



DEPARTMENT OF COMPUTER SCIENCE

Investigating Whether Decision Trees Can Facilitate User Reflection on Their Self-Tracked Data



A dissertation submitted to the University of Bristol in accordance with the requirements of the degree
of Bachelor of Science in the Faculty of Engineering.

Thursday 4th May, 2023

Abstract

Many people track data, such as their step count or sleep quality, in order to gain insights into their behaviour so they can potentially change it. Along this path towards self-improvement lies a critical stage called reflection. Reflection is considered to occur when an individual is able to describe and explain their data, and is seen to be enhanced when they can meaningfully explore the relationships between their findings. Unless people reflect on their self-tracked data, they are unable to gain insights which would facilitate beneficial behaviour change. Personal informatics systems are designed to help people track data, reflect on it and act on their insights. However, research has identified that the reflection tools built in to these systems are not very effective.

Typically, personal informatics systems present data using graphs, with the aim of having users find patterns and draw conclusions about their data by themselves. However, there are a number of barriers to people using reflection tools, including finding it difficult to interpret graphs and not having enough time to analyse their self-tracked data.

This dissertation explores the effectiveness of decision trees to facilitate reflection by users on their self-tracked data. Decision trees are a machine learning approach that can predict the value of a target variable based on the values of predictor variables. They can therefore be used to identify relationships between self-tracked data and potentially facilitate reflection. They also have a number of properties that make them suitable for use on a mobile device.

This project involved developing a decision tree algorithm capable classifying self-tracked data on a mobile phone, building an app that enabled users to track personal data and provided a reflection tool driven by the decision tree classifier. This resulted in a novel solution aimed to make it easier for users to explore how some of their behaviours affected their mood, thereby facilitating reflection.

When given the choice between the standard approach of using graphs, or the app developed for this project, an overwhelming majority of participants chose the decision tree-based app as a tool for reflection, and felt like they were able to develop insights in to how their behaviour impacted their mood. This study demonstrates that a decision tree classifier can facilitate reflection on self-tracked data and is a promising approach for future personal informatics systems. We also conducted another study over a three week period, to observe the real-world feasibility of the app. This longer study found that users were able to self-reflect on their own data, and more importantly, some were able to make positive changes to their behaviour.

Technical Contributions:

- Developed a computationally inexpensive decision tree classifier, that is able to run on any standard smartphone.
- Developed a decision tree-based reflection tool that addresses the barriers commonly encountered by personal informatics systems, proposed by Li et al.

Research Contributions:

- Explored the accuracy of decision tree classifiers in predicting mood, based on self-tracked, behavioural data.
- Determined the acceptability and feasibility of decision trees as a tool for self-reflection.

Keywords: self-tracking, self-reflection, self-insight, self-improvement, personal informatics, decision tree

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

 Thursday 4th May, 2023

Contents

1	Introduction	1
2	Background	3
2.1	Introduction	3
2.2	Self-tracking & Personal Informatics	3
2.3	Self-improvement and the importance of Reflection	4
2.4	Why Personal Informatics systems need to change	4
2.5	What's considered reflection and how can we implement this	5
2.6	Decision Trees	6
2.6.1	Building a Decision Tree	6
2.6.2	Use Cases of Decision Trees	7
2.7	Choosing a platform and addressing privacy concerns	7
2.8	Utilizing Saliency to Encourage Reflection	8
2.9	Summary	8
3	Technological background	9
3.1	JavaScript, React Native & Expo Go	9
3.2	Firebase	9
4	Project Execution	11
4.1	Introduction	11
4.2	Getting Familiar with Decision Trees	11
4.3	Developing a JavaScript Decision Tree	12
4.3.1	Extracting Rules	12
4.3.2	Improving the Rule Extraction	13
4.3.3	Publishing our Decision Tree	14
4.4	Comparing the Decision Tree with SciKit Learn	15
4.5	Mobile Compatibility	17
4.6	Tracking data and Presenting Results	18
4.7	Creating a Prediction Card	18
4.7.1	Improving the Readability	18
4.7.2	Implementing Salient Properties	19
4.8	Creating a UI for an R2 Level of Reflection	19
4.8.1	Wheel Spinners	19
4.8.1.1	Vertical Wheel Spinners	19
4.8.1.2	Horizontal Wheel Spinners	20
4.8.2	Click Spinners	21
4.9	The CHIME app	21
4.9.1	Incentivising Data Collection and the Use of the Exploration Page	22
4.9.2	Authentication	22
4.9.3	Analytics	22
4.9.4	Changing the Default Voice	24
4.10	Summary	24

