calculate the same range of features. These being:

- Mean
- Min
- Max
- Variance
- 5 Histogram Features

The 5 histogram features are simply calculated by creating a histogram with 5 equally spaced bins and then calculating the percentage of values which fall into each bin. With this combination of features the system is able to capture much of the information about the distribution of values in the signal.

Finally, there are two extra features which are calculated to try to identify if the wristband was removed for any period of time during the given segment of data. These features are calculated purely by looking at skin temperature when it falls below an experimentally determined threshold (to a temperature which is highly unlikely if the device were actually being worn). It was initially thought that segments identified as unworn would be removed, however, because the removal of the wristband was generally associated with certain activities that may associate with happiness (such as exercise or showering) this information was left in as a potentially useful feature.

4.2.5. Output

As features are calculated (in order of signal type) they are stored in a results matrix. The reason they are not immediately printed is that we would like to print all features associated with one label on the same line. Seen as we process the data in the order of signal type (and not in the order of labels) if we were to print out features as they are calculated then the order would be wrong, hence we save them in a matrix first. Once all features are calculated the results matrix is written to a CSV.

4.2.6. Windowed Feature Extraction

The windowed feature extraction module uses a similar process to that described above for the standard feature extraction. The same features are calculated, except for accelerometer data which is left out due to the fact that it is not relevant in a lab training context where subjects are expected to be stationary. The standard process was able to process data on a file by file basis because there is a much smaller number of feature vectors computed, and this number is known ahead of time, all of these could simply be stored in a results matrix as described in the previous section. However, because the windowed feature extraction process produces many more features and the number is unknown a pre-allocated vector cannot be used in the same way. As such the approach taken was to process the data from all data types in the same session simultaneously so that all features for a single window are can be calculated and output together on the same line. If a large amount of data were used with multiple participants over multiple months this would also be a beneficial approach as it is very efficient in terms of space requirements. As with all programs in the system the code for this section can be found by following the link in the Appendix A.

4.3. Lab Training Module Implementation

The Lab Training Module takes is made up of 3 major components as can be seen in the diagram in Figure 7 found in section 3.5. The aim of this module is to use a pre-annotated database as the basis for pre-training a classifier in a lab setting. Of course in order to do this a pre-annotated database is required and so this section begins by explaining the choices made in preparing this database. Following this is a description of the first of the 3 major components, the Lab Trainer, which takes the set of images and their ratings provided by the database and shows these images to the user. The second component, the Physiological Data Preprocessor, is then briefly explored as it operates in a very similar manner to the Physiological Preprocessor found in the feature extraction module. Finally, this section finished by explaining the Lab Classifier Trainer which takes the features computed by the Physiological Data Preprocessor, and the labels computed by the Lab Trainer, and uses these to train a classifier which is saved for use in the final classification module.

4.3.1. Image Database Preparation

The database chosen for use was the Nencki Affective Picture System (NAPS) [29] because of the size of the collection with more than 1350 images all rated for both arousal and valence. The collection also contains a good range of images over the valence and arousal scales. The International Affective Picture System was intended to be used also, however, permission was never received from the authorising body.

Once access to the database was obtained, the ethics committee advised that not all of the images would be appropriate for use due to their potentially disturbing nature. Though this concern was initially in regards to volunteer participants, it was also thought that in the interests of the researcher's wellbeing (as key subject) that such disturbing photos should still be avoided. As such it was decided that a threshold would be set for a minimum valence score that images needed to reach in order to be included in the study. This score was determined by starting at the middle of the valence scale, with a value of five, and randomly selecting images of decreasing valence to check their appropriateness. This was continued until marginally suitable images were encountered. This technique, along with the reading of the image descriptions was used to establish a minimum valence threshold without wasting too many images by viewing them before actual training began, and without the researcher being subject to overly disturbing images. The chosen threshold value was a valence score of 3.12 out of 10.

4.3.2. Lab Trainer

The Lab trainer presents the images from the chosen database to the subject in order to elicit an emotional response, and at the same time records the time during which each image was presented. Previous studies [8] showed that it was effective to view images in order of increasing arousal so

that earlier images had less effect on the responses to later images. As such this technique was adopted in this study. The approach used was to group the images into sets and then provide 5 images in a row each for a minimum of 6 seconds. The order of groups used was:

1. Normalisation (low arousal, neutral valence)
2. Low arousal, high valence
3. Low arousal, low valence
4. Normalisation
5. Medium arousal, high valence
6. Medium arousal, low valence
7. Normalisation
8. High arousal, high valence
9. High arousal, low valence

Normalisation images were included in order to minimise the chance that previously viewed images would affect those viewed later. These images were chosen to be low arousal (so as to calm the subject) and neutral valence so as to minimally bias the subject before the subsequent group of images was displayed.

Once an image was displayed it is removed from the list of possible future images (unless the image is only used for neutralisation) because the second viewing of an image may likely not be as emotionally evocative. In order to ensure that images from previous sessions are not shown again, the list of previously used images is saved at the end of each session and read in at the start of any new session.

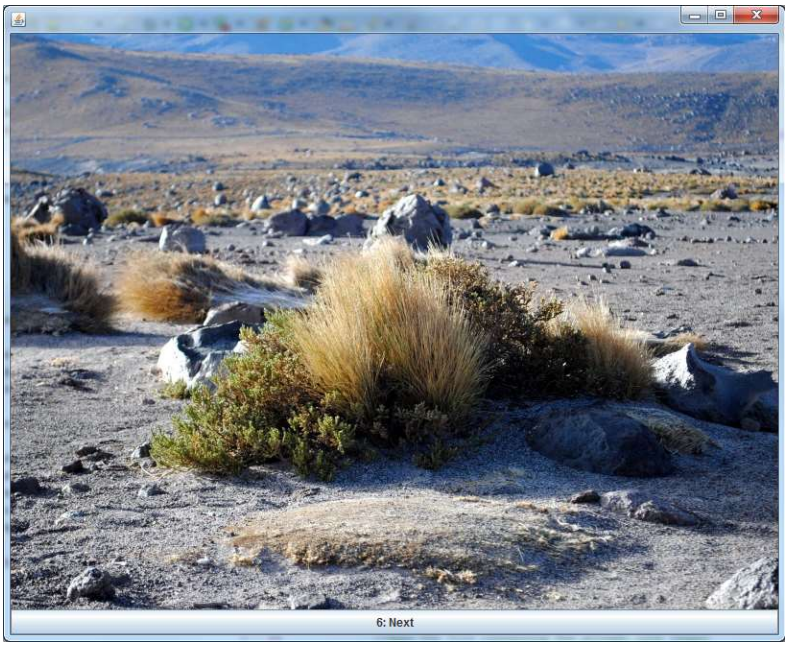

Figure 14 – Lab Trainer UI Screenshot

Images were chosen to be shown for a minimum of 6s, based on the times used in the literature, however, this was only a minimum. After 6s the button to allow the user to continue to the next photo (as can be seen in the screenshot in Figure 14 above) is enabled. However, if the subject chooses to engage with the image longer then that is also possible. Each time an image is displayed the start time for display is recorded, and it can be assumed that any time from this start time until the start time of the following images was spent looking at that image. Finally, when a session ends the end time is printed (in order to know the duration over which the final image was viewed).

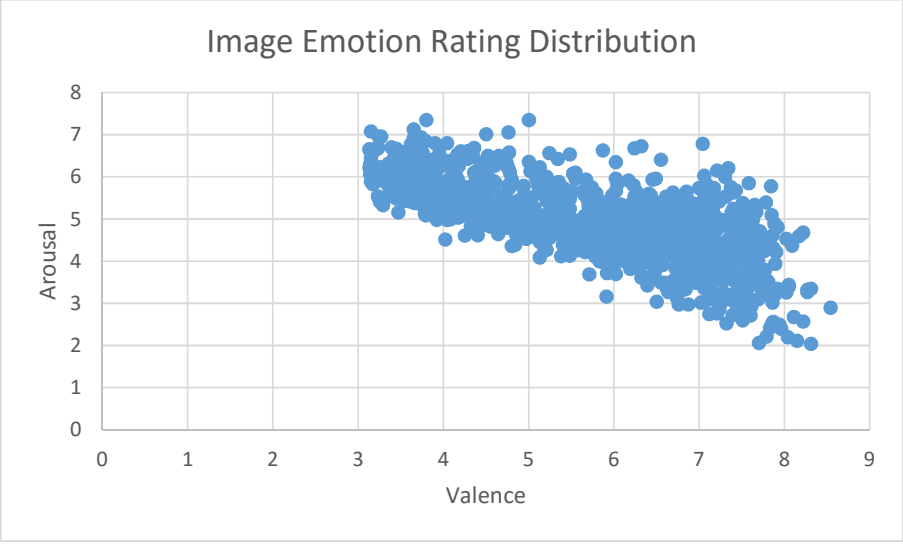

Figure 15 – Image Emotion Recognition Distribution Plot

As seen above the images were required to be placed into groups based on their arousal and valence scores (high, low or neutral). In order to balance these groups and to ensure that neutral was based properly around the average, the emotion rating distribution for all the images, shown in Figure 15, was considered. The most important consideration was that images were displayed in order of increasing arousal score and so the images were first split on arousal so that the low, medium and high arousal brackets were fairly balanced. Following this these brackets were split on valence. This process is shown in the following code snippet, keeping in mind that both arousal and valence are scored out of 10.

```
//Check this image hasn't been used unless it is a normalising image
if(!usedImageSet.contains(filename) || (arousal <= 4.54 && valence > 6.54
	&& valence <= 7.2)) {

	//First split on arousal, then on valence
	if(arousal <= 4.54) {
		//Low Arousal
		if(valence <= 6.54) {
			//Low Valence
			lalvFiles.add(filename);
		} else if(valence <= 7.2) {
			//Neutral Valence
			normFiles.add(filename);
		} else {
			//High Valence
			lahvFiles.add(filename);
		}
	} else if(arousal <= 5.23) {
		//Med Arousal
		if(valence <= 5.87) {
			//Low Valence
			malvFiles.add(filename);
		} else {
			//High Valence
			mahvFiles.add(filename);
		}
	} else {
		//High Arousal
		if(valence <= 4.3) {
			//Low Valence
			halvFiles.add(filename);
		} else {
			//High Valence
			hahvFiles.add(filename);
		}

	}
}
```

Finally, the ratings that are provided for each image by the data base need to be translated into a score that falls on the scale that has been used in the rest of the study i.e. a rating from 1 to 5. This

was done using thresholds as shown in the following code snippet, with the levels for each threshold having been determined by manually reviewing images after testing had taken place.

```
if(valence < 4.2) {
	rating = 1;
} else if(valence < 5.2) {
	rating = 2;
} else if(valence < 6.2) {
	rating = 3;
} else if(valence < 7.2) {
	rating = 4;
} else {
	rating = 5;
}
```

4.3.3. Physiological Data Preprocessor

The Physiological Data Preprocessor in this module is almost an identical copy of that used in the feature extraction module explained in section 4.2.6 to create windowed feature vectors. Clearly in this module only the segments of physiological data relating to times where the Lab Trainer was run are input to the system. The other input is the list of ratings times which are received from the first Lab Trainer component described above.

4.3.4. Lab Classifier Trainer

The Lab Classifier Trainer is a very simple program that takes the features and labels generated by the aforementioned components in this module and uses them to train a classifier. This classifier was chosen to be very simple. An SVM classifier was taken as an arbitrary choice because preliminary results without the Lab Classifier Trainer showed KNN and SVM to be equally as effective on physiological data. Hyper parameters were also chosen to be in a standard range that preliminary results had shown to be effective. Of course, a grid search in order to establish optimal hyper parameters and classifier would be ideal here and this is a limitation that is discussed at the end of this chapter.

The core code of this simple classifier, which makes use of the Scikit Learn libraries, is shown below (as it is quite a short code snippet). More detail is provided on classifiers, and their hyper parameters in the next section of this chapter.

```
#Open the csv file of physiological features output by matlab
file = open("pretrainerFeatures.csv", "r")
physData = preprocessing.scale(np.loadtxt(file,
delimiter=","))

#Load the target ratings
file = open(ratingsFile, "r")
physTarget = np.loadtxt(file, delimiter=",")
```

```
classifier = svm.SVC(gamma=1, C=1)

#Fit the classifier to the training data
classifier.fit(physData, physTarget)

#Save the classifier to be opened in another file
joblib.dump(classifier, 'pretrainedClf.pkl')
```

4.4. Classification Module Implementation

The classification module is used to investigate all the data that has been collected and processed to see whether or not meaningful predications are able to be made. This module makes use of the machine learning toolkit called Scikit Learn which is based in Python and as such this final module takes the form of a Python script. To explore the implementation of this module we first look at the pseudocode in a general overview. Following this, the finer details of the most important sections in the pseudocode are discussed.

4.4.1. Overview

The following pseudocode represents the procedure used by the classifier suite to set up the variety of scenarios, create and optimise a classifier for each of these scenarios, and then calculate performance metrics for evaluation (also for each scenario).

```
Choose settings to be used this run.

Initialise lists of label data types, feature types
scale types, and classifier types over which to test.

Initialise lists of hyper parameters and classifier
specific settings from which to choose.

For each scale type:
     For each label data type:
          Load the data associated with that label type.
          Calculate the lab training based features.
          For each feature type:
               Prepare the related features.
               Create training and validation sets.
               Compute the data distributions in these sets.
               Compute trivial classifier scores.
               For each score method:
                    For each ML algorithm:
                         Tune hyper parameters with
                         cross validation.
                         Apply feature selection and/or PCA.
                         Calculate performance measures.
                         Output performance results.
.
```

When the above code is run two files are generated. One is a human readable output file while the other is a CSV which can then be opened in an analytical software package such as Microsoft Excel to extract useful results.

4.4.2. Setting Selection and List Initialisations

A variety of settings are able to be selected when running the classifier suite which include options such as whether or not feature selection is enabled, which results are printed, and what debug information is displayed. Similarly, the range of data annotation types, feature types, scale types, and classifier types which are used to generate the different scenarios are also initialised at the top of the file. These initialisation lists are composed of:

- Data annotation types: Reviews, Ratings, Combination (of reviews and ratings)
- Feature types: Phone, Physiological, Both (phone and physiological), Physiological + lab training (PhysPlus), Both + lab training (BothPlus).
- Scale types: 5 point scale, 3 point scale
- Classifier types: Support Vector Machines (SVM), K Nearest Neighbours (KNN)

The range of data types are simply the two different styles that were collected with the app as well as a third style that is created by combining the two (taking ratings where possible and filling in the gaps with reviews). The feature types are the various combinations of phone, physiological and lab training features. The scale types refer to the scale that is used for labelling. The 5 point scale is the standard scale that emotion data was collected upon (with 5 points each corresponding to a different face which the user can select in the app as can be seen in section 4.1.5). The 3 point scale is a simplified version of this, grouping emotions as simply negative, neutral or positive. This is done by grouping labels of 1 or 2 out of 5 together to represent negative, and 4 or 5 out of 5 together to represent positive. Finally, the two classifier types used are the SVM and KNN classifiers provided by the Scikit Learn libraries.

As mentioned previously combinations of these different types are used to generate the variety of scenarios which are tested. All possible combinations of the 3 type groups above (data, feature, scale, classifier) are created and as such this leads to 60 (3x5x2x2) different classifier scenarios.

4.4.3. Preparing Features

As mentioned above there are 5 different feature types which are used in this module. The first two types, phone and physiological, are simply loaded from the files created by the feature extraction module. The phone feature set is ready to use straight from the file, however the physiological feature file contains some samples which were associated with corrupted data (represented in the data file as feature vectors of -1s). These are included simply so that the physiological feature

vectors can still be matched appropriately with the phone feature vectors when combined to create the "both" set. Before combining the index of each corrupt row is found and then the rows at these indexes are removed from copies of both the phone and physiological feature sets before concatenating the two horizontally (row 1 from one vector is concatenated with row 1 from the other).

In order to create the final two feature sets, physPlus and bothPlus, the extra windowed feature vectors are read and in and classified using the lab trained classifier. The results from this classification is used to create extra features which are concatenated horizontally to copies of the physiological and both feature vector sets to create 2 further sets. For each time period, there are 6 extra features created 5 of these are the percentage of window classifications that were made for each rating value during the given time period i.e. what percentage were 1s, what percentage were 2s and so on. The final extra feature is simply an average of the classifier ratings for all windows in that time period.

For each of these feature types the number of features, and the number of samples for each annotation type is displayed in Table 6. Note that the varied number of samples for a given annotation type is due to the fact that while phone data may not have been corrupted or missing for a period, this may not have been the case for the physiological data of that same period.

Feature Type	Num Features	Review Samples	Rating Samples	Comb'd Samples
Phone	107	383	251	469
Physiological	137	220	158	289
Both	244	220	158	289
PhysPlus	144	220	158	289
BothPlus	251	220	158	289

Table 6 – Feature vector lengths and sample sizes by feature and annotation type.\

4.4.4. Training and Validation Set Separation

In order to select appropriate hyper parameters and train each classifier produced, we separate our data into a training set and a Held Out Validation set. The Held Out Validation set is only used to obtain final performance measures after the hyper parameters have been optimised and the classifier trained. This ensures that there is no overlap between training and validation data. This implementation used 3 quarters of the data for training and 1 quarter for validation, with the division made by a random selection of indices which is different each time a scenario is evaluated. This can be seen in the code snippet below:

```
#Create a training set and validation set
indices = np.random.permutation(len(currentData))
numValSamples = math.ceil(len(currentData)/4)

data_train = currentData[indices[:-numValSamples]]
target_train = currentTarget[indices[:-numValSamples]]
data_val = currentData[indices[-numValSamples:]]
target_val = currentTarget[indices[-numValSamples:]]
```

Once these two sets are generated, their label distributions are then calculated and shown i.e. for each label value the percentage of the total labels with that value is calculated. The value which is most frequent (has the highest percentage of occurrences) in the training set is then identified and from this a "trivial classifier" which always predicts this most frequent value is evaluated on the validation set.