5	Critical Evaluation	25
5.1	Introduction	25
5.2	Poster fair study	25
5.2.1	Results	26
5.2.2	Poster fair study summary	27
5.3	User Evaluation in the Wild	27
5.3.1	Participants	27
5.3.2	End of Study Interviews	28
5.3.3	Results	28
5.3.3.1	App Experience	29
5.3.3.2	Mood Predictions & Exploring Relationships	29
5.3.3.3	Self-Reflection & Behaviour Change	30
5.3.3.4	Suggestions	30
5.3.4	Evaluation in the Wild Summary	31
5.4	Evaluating the Performance of the Decision Tree	31
5.5	Summary	32
6	Conclusion	33
6.1	Overview	33
6.2	Achievements Summary	33
6.3	Future work	34
A	Poster Fair Study	38
A.1	Questionnaire	38
A.2	Results	39
B	Evaluation in the Wild	40
B.1	Distribution	40

List of Figures

2.1	The Stage-Based Model of Personal Informatics Systems and its four properties: 1) barriers in a stage cascade to later stages; 2) stages are iterative; 3) stages are user- and/or system-driven, and 4) uni- or multi-faceted. The visuals for 3) and 4) can be used to show these properties for a particular system. [31]	4
2.2	Original reflection tool built-in to the CHIME app.	5
2.3	Illustration of a Decision Tree [36].	6
4.1	Accuracy of classifier for varying test dataset size.	11
4.2	Graphviz render of SciKit-Learn’s decision tree.	15
4.3	Graphviz render of our decision tree.	16
4.4	Difference in accuracy of our decision tree compared with SciKit-Learn’s.	16
4.5	Buildtime of decision tree on iPhone 13.	17
4.6	Prediction card.	18
4.7	Prediction card with improved readability.	19
4.8	Prediction card with traffic-light border.	19
4.9	Exploration UI using vertically stacked wheel spinners.	20
4.10	Exploration UI using horizontally stacked wheel spinners.	20
4.11	Exploration UI using click spinners.	21
4.12	Prediction ready card.	22
4.13	Settings page to change the default voice.	24
5.1	Choice of reflection tool when presented with the option of paper-based graphs or the decision tree-based app.	26
5.2	Ability to predict a user’s current mood	27
5.3	Ability to explain a user’s mood	27
5.4	Ability to suggest a mood boosting action	27
5.5	True and False Predictions	31
5.6	Cumulative accuracy of the decision tree over two weeks	32
5.7	Distance of the prediction from the ground truth	32
A.1	Screenshot of questionnaire	38
B.1	Screenshot of the Expo Go QR code used for distribution	40

List of Tables

2.1 Reflection Barriers [31].	5
A.1 Poster fair raw data	39



List of Equations

2.1 Gini impurity.	7
----------------------------	---

Ethics Statement

This project fits within the scope of ethics application 0026, as reviewed by my supervisor, Dr. Jon Bird.

Supporting Technologies

- I used React Native and Expo to develop my application.
- I used Expo Go to debug and distribute my application.
- I used Firebase for user authentication and to collect user analytics.
- I used SciKit-Learn's `DecisionTreeClassifier` class to support my initial decision tree experiments.
- I used Graphviz to generate a visual representation of my decision tree.
- I used NPM to publish my decision tree, which can be obtained at 

- I used the `WheelPicker` React Native component when iterating over UI design ideas, obtained from <https://www.npmjs.com/package/react-native-wheel-picker>.
- I used the `InputSpinner` React Native component when iterating over UI design ideas, obtained from <https://www.npmjs.com/package/react-native-input-spinner>
- I used the Expo Speech API, to include text-to-speech functionality within the application.