4.4.5. Machine Learning Algorithms and Hyper-parameters

As previously mentioned the two machine learning algorithms that are used in this study are SVM and KNN. Both algorithms were chosen based on their popularity in previous studies on emotion recognition and reported success relative to other algorithms in these studies. Emphasis is deliberately not placed on machine learning algorithms in this study as the focus is intended to be on annotation methods.

Support vector machines work by separating data in multiple dimensions using a set of hyperplanes. This can still be used to achieve essentially non-linear separation of data by using what is referred to as the "kernel trick". To achieve this the inputs are mapped into even higher dimensional vector spaces. One example of such a kernel is the radial basis functions, and this is the kernel that is used in the support vector classifier in this study.

K Nearest Neighbours is a simpler algorithm where a model is trained by essentially creating a database of all the training examples it receives. When a new sample is given, for which a prediction is to be made, the algorithm simply calculates which are the K nearest samples in its training database (nearest can be defined in a range of ways) and the associated true values for each of these K "neighbours" are counted and the dominant class is returned as the prediction.

Both of these classifier algorithms also require hyper parameters to be provided. For the support vector classifier, using a radial basis function (RBF) kernel, these parameters are:

- C – A prediction penalty weighting which essentially allows a trade-off to be made between training set classification accuracy and simplicity of the separating surface.

- Gamma – Determines the distance of influence of individual samples. Low gamma means that samples had a wide-ranging effect, where as high gamma means their region of influence is only small.
- Balanced – Decides whether or not the accuracy achieved over classes will be weighted naturally or "balanced" to account for datasets where the number of training samples in each class is not roughly equal.
- Linear – Decides whether the SVM will use a linear kernel (in which case the Gamma parameter is not applicable) or an RBF kernel.

On the other hand, the KNN classifier has only one hyper parameter that needs to be trained:

- K – The number of nearest neighbours to be considered when determining a new sample's value

In order to determine the values of these hyper parameters a grid search is carried out. To do this a set of possible values to be considered for each hyper parameter are provided e.g. C = [0.1, 1, 10, 100], and every possible combination of these hyper parameters is used to create a classifier which is evaluated using cross validation on only the training set. Then whichever set of parameters achieves the highest cross validation accuracy is chosen to train the final classifier which is then evaluation using the validation set.

4.4.6. Feature Selection and Transformation

Feature selection can optionally be carried out using sequential forward floating search. The library which was used to provide this in this study is known as mlxtend. Sequential forward floating search essentially selects features by beginning with the empty set and iteratively adding features whichever un-included feature increases the accuracy of the classifier the most. At each iteration, however, it is also checked that no currently included feature can be removed to increase the accuracy. This can also be applied during the grid search however it is prohibitively slow so instead in this study we opted to tune hyper parameters without feature selection and then use it at the end when evaluating performance. Principle Component Analysis was also used in a similar way. Both of these methods, however, did not show any significant improvement over the un transformed data and thus were not used in calculating final results. The implementation, however, can be seen in the files provided through the link to the code base in the Appendix A.

4.4.7. Performance Metrics

A range of performance metrics are calculated for each scenario. For a given scenario both the validation set performance and associated trivial classifier performance are calculated. These performances are each reported using 3 different scoring methods:

- Accuracy
- Mean absolute error
- Mean squared error

The above results are reported 3 separate times where in each case the hyper parameter tuning is carried out using a different one of the above 3 different scoring methods.

This results in 18 (3x2x3) different performance metrics being reported for each of the 60 scenarios. These are saved in 2 separate files. One which is human readable, and one which is a CSV. The results in the CSV were then analysed in Microsoft Excel to create tables and graphs which are presented in the next section.

4.5. System Benefits

There are a variety of ways in which the system designed particularly benefits this study in its aims to investigate the effectiveness of data annotation methods over other existing systems, and these will be explored in this section.

Firstly, in order to make fair comparisons of effectiveness between different annotation types it is important that annotation labels collected are able to be applied to the same data. This lessens the effect of systematic differences between data sets over which the effectiveness may be evaluated. This system provides this benefit by collecting both on the spot ratings and review data for the same time periods and hence can be applied to the same raw data.

Furthermore, one of the concerns that researchers might have with the method of using end of day reviews as an annotation scheme is that it might be difficult for users to remember what was going on at specific times throughout their day. If the user cannot remember what they were doing, then they have nothing upon which to base their assessment of their emotion. Simply asking the question “how were you feeling between 10am and 11am?” might not elicit very accurate assessments from users. This system, however, is able to combat this by providing the user with contextual information in the review activity in order to remind them of how they might have been feeling at the time.

Another concern that past researchers have shown is that the mere action of asking for users to submit ratings throughout their day might be considered obtrusive, possibly even having the effect of changing the users emotion if it is too taxing or difficult a task. In the worst case users may not even submit ratings if it is seen to be too much of a chore. This system has the benefit of a very simple annotation system. The user is only required to report on a single intuitive scale so very little thinking is required to decide what rating will be submitted. Furthermore, the user is prompted to submit ratings by push notifications which go straight to the rating activity so that ratings are

submitted using a total of 2 taps (one to open the notification, and one to select one of the 5 faces). These notifications also have the benefit of reducing the likelihood that users will forget to submit ratings.

Another key benefit is the flexibility and extensibility of the system. In all modules of the system it is relatively easy to add new feature or data types. For the sake of feature extraction of physiological signals, it suffices to add the name of the new signal's data file to the list for processing, and for the sake of phone data this task is as simple as adding a new class which inherits from the generic DataSensor class. In the classifier suite, this extensibility is again a key benefit. All the operations of the classifier suite have been made automatic so that new settings, approaches or parameters can quite easily be added. What's more, the use of the Scikit Learn libraries in this final module makes it very easy to adapt the current framework to include other types of classifiers, such as neural networks or logistic regression. This extensibility and flexibility is particularly useful for achieving the aims of investigating data annotation because the scope of investigations can easily be expanded to include new data and hence explore the effectiveness of the various strategies in new scenarios.

Finally, while many studies only report performance using accuracy, this system also evaluates performance using the distance based metrics of mean squared error and mean absolute error. This has the benefit of providing deeper insight into the way in which the generated classifiers are predicting values.

4.6. System Challenges

While this system provides many benefits, these did not come about without a range of associated challenges. As no study had investigated both reviews and ratings before simultaneously a new tool had to be developed to collect this information. This required an Android app to collect not only annotation data but also phone metadata. App development adopts quite a different paradigm to the approaches used to build and design other software systems and as such learning how to do this, from the basics, was a major challenge in implementation.

Similarly, once the overall design paradigm for apps becomes familiar there are still many challenges that arise when interfacing with various libraries and APIs. This application relied heavily on the Android system APIs to access logs and data directly for information such as call or message logs, or even to be able to send push notifications to the user. The application also made use of the google maps APIs, Scikit Learn, mlxtends, and of course the emotion sense libraries. Integrating with each of these is done in slightly different way and can be quite challenging to set up correctly. Even once the implementation using these libraries is complete there are still many challenges associated with testing, due to the fact that many of these libraries rely on real interactions with the device over a

long period of time. As such, test cases cannot be written to be performed programmatically and actually require test plans to be designed and carried out manually.

Of these libraries, specifically, EmotionSense provided the most technical challenges. It did not become apparent to the researcher, until much work had been done to learn and develop a strong understanding of Android programming, that the EmotionSense libraries were actually quite out of date. These libraries not only were incompatible with the more recent versions of the android operating system, but the open source project code base associated with the libraries wasn't even compatible with the currently available Android development tools (Android Studio). Even once a sufficient knowledge of Android had been achieved to fix the issues with the library, there were still a range of drawbacks that prevented the library for being used to collect certain data types. Thus, information such as app usage, call and message data was collected directly through the system APIs and automatic polling of sensors also needed to be implemented.

Significant challenges were also encountered in the preparation of an image database for use in the lab training module. Vetting images for appropriateness, to ensure that the content was not too disturbing, was difficult to do without firstly being exposed to too much disturbing content, or secondly seeing to many of the images and reducing their effectiveness at inducing emotions during the actual collection of data. Similar challenges were also encountered in deciding on the thresholds that would be used for arousal levels in order to split the data set into low, medium, and high arousal categories. Again, these challenges were encountered in deciding on the thresholds for translating the valence ratings provided by the database to the scheme used in the wider system.

There were also many challenges associated with the automatic generation, training and evaluation of such a large number of classifiers associated with the range of scenarios. A grid search needed to be run to select optimal hyper parameters and settings for each scenario and some of the settings being investigated were prohibitively inefficient and slow. Aside from the wide range of configurations being difficult to coordinate the issues of speed and efficiency also provided practical challenges.

The 4 main modules of this system each have quite unique and specialised functions. As a result, each module also uses quite a different set of tools to achieve these functions. In the case of software programs the most important type of tool is the programming language that is used. Due to the wide range of functions in the overall system a wide range of languages also needed to be learnt. These included Java, Matlab, Python and the Android Framework (based on Java).