Notation and Acronyms

QS : Quantified Self
PI : Personal Informatics
UI : User Interface

Chapter 1

Introduction

The idea of self-tracking data is not a new one. It predates the invention of all digital devices even. For example, we have been tracking our weight with the use of weight scales for over a century. The collection of this sort of data allows us to develop self-insight and learn about ourselves. Having this understanding makes identifying harmful behaviour much easier, and gives us the necessary tools to make healthier choices. Although the idea of self-tracking data is not a new one, the technological revolution birthed a new philosophy called the Quantified Self, which has the aim of gaining this self-insight through the use of technological devices. To help reach the goal of the Quantified Self [23], a new class of technological device was created: Personal Informatics systems. The purpose of these systems is to streamline the processes that must occur to reach the stage of self-insight, ultimately resulting in an environment that allows for self-improvement.

A widely adopted model for personal informatics systems is the stage-based model proposed by Li et al. (2010) [31]. They propose that there are 5 stages that must occur for a personal informatics system to successfully function; *preparation*, *collection*, *integration*, *reflection* and *action*. It is important to understand that a key property of the stage-based model is that barriers in one stage cascade into the next. Considering that any effort to self-improve would occur during the action stage, the importance of reflection becomes transparent. If somebody is unable to reflect, then they will be unable to reach the action stage and therefore will not be able to self-improve.

The de facto solution for the reflection stage of the stage based model is to present the user's data in graphs, allowing the user to find patterns and develop self-insight on their own. However, studies have found that this solution is very ineffective. The CHIME app for example, which was designed to support self-care and living well with HIV through tracking and reflecting on personal data, employed this exact solution. The creators of the CHIME app, INTUIT, found that their reflection tool went widely unused [10]. This was a result of its difficulty of use. When you combine this with the fact that 21% of U.S. adults lack the literacy required to complete tasks that require comparing and contrasting information [33], and one third of people in the U.S. and Germany have problems understanding numerically presented information and standard visual displays [21], it is no surprise that this graph based solution for reflection is not effective at all.

In their paper, Li et al. list the most common barriers for reflection. As expected, visualisation and interpretation are listed, however lack of time is also listed as the most common barrier for reflection. This meant that when exploring solutions for a reflection tool, not only did I want something that could make and explain the inferences that people find difficult to make with the current standard, but also do it quickly and conveniently, and preferably on their phone.

Decision trees are a supervised machine learning algorithm used for classification and regression. They are a white-box model which means that every step taken to reach a classification can be extracted and converted into human readable rules. These rules address the problem faced when not being able to interpret and find patterns in data, as the algorithm can make predictions and give explanations. They are also incredibly light-weight and do not require much data to make accurate predictions, meaning that computation can be computed entirely on a mobile device, and all data can be kept localised.

I hypothesise that the inclusion of a decision tree as a tool for reflection within a personal informatics system can address the problems currently faced by the other solutions currently available. The decision tree can make predictions based on a users tracked data, provide an explanation and also provide functionality to explore the consequences of changing their habits. In section 2.5 we outline the levels of reflection proposed by Fleck and Fitzpatrick [20], which range from the lowest, R0, to the highest, R4.

The combination of a classification accompanied by an explanation, with the ability to explore different combinations of data quickly to explore relationships, means that this solution is considered an R2 level of reflection.

The concept of building and running a decision tree on a mobile device is novel and has not been tried before. The objective of this project is to see whether or not we can exploit the decision tree classifier's ability to analyse categorical and numerical data, paired with their low computational and data requirements, to facilitate reflection and address the barriers met by the current standard for personal informatics systems. The contributions of this thesis are:

1. The creation of a decision tree classifier that is able to build and run on mobile devices.
2. The development of a personal informatics app that enables people to track personal life style data and their mood
3. The development and evaluation of a reflection tool that is driven by a decision tree classifier and incorporated in to the personal informatics app
4. A short study that determined the feasibility of the app as a tool for self-reflection.
5. A three week, in the wild study, that demonstrated that users are able to self-reflect, and in some cases make positive changes to their behaviour, by using the app.

Chapter 2

Background

2.1 Introduction

This chapter aims to provide a contextual background for the use of decision trees in a personal informatics application for the purpose of facilitating reflection by users on their self-tracked data. It will give an overview of the field of personal informatics and self-tracking, focusing on the importance of reflection for gaining self-insight. It will discuss the barriers commonly encountered by current personal informatics systems. Additionally, it will explain what is meant by reflection and how it can be effectively implemented into a personal informatics system. It will also touch upon the design decisions made regarding platform and privacy. It will examine the potential of decision trees for facilitating user reflection and some techniques that can be utilised to boost the user experience. Finally, it will cover the main technologies used for this project.

2.2 Self-tracking & Personal Informatics

Understanding why people track data about themselves is crucial to understanding the importance of facilitating reflection on those data. Self-tracking is not a new concept, in fact, Lupton (2016) states that “People have been recording their habits and health-related metrics for millennia, as part of attempts at self-reflection and self-improvement” [32]. This idea can be drawn all the way back to ancient Greece where the words *know thyself* were inscribed on the temple of Apollo, highlighting the importance for the search of self-understanding [48]. Here, it can be established that people self-track their data as a means of having the necessary information required to gain self-insight, and aim towards a state of self-understanding so that they may change their behaviour for the better, if necessary.

This project is concerned with the use of external tools to help measure and collect this personally relevant information, a concept that dates back over a century ago. For example, since the invention of the weight scale in the late 19th century, people have had access to the necessary information to find patterns between their behaviour and weight change [9]. More recently, self-tracking has taken a technological trajectory. From as early as the 1970s, people have been using computational devices and wearable sensors to aid themselves in their self-tracking journey [44]. This idea of self-tracking with the help of technology birthed a new philosophy called the **Quantified Self** (QS), which simply put, is the philosophy of gaining self-insight through using technology and self-tracking devices [23]. And this is not some frivolous technological idea either. Swan (2013) reports that “The quantified self is starting to be a mainstream phenomenon as 60% of U.S. adults are currently tracking their weight, diet, or exercise routine, and 33% are monitoring other factors such as blood sugar, blood pressure, headaches, or sleep patterns” [52].

With the dawn of this new philosophical movement came a class of tool coined **Personal Informatics** (PI) systems. Personal informatics systems are designed to help people track data, reflect on it and act on their insights [28]. They are a core tool for the Quantified Self’s philosophy, and they aim to facilitate the same goal of self-understanding and self-improvement that the ancient Greeks had back in 400BC.

2.3 Self-improvement and the importance of Reflection

Now that it is understood that people self-track data for the purpose of self-improvement, it is important to understand the crucial role that reflection plays in this process. Reflection is a mandatory step that lies between self-tracking and self-improvement. Li et al. (2010) have proposed a widely accepted framework for the successful implementation of personal informatics systems [31]. They categorise the process of self-improvement through self-tracking into stages, with reflection being the penultimate stage. Figure 2.1 below illustrates the stage-based model proposed by Li et al.