Not all challenges were technical, however, and in fact one of the greatest challenges which in the end proved to be a limitation (as will be discussed in the following section) was meeting the requirements of the ethics committee. This study initially intended to use multiple participants to investigate how data collected methods were different between subjects. However, it soon became clear that this was not going to be feasible for an honours level thesis (which already required the implementation of full systems for collection, feature extraction, lab training, and classification) to be able to meet these requirements. A multitude of forms and paperwork were filled out in an attempt to gain ethics approval, however, the submission of these documents made clear that there were many additional challenges that needed to be considered to add further participants to the study on top of the researcher. Firstly, there would be a need to anonymise data and pay extra special attention to data security for all information collected from third parties. Secondly, there would have been a need for extensive technical support for participants who may have owned devices with a range of operating system versions each with their own subtleties and quirks. Thirdly, the requirements of the ethics committee prevented friends and acquaintances of the researcher from being recruited for the study and as such with minimal resources available to reward subjects for their participation recruitment would have posed many challenges.

4.7. System Limitations

Due to various factors, such as resources, available time and level of knowledge of the researcher there were limitations to what was possible in this study. Most of these limitations are issues that could be overcome given more time and resources or slightly modified approaches and as such suggest potential areas for future work.

One of the most important limitations was that the system was only able to be tested on one participant, and what's more that participant was the researcher. Having the researcher act as a participant has the potential to introduce what is known as experimenter bias, which is the tendency for results to shift towards the result which is expected to be received rather than what is actually observed [30]. There were many reasons for this limitation, however. Only one physiological data collection device was available and so data could not be collected from users in parallel. What's more even if there had been more devices much more work would have been required to recruit participants ethically, ensure that other ethical concerns were appropriately considered, add privacy preserving aspects to the data, and provide tech support for issues with different devices using the android app. As such using multiple participants was not feasible. Clearly this is an issue, however, because emotion recognition systems are highly user dependent and results could differ wildly between people. This is particularly the case for physiologically based systems as different users have different physiological responses to emotion. It is also not ideal for the researcher to be used

as a participant in the study as they are aware of how data is being collected, and how it will be analysed, and it is possible that even subconsciously this could affect the way they report data.

Another participant related limitation to this study relates to the range of emotions that the participant experienced over the course of data collection. The participant used in this study did not experience a large amount of negative emotions during data collection which affects the distribution of the annotation data collected.

Another limitation of this system is that it has only made use of a 1 dimensional model, which is essentially the valence axis from the circumplex model (even though this is never explicitly stated to the user, and they simply report how they are "feeling" by choosing the most relevant face). Most other studies also collect arousal data. This allows emotions to be separated more easily and often also provides more insight into the subtleties of the user's emotional state. The single axis was used for simplicity of submitting ratings and to reduce the complexity of the study.

The set of physiological signals which were collected and analysed were also limited to the set which were able to be collected by the Empatica E4 wristband. While this set does cover the most frequently used and proven successful signals there could be added benefit in the use of further signals but we are limited by the impracticality of asking the user to wear multiple wristbands or collection devices at once.

In relation to the lab training module there were some clear limitations imposed by a lack of time and resources. Firstly, a larger number of images for use in the lab training would have been beneficial. What's more the ratings provided for each image are obtained as an average of the ratings provided by a range of users, and as such may not be correct for any one individual as the effect of a specific image may be highly personal. Further limitations are seen in the classifier generation stage of the lab training such as the lack of hyper parameter tuning and investigation of different classifier types.

Due to the large range of programming languages and systems used, as was discussed in the previous section. It was difficult to incorporate all the aspects of the system into one automated script that would take the data collected as input and automatically return the performance results. As it is difficult to interface the various programming languages there is an element of integration between modules that is done manually by generating output files and providing these as input to the next program in the chain. This limits the ease of use of the system but doesn't reduce its effectiveness or performance.

The final limitation that is discussed is the fact that the periods which are used for prediction in this system never overlap. It was not investigated whether it could be useful to use data over longer periods of time before a label is given, for the calculation of features. Some other studies do use longer periods like this and it could prove to be beneficial.

5. Results

The primary aim of this study was to investigate data annotation methods for use in real world unconstrained contexts. After collecting and annotating various types of data, extracting features from this data, and then creating classifiers to test a wide range of different scenarios, a variety of performance measures were able to be calculated. These performance measures are analysed in this section in a variety of ways. As explained in section 4.4.2 there are 60 different scenarios created and each of these was evaluated using the validation set firstly with no feature changes, then with PCA, and finally with SFFS. It was found that both PCA and SFFS methods did not provide any significant improvements in accuracy and as such focus has been made on results for the unchanged validation set (also for the sake of keeping results brief).

Furthermore, for each of the 3 above feature transformation methods results were calculated when hyper parameter tuning was carried out with accuracy (ACC), mean squared error (MSE) and mean absolute error (MAE). However, it was found that the trends seen when using accuracy for tuning were the same as those seen when using the other metrics so again for the sake of brevity only the results attained using accuracy scoring for tuning have been included.

In order to meet the aim of investigating the effectiveness of different annotation types this chapter first compares the performance of the 3 annotation types (reviews, ratings, and combined) in a variety of ways. To begin the 3 types are compared by their results in all relevant scenarios. Then the performances of each annotation method for specific scale types, feature types, and classifier types are explored. This provides a good overview of each annotation type's effectiveness for different scenarios.

Following this the results address the second aim of investigating the potential of using pre-annotated stimuli to pre-train a classifier in a lab setting which can then assist in real world classification. This analysis is achieved by presenting the results of the classifiers for each scenario when split by feature type (rather than annotation type). This allows for comparison between the performance of scenarios with and without lab trained assistance.

There are 4 measures used to evaluate the effectiveness of either annotation types or feature types. The first is the number of scenarios in which the trained classifier was able to beat the trivial classifier. The second is the mean margin by which the trivial classifier was beaten (i.e. the mean of the difference between the trained classifier performance and the trivial classifier performance). The third is similar to the second but the mean is calculated only for the classifiers which successfully beat the trivial classifier. The fourth is the maximum margin. These measures help to give a good idea of how the various data annotation types have performed overall in the real world

unconstrained setting rather than looking at simply their best case performance under a restricted scenario. These 4 measures are presented in tables, and occasionally highlighted in graphs, in the subsequent sections.

These 4 measures all provide information about the effectiveness of a data set (and by extension the annotation scheme or features used) by examining how prediction can be improved above a trivial classifier. This is useful because the end goal for creating real world systems is to be able to predict how a person would rate their emotion at a given time without them having to actually submit the rating at all. This is why, for example, we deem an annotation type which is able to predict the labels given in its own scheme as being effective (i.e. an annotation scheme that produces predictable data), because in a practical application this is the behaviour that is desired.

5.1. Annotation Type Results

The results in this section compare the performance of the 3 different annotation methods: ratings, reviews and combined. The overall results are presented for all 3 scoring metrics: accuracy, MAE and MSE. These overall results show that the trends displayed by each of these metrics are the same and as such for the subsequent sections where annotation types are compared in specific contexts only the results for accuracy are presented. The complete table of raw results is available along with the system's code in Appendix A for the reader's reference.

5.1.1. Overall

The results in the following tables 7 to 9 show the performance of each annotation type in terms of accuracy, MAE and MSE respectively. As can be seen in Table 7 the combined annotation type is most effective in terms of percentage of trivial classifiers beaten, mean margin, mean successful margin, and max margin. For each of these measures combined has beaten the other two types significantly. Ratings appear to be slightly more effective overall then reviews in terms of accuracy, beating reviews for percentage of trivial classifiers beaten and performing comparably in the other metrics.

Accuracy	Combined	Ratings	Reviews
Percentage Beat Trv	0.8	0.65	0.55
Mean Margin	0.040805	-0.00581	-0.00558
MeanSuccessfulMargin	0.063231	0.029335	0.031905
Max Margin	0.19178	0.11111	0.07292

Table 7 – Accuracy results for each annotation type

The results based on mean absolute error in Table 8 tell a very similar story to accuracy. Combined is again performing convincingly better than the two other annotation types. The main difference

between accuracy and MAE is that ratings are being shown more clearly to be performing better than reviews in terms of every metric.

MAE	Combined	Ratings	Reviews
Percentage Beat Trv	0.85	0.75	0.65
Mean Margin	0.063958	0.02258	0.015729
MeanSuccessfulMargin	0.084068	0.056773	0.044391
Max Margin	0.20548	0.15	0.12727

Table 8 – Mean absolute error results for each annotation type

Mean squared error again tells a very similar story in which combined annotations are clearly outperforming the other methods. This time reviews appear to outperform ratings very marginally in terms of percentage of trivial classifiers beaten but ratings perform better with the 3 other measures.

MSE	Combined	Ratings	Reviews
Percentage Beat Trv	0.9	0.8	0.85
Mean Margin	0.111638	0.079365	0.063799
MeanSuccessfulMargin	0.130601	0.117956	0.083012
Max Margin	0.43836	0.35	0.27273

Table 9 – Mean absolute error results for each annotation type

The results in terms of percentage of classifiers beating the trivial are summarised graphically for each of these 3 metrics in Figure 16 below.