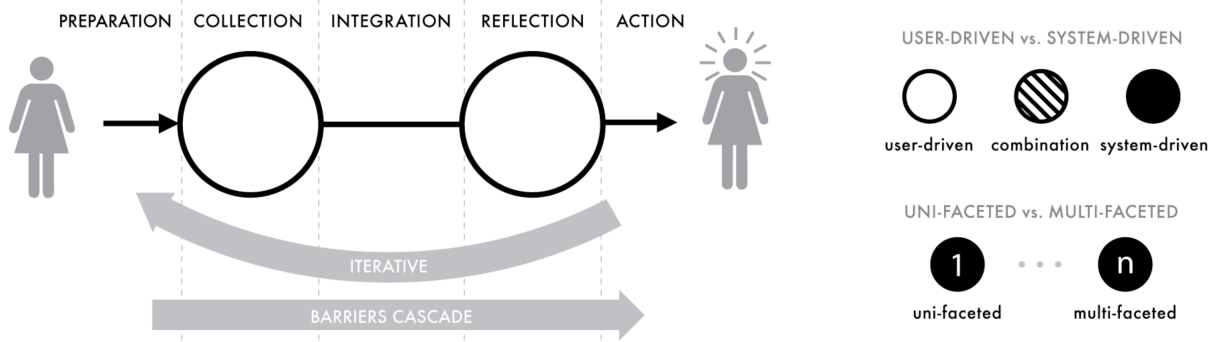


Figure 2.1: The Stage-Based Model of Personal Informatics Systems and its four properties: 1) barriers in a stage cascade to later stages; 2) stages are iterative; 3) stages are user- and/or system-driven, and 4) uni- or multi-faceted. The visuals for 3) and 4) can be used to show these properties for a particular system. [31]

Considering that the intended outcome of self-improvement happens during the action stage of the stage-based model, it is crucial to also understand a key point highlighted by Li et al. (2010), “An important property of personal informatics systems that the stage-based model reveals is that barriers cascade, i.e., problems in earlier stages affect the later stages...Problems in the Reflection stage, such as having trouble using visualizations effectively, prevent users from transitioning to the Action stage” [31]. This is saying, that for a user of a personal informatics system to progress through the stages, they must successfully complete the previous stage. Therefore, if a user is unable to successfully reflect on their data, they will not be able to progress to the action stage where they are able to self-improve. This justifies the importance of reflection when designing a personal informatics system.

2.4 Why Personal Informatics systems need to change

The CHIME app is a personal informatics application, designed to support self-care and living well with HIV through tracking and reflecting on personal data. The built-in tool designed to facilitate reflection makes use of graphs, and presents the user data visually. This is the standard approach taken when designing a reflection tool for a personal informatics system. The aim of this reflection tool is to help people find patterns and draw conclusions about their data by themselves. The app was developed as part of the INTUIT project and researchers found that the reflection tool went largely unused [10]. Davies (2022) completed his own study using the CHIME app. When he asked his participants about the reflection tool, and whether they had used the graphs to visualise their data, one participant said that they had, but only once. Their reasoning for only using the reflection tool once was, “because it’s quite a convoluted system I guess” [10]. Figure 2.2 shows an example of the graph based reflection tool built-in to the CHIME app.

It is perhaps no surprise that a graph based reflection tool does not facilitate reflection when you consider that 21% of U.S. adults lack the literacy skills required for comparing and contrasting information, and making low-level inferences [33]. Or the approximate one third of U.S. and German adults who lack graph literacy skills [21]. These findings also align with the barriers for reflection observed by Li et al. (2010), where issues with visualisation and interpretation were commonly encountered [31]. Interestingly, they found the most common barrier reflection to be a lack of time. The complete list of barriers is shown in table 2.1. By now, the need to lean away from using graphs with the expectation of having users complete lengthy and difficult pattern finding activities, should be obvious. Instead, there is a need for a redesign of a personal informatics system’s reflection tool, with the automation of the difficult, time

consuming process expected of users, kept in mind.