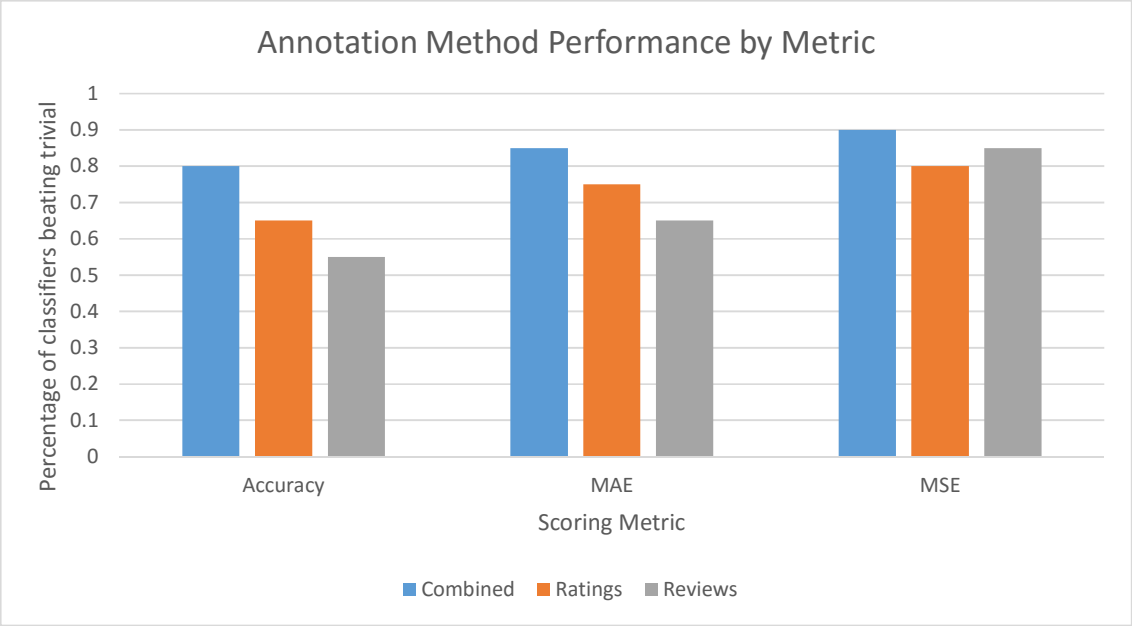

Figure 16 – Graph of annotation method performance by metric

5.1.2. Scale Type

The tables 10 and 11 below show how each annotation type performs for scenarios which use only the 5 point scale and the 3 point scale respectively. It is clear from Table 10 that the combined rating method is most effective in terms of every measure, however, it is closely followed by ratings which also appear to be quite effective. Reviews on the other hand appear to perform very poorly on the 5 point scale.

5 Point	Combined	Ratings	Reviews
Percentage Beat Trv	0.9	0.8	0.2
Mean Margin	0.075667	0.018135	-0.03534
MeanSuccessfulMargin	0.085016	0.035169	0.02727
Max Margin	0.19178	0.11111	0.05454

Table 10 – Results on 5 point scale for annotation types

Interestingly, Table 11 provides very different results to Table 10, showing reviews to be the best performing annotation type for a 3 point scale, at least in terms of number of classifiers beating the trivial. While the combined classifier still beats reviews by a small margin in the 3 other measures the two methods appear to perform quite similarly. Ratings, on the other hand, do not perform as the other two methods or as well as ratings on the 5 point scale.

3 Point	Combined	Ratings	Reviews
Percentage Beat Trv	0.7	0.5	0.9
Mean Margin	0.005942	-0.02976	0.024187
MeanSuccessfulMargin	0.035223	0.02	0.032934
Max Margin	0.08219	0.05	0.07292

Table 11 – Results on 3 point scale for annotation types

These results are summarised in Figure 17 below where the clear strengths of combined and ratings methods in 5 point and reviews in 3 point scale scenarios are highlighted.

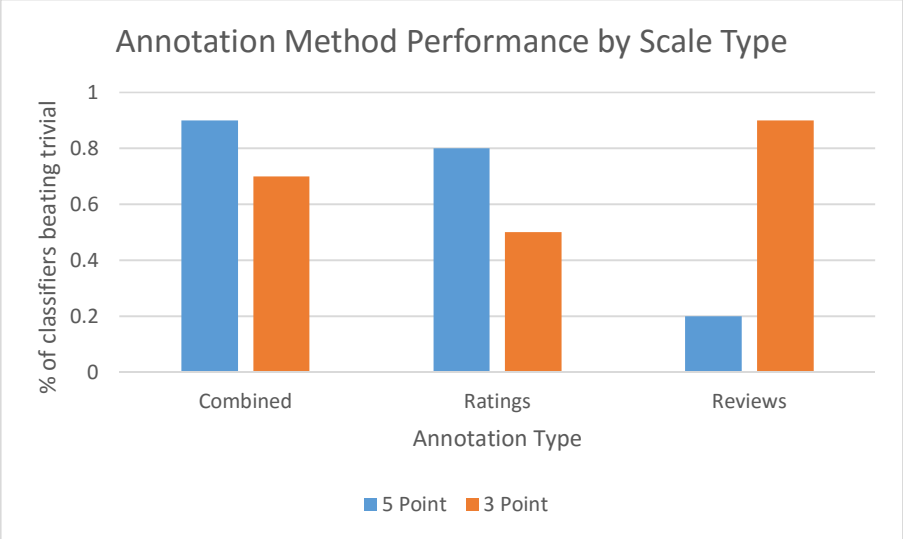

Figure 17 – Graph of annotation method performance by scale type

5.1.3. Classifier Type

The tables 12 and 13 below show how each annotation type performs for scenarios which use only the KNN and SVM classifier types respectively. Table 12 shows that the combined annotation type performs well according to all measures when using KNN while ratings and reviews appear to perform similarly and at a passable level.

KNN	Combined	Ratings	Reviews
Percentage Beat Trv	0.9	0.6	0.5
Mean Margin	0.037108	-0.01706	-0.00987
MeanSuccessfulMargin	0.063231	0.029335	0.031905
Max Margin	0.12329	0.09524	0.05455

Table 12 – Results for annotation types using KNN

Table 13 shows all 3 annotation types performing at a similar level.

SVM	Combined	Ratings	Reviews
Percentage Beat Trv	0.7	0.7	0.6
Mean Margin	0.056614	0.005437	-0.00129
MeanSuccessfulMargin	0.055614	0.032808	0.036395
Max Margin	0.19178	0.11111	0.07292

Table 13 – Results for annotation types using SVM

Finally, a comparison between the two classifier types for each annotation method seems to show combined performing slightly better using KNN while ratings and reviews perform slightly better with the SVM

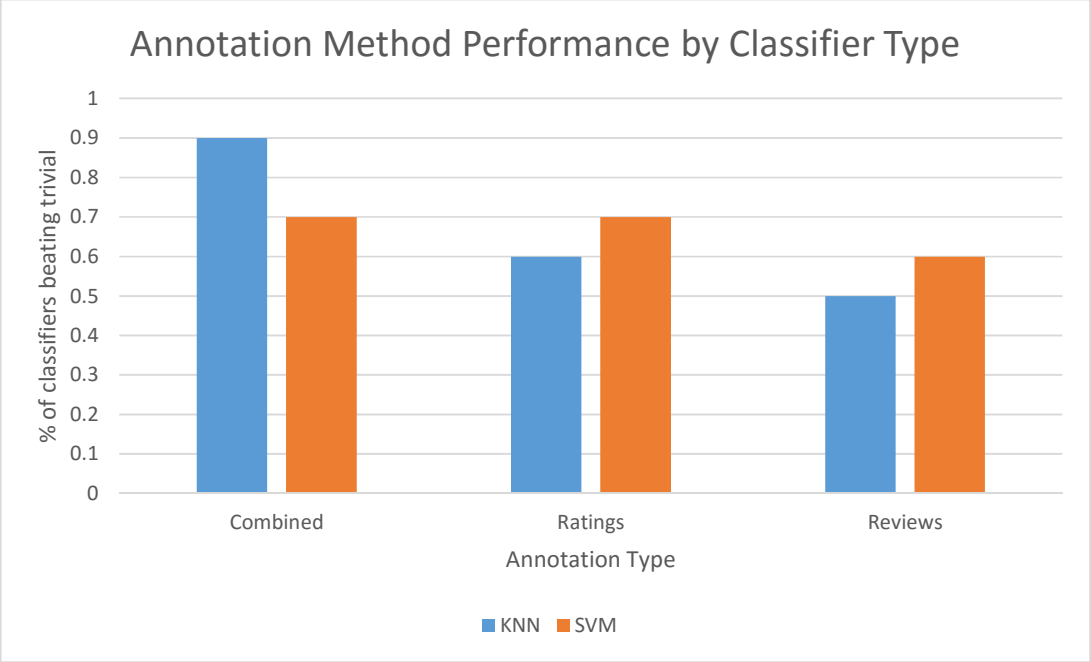

Figure 18 – Graph of annotation method performance by classifier type

5.1.4. Feature Type

The performance of the different data annotation types when used with specific feature types has been outlined in Table 14 below. Combined ratings appear to perform most effectively for all feature types except phone, for which both ratings and reviews appear to work equally well.