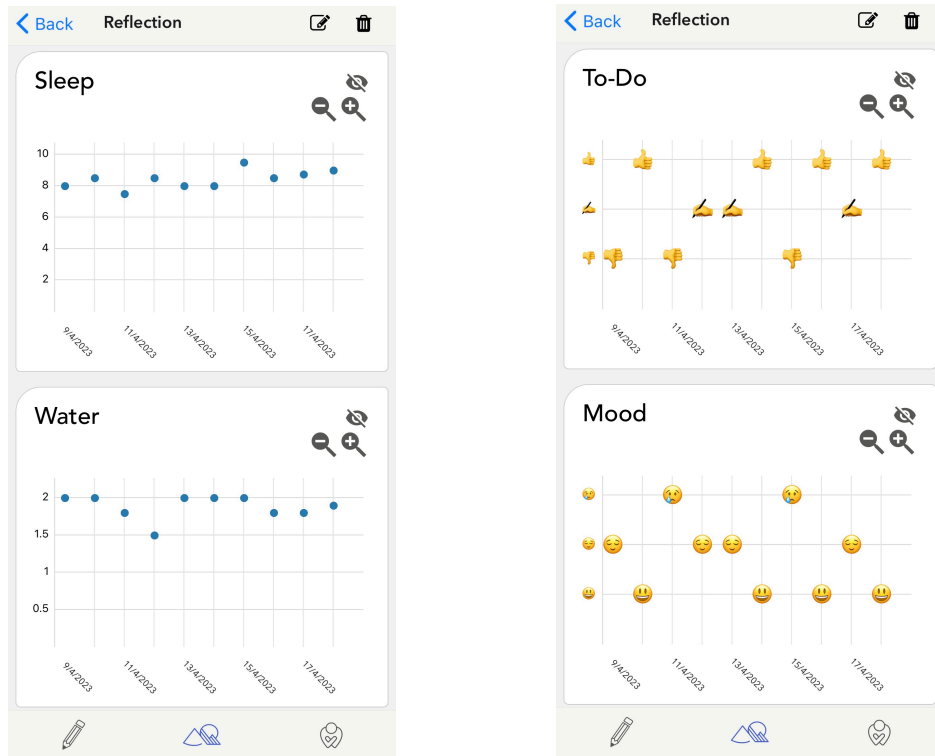


Figure 2.2: Original reflection tool built-in to the CHIME app.

Reflection Barriers	Example Quote
Lack of time (10/68)	"Having time to go through everything, but that is also one of my biggest pleasures is finding that time." P19
Visualization (6/68)	"It's hard to get a holistic view of the data since the time filters are at most one month and I'd like to look at several months at once." P48
Self-criticism (5/68)	"It's extremely difficult (psychologically) to look back on my earliest journals. Much of that information is very emotional and innocent." P12
Interpretation (5/68)	"Sometimes its very difficult to interpret the media" P54
Search (4/68)	"not too tough. sometimes have to wait while search occurs... but it's a couple minutes at most" P14
No context (3/68)	"Not having an overlay of changes in circumstance" P11
Sparse data (3/68)	"Not enough; My collection of data has been intermittent enough that I don't get good time series." P44
Data is not useful (3/68)	"it's really not very useful and it's kind of annoying. I mean, I walk a lot. What else do I really want to know?" P22

Table 2.1: Reflection Barriers [31].

2.5 What's considered reflection and how can we implement this

Before being able to design a tool that facilitates reflection, it is necessary to understand the process of reflection in more detail. Fleck and Fitzpatrick suggest that reflection can be categorised into different levels, starting at R0 and ending at R4 [20]. A higher level indicates deeper reflection than lower levels. The first level, R0, describes a situation where somebody is able to give a statement about their data, without any elaboration. This level is not considered to be reflective. A very low level of reflection is facilitated in R1, and is where a user is able to describe and explain their data. This project will aim to address the requirements for an R2 level of reflection. This is where a user can not only give

a statement and explanation about their data, but also explore the relationships between variables and consider hypothetical ‘what if’ situations related to their data.

For the successful automation of an R2 level of reflection within a personal informatics system, a tool will need to be designed that meets the following criteria: 1) give a statement about the relationships between self-tracked data; 2) explain the reasoning behind the stated relationship; and 3) facilitate users exploring relationships between their self tracked data. Meeting the first criterion suggests the use of a machine learning method capable of classification. Supervised classification models are used to make a prediction on the class or *label* of some input data based on its features, after being trained on past data [5]. More specifically, to meet the second criterion, a white-box classification approach is required. White-box machine learning approaches refer to algorithms that can be easily interpreted and understood by humans. These approaches allow for the extraction of rules from the model to explain the classification process [47]. Finally, to meet the third criterion, the white-box machine learning approach will need to be able to make quick predictions in conjunction with a user interface that allows for the experimentation of different data values, i.e. it needs to require minimal computational resources, particularly if it is running on a mobile device. All of these requirements point to using **Decision Trees**, a popular white-box machine learning approach that allow for the extraction of simple classification rules [30].