Feature Type	Measure	Combined	Ratings	Reviews
both	Percentage Beat Trv	1	0.25	0.25
	Mean Margin	0.113013	-0.075	-0.05909
	MeanSuccessfulMargin	0.113013	0	0.01819
	Max Margin	0.19178	0	0.01819
bothPlus	Percentage Beat Trv	1	1	0.5
	Mean Margin	0.06849	0.03125	-0.00454
	MeanSuccessfulMargin	0.06849	0.03125	0.027275
	Max Margin	0.17808	0.05	0.05455
phone	Percentage Beat Trv	0.25	0.5	0.5
	Mean Margin	-0.02542	0.039685	0.013023
	MeanSuccessfulMargin	0.02542	0.103175	0.057295
	Max Margin	0.02542	0.11111	0.07292
physiological	Percentage Beat Trv	1	0.75	1
	Mean Margin	0.017125	-0.0125	0.013635
	MeanSuccessfulMargin	0.017125	0.008333	0.013635
	Max Margin	0.0411	0.025	0.05454
physPlus	Percentage Beat Trv	0.75	0.75	0.5
	Mean Margin	0.030818	-0.0125	0.009085
	MeanSuccessfulMargin	0.063923	0.008333	0.05454
	Max Margin	0.09589	0.025	0.07272

Table 14 – Results for annotation types using different feature sets

The applicability of annotation types to feature sets is further explored in the graph of Figure 19.

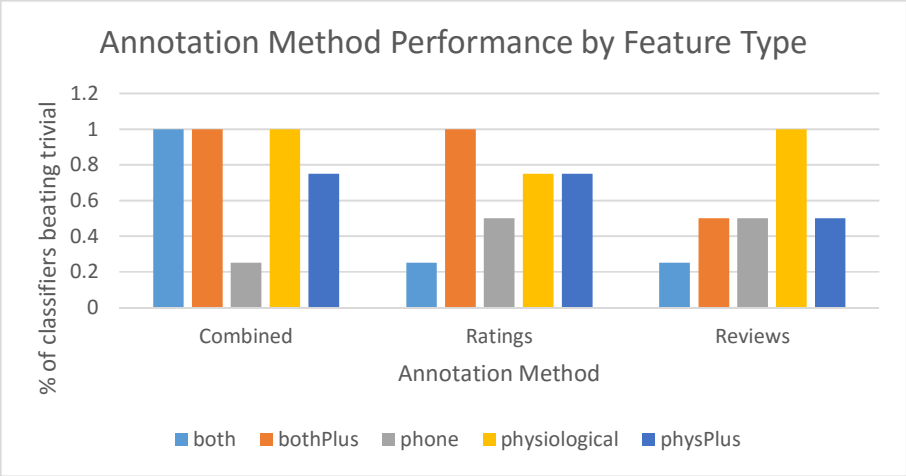

Figure 19 – Graph of annotation method performance by feature type

5.2. Feature Type Results

The results in this section aim to compare the effectiveness of the different feature types used in this study and specifically to investigate whether the addition of lab training has improved the effectiveness of any features. Table 15 shows a summary of the results for all measures. Physiological features appear to be the most successful when considering percentage of classifiers beating the trivial, however, the mean margin measure shows the bothPlus feature type to be the most successful. Upon examination, there does not appear to be a consistent trend between any of the measures.

	both	bothPlus	phone	physiological	physPlus
Percentage Beat Trv	0.5	0.833333	0.416667	0.916667	0.666667
Mean Margin	-0.00702	0.031733	0.009095	0.006087	0.009134
MeanSuccessfulMargin	0.078373	0.045351	0.069272	0.013458	0.040731
Max Margin	0.19178	0.17808	0.11111	0.05454	0.09589

Table 15 – Results for feature types

The results of the percentage of classifiers which beat the trivial classifier for each feature type are shown in the graph of Figure 20 below, however, as noted above the trend shown by this measure does not appear to correlate with the trends of any of the other measures.

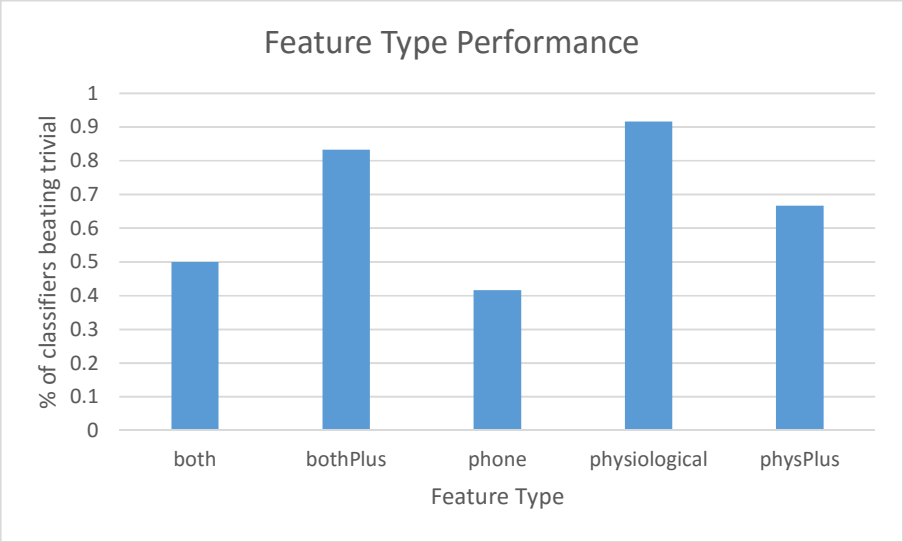

Figure 20 – Graph of annotation method performance by feature type

6. Discussion

In this section, we explore the implications of the results presented in the previous section. In order to do this, we will frequently refer back to the aims presented at the beginning of this paper and explore how the system has met these aims. The first results to be discussed will be those relating to the performance of annotation types. By exploring the overall results and then the results by scale, classifier and feature type it should be possible to make inferences about which annotation types are most appropriate for emotion recognition classification in a variety of contexts. Following this the results of each feature type are examined in order to determine whether any conclusions can be drawn about the usefulness of lab training for applications in real world unconstrained contexts. After discussing all these results, we make a series of comparisons between our findings and the relevant literature, before noting some of the potential future improvements.

6.1. Annotation Type Discussion

6.1.1. Overall Results

The overall results examined the performance of each classifier and then split the classifiers by annotation type to compare the effectiveness of each annotation type more broadly over a range of scenarios. These results were presented using the 3 metrics of accuracy, MAE and MSE.

For accuracy, it is clear that the combined classifier is performing most effectively. It has beaten the trivial classifier 80% of the time (16/20) which is 15% more than the ratings were able to achieve and 25% more than the reviews. In terms of how much it beat the trivial classifier by, Table 7 shows a 4% mean margin, and a 6% mean margin when only accounting for the classifiers which beat the trivial. The maximum increase in accuracy that was able to be achieved was 19%. These results are convincingly better than those of the ratings and reviews with mean margins around zero, mean successful margins around 3% and maximum margins closer to 10%.

Ratings also appear to be performing slightly better than reviews for the most part. In terms of accuracy, ratings beat the trivial classifier 65% of the time (13/20) while reviews only managed 55% (11/20). Rating was also more successful in terms of max margin with an 11% increase versus 7% for reviews. However, their performance was almost identical in terms of mean margins. Both MAE and MSE show a similar trend in the relationships between performance of the annotation types. As a result of the similarity in these trends the subsequent sections will only explore results based on accuracy for the sake of simplicity (as explained in section 5.1).

From the above observations, which are highlighted in the graph of Figure 20, it can be concluded that, in general when applied broadly to a range of scenarios, this study's novel combined review and rating scheme is the most effective at producing predictable data. Confidence can be taken in

this assessment as it is backed up by the results not only for accuracy but also for MSE and MAE. This result is also what one might intuitively expect because in essence the combined scheme provides more information to the classifier. By providing more frequent ratings the time periods over which features are calculated become smaller and thus it should make it easier for the true cause of the emotional response to become apparent. For example, let's imagine that the system received a rating of 5 for the time period from 10:00 to 12:30. Let us also assume that the true cause for the high rating was a string of messages sent to and received from the user's best friend between 12:00 and 12:30. Finally, let us assume that from 10:00 to 12:00 the user was at work and was calling clients and colleagues and that this would generally result in a neutral mood. In this case the time period from 10:00 to 12:00 is not relevant to the rating and the events that took place during this time are going to act like noise in the system, obscuring the true reason for the rating. However, if more frequent ratings were given, say for example a rating of 3 at 11:30 and as well as the rating of 5 at 12:30 then it becomes more apparent that calling work colleagues results in neutral ratings, and calling the user's friend results in high ratings.

6.1.2. Scale Type

When the annotation types are evaluated by splitting up the 5 point and 3 point scale scenarios and analysing the performance over each other these sets separately another very interesting result becomes apparent. For the 5 point scale it appears that the novel combined method is again performing most effectively with 90% of classifiers beating the trivial (9/10), a mean margin of 7.5%, and mean successful margin of 8.5% and a max margin of 19%. Ratings appear to be performing only slightly worse with 80% beating the trivial (8/10), a mean margin of 1.8%, a mean successful margin of 3.5% and a maximum margin of 11%. Reviews on the other hand have performed quite poorly with only 20% beating the trivial classifier (2/10).

The results for the 3 point scale, however, tell a very different story. While combined ratings still perform well beating the trivial classifier 70% of the time (7/10), with a mean margin of 0.5%, a mean successful margin of 3.5% and a max margin of 8%, it appears that reviews are actually performing much better. In this instance reviews are beating the trivial classifier 90% of the time with mean margin of 2.4%, mean successful margin of 3.2% and max margin of 7.2%. Ratings appear to be having only average performance beating the trivial classifier 50% of the time.

From these observations, it is possible to draw a number of interesting conclusions. Firstly, it appears that the combined classifier is again the most effective for a 5 point scale. As explained in the previous section this is not all that surprising. However, if only one rating scheme is desired then ratings perform quite well on their own also, much better than reviews. This also makes intuitive sense as a 5 point scale requires information about the subtleties of emotion as it needs to

differentiate between happy and very happy. Ratings should work well for capturing these subtleties as they are submitted as the user is experiencing the emotion. Reviews on the other hand have the potential to miss these subtleties if the user can't remember them at the end of the day.

When a 3 point scale is desired, however, then it is most interesting to see that reviews are the most effective annotation technique beating not only ratings but also the combined technique. It also makes intuitive sense that reviews should perform better on a 3 point scale. As a subject reflecting on the emotions felt at various points in the day it is quite easy to remember (with the assistance of the data shown in reviews) generally how one was feeling (negative, neutral, positive) however it is much harder to remember whether the emotion was "happy" or "very happy". This is one reason why reviews might be expected to work well on a 3 point scale. Another view could be that the time separation between experiencing and reporting the emotions might act like a low pass filter on the perception of emotion, removing the high frequency emotional variations and leave the low frequency general trends. This result could be quite useful for a range of applications which only require a 3 point scale because many users would consider reviews to be a less intrusive option for annotation as users are not interrupted throughout the day.

6.1.3. Classifier Type

This section discusses the results that were achieved when the different annotation types were used with specific classifier types as seen in section 5.1.3. The combined method is clearly more effective that the other 2 methods when used with KNN, however, in general the combined method is more effective than these two and as such it is not justified to attribute this success to the classifier type.

If comparisons are made between the performance of an annotation type for the 2 different classifiers, as is done in Figure 18, it would appear at first that the combined method is most suited to KNN and that ratings and reviews work better with SVM. This conclusion could make some intuitive sense, perhaps due to that way in which KNN looks for nearest data points as oppose to SVM which tries to separate data with hyperplanes. The latter might be more adversely affected by the combination of data from two different annotation systems. However, when the results are examined more closely it is difficult to see a clear pattern. While the combined classifier beats the trivial 90% of the time for KNN and only 70% for SVM, it has lower mean and max margins of 3.7% and 12.3% for KNN compared to 5.7% and 19% for SVM. The measures seem to conflict less in the case of ratings and reviews which perform consistently better across the SVM measures, however, the difference in performance is quite small.

Overall these patterns do not appear to be very strong. This suggests that the success of a classifier is not closely linked to the annotation type used, however, more work should be done to reach a confident conclusion.

6.1.4. Feature Type

This section explores the relationship between the performance of annotation types and the feature sets which they are used with. The results appear to show that the combined annotation type is the most effective for all feature sets except phone. This results suggests that the novel combined type has wide scale applicability. As previously explained this would make sense as this annotation type, in a sense, contains the most information. However, it is important to note that in the case of these results only 4 scenarios are used to calculate the performance of each annotation type for each feature set. This is because out of the 60 scenarios, only 12 belong to each feature type. Then of these 12 for each feature type only 4 refer to each annotation type.

Behind the combined annotation method, ratings and reviews seem to perform similarly well. Ratings show stronger results for the bothPlus, and physPlus feature sets. Whereas reviews show the best performance on physiological features. The patterns here are not all that strong, however, as discussed above, and it is hard to justify drawing solid conclusions from these particular results.

6.2. Lab Training and Feature Type Discussion

The purpose of providing results of the effectiveness of each feature type, as oppose to each annotation type as has been done up until now in section 6.1, is mainly to compare the effectiveness of scenarios which use the lab training features and those which don't. Thus, the main focus is on comparing the physiological features against the physPlus features, and the 'both' features with the bothPlus features.

In the case of the 'both' features, the addition of lab training based features seems to have improved effectiveness. With an increase in the percentage of classifiers beating the trivial from 50% to 83%, and mean margin increase from -0.7% to 3.2%. The mean successful margin and max margin, however, are slightly worse after adding lab training features. While these results seem somewhat conflicted they do seem to suggest that the lab training has improved effectiveness. Both max margin and mean successful margin are less useful at indicating the general performance across the range of scenarios (as they both focus on only a subset of the scenarios that relate to the given feature type).

The comparison between physiological features and physPlus features provides even less clear results. While physPlus performs significantly worse in terms of percentage of classifiers beating the trivial with 67% compared to 92%, it performs slightly better for mean margin with 0.9% compared

to 0.6%, mean successful margin with 4% compared to 1.3% and max margin with 9.6% compared to 5.5%. This makes it difficult to say that one feature set has performed better than the other.

There do not appear to be any other consistent trends in the data and because those which have already been identified are not very strong there is no real justification for drawing any conclusions about the effect that lab training has had on the effectiveness of classification. The comparison between the both and bothPlus feature sets has shown that there is potentially a benefit from this technique, however, more work needs to be done to establish this with confidence. Some of the limitations associated with the lab training system were discussed in section 4.7 and future work will be discussed further in section 6.4 below.

6.3. Comparison with Previous Works

No previous work, that this study is aware of, has ever combined ratings and reviews to create a composite annotation method nor have they compared annotation methods in an experimental fashion. However, the result that was achieved is what was expected. Firstly, the fact that a combined method is more successful than its two constituent methods individually, makes sense in that a larger amount of useful information is being provided to the classifier.

The majority of other studies only report their results in terms of accuracy and measures such as MAE and MSE are not used extensively. However, those studies which have explored the use of such measures for classifying ordinal scales have found it to be a useful scale [27, 28] so it is unsurprising that it frequently reinforced the trends seen in accuracy results.

The scales used in other studies are most frequently a 2 or 3 point scale, and these are often used in conjunction with reviews [4, 5] so it is not all that surprising that the 3 point scale was most effective with this annotation type. Ratings have also previously been used successfully on 5 point scales such as in the Moodscope app [2].

The result of both KNN and SVM providing similar levels of effectiveness is not unsurprising either as both classifiers were chosen due to their popularity in the emotion recognition domain, both having been used extensively before as is discussed in section 2.5.2.

6.4. Suggestions for Future Work

Being an honours thesis, this study was limited in terms of the scope that was able to be addressed. There is still much future work that could be done to build upon the findings of this study and to extend the systems developed. Some of the most prominent opportunities for future work are discussed in this section.

As was identified in section 4.7 one of the major restrictions in the scope of this study was the number of participants used. With more time and resources, a lot of interesting investigations could be made into the effectiveness of this system over a variety of participants. Not only would more participants provide more data to give more confidence to results but this would also allow for an exploration of the effect of participant backgrounds on the system's effectiveness. Perhaps the study could be expanded to include subjects who have pre-existing diagnoses of mental illnesses such as depression or anxiety as these people might experience a different spectrum of emotions to a subject with normal mental health. This would also allow for investigation into the applicability of such systems to health care. Furthermore, if a number of participants were used then it would be possible to investigate the potential of using global training methods where the data from many subjects is used to train a model which is then applied to new individuals. If this were possible to do effectively this could be highly beneficial to real world applications as training time for new users would be greatly reduced.

For simplicity, this study only investigated a single dimensional emotion model, asking the user to rate how they were feeling by selecting 1 of 5 faces representing different valence levels. Many studies also investigate an arousal dimension as well and it would be useful to see whether arousal has similar relationships with the annotation methods to valence. For practical applications, it is also beneficial to be able to predict arousal as it allows models to distinguish between more complex emotions such as anger or sadness.

The results achieved from the lab training module were not conclusive and as such further work on this area would be useful. There are many improvements that could be made in future work. Firstly, more images could be sourced to allow for further training. In order to combat the differences in personal responses to images, verification could be provided by subjects after completing training sessions to ensure that the intended emotional response for each image was in fact what they experienced. Finally, in the building of the lab trained classifier, further setting and hyper parameter tuning could be used to ensure that the optimal classifier is being created.

As was mentioned above there would be great benefit in investigating the applicability of these annotation methods to real world applications such as mental health monitoring systems, or adaptive software like mood based recommender systems or education platforms.

7. Conclusions

Comparing the annotation methods of 'on the spot' ratings, retrospective reviews and the novel combination of the two, in a variety of real world unconstrained settings, suggests that reviews are more successful at recording simple emotion (i.e. the 3 point scale of happy, neutral or sad) while ratings are better at discerning emotions on a more detailed 5 point scale. However, the best results for a detailed 5 point scale come from combining ratings and reviews using the novel technique described in this study. As well as these methods, the technique of image pre-training was also designed and tested. While it showed potential for the enhancement of combined phone and physiological classifiers, the results for purely physiological classifiers were less encouraging. Therefore more work is required in this area.

On top of these findings this study has also seen the implementation of a full Android application for emotion data collection with annotation at a timescale of much higher granularity than has been seen before for phone, or combined phone and physiological, emotion recognition systems. This application also features a new method for stimulating a user's memory when submitting reviews. Similarly, a full lab training system has been implemented for the investigation of the application of lab trained models to real world unconstrained contexts. A final product of this study is the data set, which was collected in a real world unconstrained setting for one individual and annotated with emotion labels for a period of around one month.

There are also many ways in which the work done in this study could be extended. Suggested future work could include adding a wider range of participants, investigating other dimensions of emotion, refining the lab training system, and investigating real world applications of these annotation methods.

8. Bibliography

1. Rachuri, K.K., et al. *EmotionSense: a mobile phones based adaptive platform for experimental social psychology research*. in *Proceedings of the 12th ACM international conference on Ubiquitous computing*. 2010. ACM.
2. LiKamWa, R., et al. *Moodscope: Building a mood sensor from smartphone usage patterns*. in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. 2013. ACM.
3. Constantine, L. and H. Hajj. *A survey of ground-truth in emotion data annotation*. in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. 2012. IEEE.
4. Jaques, N., et al. *Predicting students' happiness from physiology, phone, mobility, and behavioral data*. in *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*. 2015. IEEE.
5. Sano, A., et al. *Recognizing academic performance, sleep quality, stress level, and mental health using personality traits, wearable sensors and mobile phones*. in *Wearable and Implantable Body Sensor Networks (BSN), 2015 IEEE 12th International Conference on*. 2015. IEEE.
6. Sano, A. and R.W. Picard. *Stress recognition using wearable sensors and mobile phones*. in *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*. 2013. IEEE.
7. Bamidis, P.D., et al., *An integrated approach to emotion recognition for advanced emotional intelligence*, in *Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction*. 2009, Springer. p. 565-574.
8. Haag, A., et al. *Emotion recognition using bio-sensors: First steps towards an automatic system*. in *Tutorial and research workshop on affective dialogue systems*. 2004. Springer.
9. Lisetti, C.L. and F. Nasoz, *Using noninvasive wearable computers to recognize human emotions from physiological signals.* EURASIP Journal on Advances in Signal Processing, 2004. **2004**(11): p. 1-16.
10. Charland, L.C., *Emotion experience and the indeterminacy of valence.* Emotion and consciousness, 2005: p. 231-54.
11. Lee, C.K., et al. *Using neural network to recognize human emotions from heart rate variability and skin resistance*. in *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*. 2006. IEEE.
12. Wagner, J., J. Kim, and E. André. *From physiological signals to emotions: Implementing and comparing selected methods for feature extraction and classification*. in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. 2005. IEEE.
13. Lang, P.J., M.M. Bradley, and B.N. Cuthbert, *International affective picture system (IAPS): Technical manual and affective ratings.* NIMH Center for the Study of Emotion and Attention, 1997: p. 39-58.
14. Dan-Glauser, E.S. and K.R. Scherer, *The Geneva affective picture database (GAPED): a new 730-picture database focusing on valence and normative significance.* Behavior research methods, 2011. **43**(2): p. 468-477.
15. Bianchi-Berthouze, N. and A. Kleinsmith, *11 Automatic Recognition of Affective Body Expressions.* The Oxford Handbook of Affective Computing, 2014: p. 151.
16. Picard, R.W., E. Vyzas, and J. Healey, *Toward machine emotional intelligence: Analysis of affective physiological state.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2001. **23**(10): p. 1175-1191.

17. Jerritta, S., et al. *Physiological signals based human emotion recognition: a review*. in *Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on*. 2011. IEEE.
18. Carvalho, S., et al., *The emotional movie database (EMDB): a self-report and psychophysiological study.* Applied psychophysiology and biofeedback, 2012. **37**(4): p. 279-294.
19. Ekman, P., R.W. Levenson, and W.V. Friesen, *Autonomic nervous system activity distinguishes among emotions.* Science, 1983. **221**(4616): p. 1208-1210.
20. Madan, A., et al., *Sensing the" health state" of a community.* IEEE Pervasive Computing, 2012(4): p. 36-45.
21. Yang, N. and A. Samuel, *Context-rich Detection of User's Emotions using A Smartphone.*
22. Garbarino, M., et al. *Empatica E3—A wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition*. in *Wireless Mobile Communication and Healthcare (Mobihealth), 2014 EAI 4th International Conference on*. 2014. IEEE.
23. Lathia, N., et al. *Open source smartphone libraries for computational social science*. in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. 2013. ACM.
24. Robert, C., *Machine Learning, a Probabilistic Perspective.* CHANCE, 2014. **27**(2): p. 62-63.
25. Gross, J.J. and R.W. Levenson, *Emotion elicitation using films.* Cognition & emotion, 1995. **9**(1): p. 87-108.
26. Uhrig, M.K., et al., *Emotion elicitation: A comparison of pictures and films.* Frontiers in psychology, 2016. **7**.
27. Gaudette, L. and N. Japkowicz. *Evaluation methods for ordinal classification*. in *Canadian Conference on Artificial Intelligence*. 2009. Springer.
28. Baccianella, S., A. Esuli, and F. Sebastiani. *Evaluation measures for ordinal regression*. in *2009 Ninth international conference on intelligent systems design and applications*. 2009. IEEE.
29. Marchewka, A., et al., *The Nencki Affective Picture System (NAPS): Introduction to a novel, standardized, wide-range, high-quality, realistic picture database.* Behavior Research Methods, 2014. **46**(2): p. 596-610.
30. Kantowitz, B.H., H.L. Roediger III, and D.G. Elmes, *Experimental psychology*. 2014: Nelson Education.

Appendix A

The code for all modules can be found in the following public Github repository:

https://github.com/abar654/ThesisCodeBase

Appendix B

Full list of phone based features

Calls	numIncomingCalls,numOutgoingCalls,percentageIncomingCalls,totalNumCalls,callsOverallTimeSD,callsMeanIndivSDs,incomingCallDurationSum,outgoingCallDurationSum,totalCallDurationSum,percentCallDurationIncoming,incomingCallDurationMean,incomingCallDurationSD,incomingCallDurationMedian,outgoingCallDurationMean,outgoingCallDurationSD,outgoingCallDurationMedian,totalCallDurationMean,totalCallDurationSD,totalCallDurationMedian,maxCallDurationOneContact,percentCallDurationToOne,totalNumberContactsCalled,numIncomingContactsCalled,numOutgoingContactsCalled,percentageIncomingContactsCalled,maxCallsToOneContact
Messages	numIncomingMsgs,numOutgoingMsgs,percentageMsgsIncoming,totalNumMsgs,overallMsgTimesSD,meanIndivContactMsgTimeSDs,incomingMsgLengthSum,outgoingMsgLengthSum,totalMsgLengthSum,percentMsgLengthIncoming,incomingMsgLengthMean,incomingMsgLengthSD,incomingMsgLengthMedian,outgoingMsgLengthMean,outgoingMsgLengthSD,outgoingMsgLengthMedian,totalMsgLengthMean,totalMsgLengthSD,totalMsgLengthMedian,maxMsgLengthOneContact,percentMsgLengthToOne,totalNumberContactsMsged,numIncomingContactsMsged,numOutgoingContactsMsged,percentageContactsMsgedIncoming,maxMsgsToOneContact
App Usage (note SM refers to social media)	totalAppUsages,overallAppTimeSD,meanIndivAppTimeSDs,appDurationSum,appDurationMean,appDurationSD,appDurationMedian,maxDurationOneApp,percentTimeMaxApp,totalNumberAppsUsed,maxNumUsesOneApp,smDurationSum,smNumUses,smDurationMean,smDurationSD,smDurationMedian,percentNumUsesSM,percentDurationSM
Connection State	connTimeSD,numTypeMobile,numTypeWifi,numTypeNoneOrOther,totalDurationNoneOrOther,totalDurationMobile,totalDurationWifi,mobileConnDurationMean,mobileConnDurationSD,mobileConnDurationMedian,wifiConnDurationMean,wifiConnDurationSD,wifiConnDurationMedian,otherConnDurationMean,otherConnDurationSD,otherConnDurationMedian
GPS	Distance,latMean,latSD,latMedian,lngMean,lngSD,lngMedian,sumMean,sumSD,sumMedian,latRange,lngRange
Screen	screenTimeSD,hourOfFirstUsage,hoursSlept,numScreenOns,totalScreenOnDuration,screenOnDurationMean,screenOnDurationStdDev,screenOnDurationMedian
General	hour_of_day