2.6 Decision Trees

Decision trees are a type of supervised machine learning algorithm that is widely used for classification and prediction tasks. Decision trees recursively partition the feature space based on the values of the input features and create a tree-like structure to represent the decision-making process. The resulting tree can be used for making predictions about the outcome variable based on the input features [26]. Figure 2.3 gives a visual example of what a decision tree might look like.

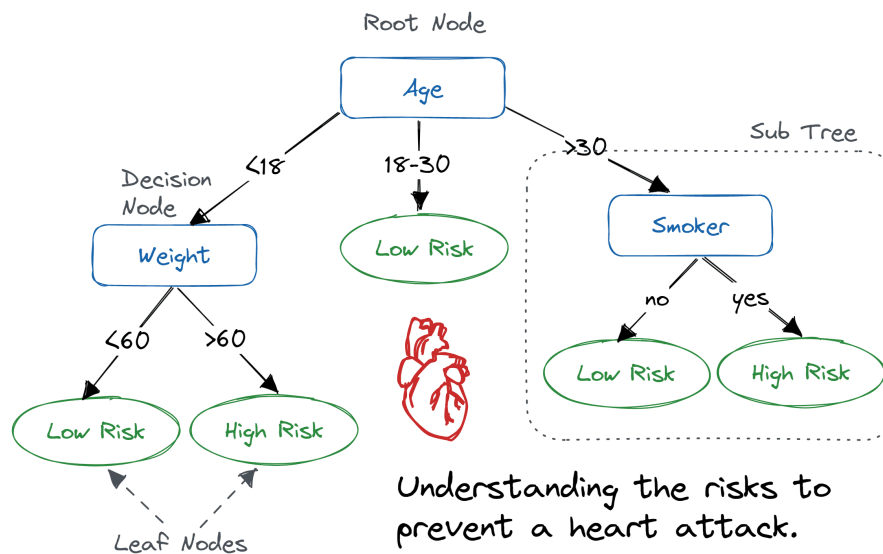


Figure 2.3: Illustration of a Decision Tree [36].

2.6.1 Building a Decision Tree

Programmatically, the tree structure of a decision tree is stored as a set of nodes, each of which is either a decision node or a leaf node. A decision node contains a binary question about a specific feature, and has two child nodes, one for the true answer and one for the false answer. A leaf node contains a predicted class label. The root of the decision tree is always a decision node, except in two cases: when all instances in the dataset belong to the same class, or when the dataset contains no features, only labels. In the first instance, the root of the decision tree becomes a leaf node of the class label. And in the second instance, the root node becomes the class label of the majority class in the featureless dataset [53].

To construct a decision tree, a build function is called using the input dataset and a list of feature names. The build function selects the feature and threshold that minimises the Gini impurity of a node,

a measure of the probability of misclassifying a randomly chosen element from the set. The formula for Gini impurity is shown below in equation 2.1, where $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node. [22]. The discovery of the best feature and threshold is done greedily, by iterating over every possible question and calculating the smallest Gini score.

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2 \quad (2.1)$$

Equation 2.1: Gini impurity.

The algorithm then uses the selected feature and threshold to create a binary question for the decision node, and partitions the dataset into two subsets based on the answer to the question. The process is repeated recursively for each subset until a stopping criterion is met. The recursive nature of the decision tree's build function is shown in algorithm 1

Algorithm 1 Algorithm of a decision tree's build function.

```

1: function BUILDTREE(data, features)
2:   gini, question  $\leftarrow$  FINDBESTSPLIT(data, features)
3:   if gini = 0 then
4:     return LEAF(data.class)
5:   end if
6:   trueSplit, falseSplit  $\leftarrow$  PARTITION(data, question)
7:   trueBranch  $\leftarrow$  BUILDTREE(trueSplit, features)
8:   falseBranch  $\leftarrow$  BUILDTREE(falseSplit, features)
9:   return DECISIONNODE(question, trueBranch, falseBranch)
10: end function

```

2.6.2 Use Cases of Decision Trees

Decision trees have a wide range of applications across various industries, including healthcare. Decision trees have been used in healthcare to support clinical decision-making, diagnose diseases, and predict patient outcomes [42] [6]. Decision trees are popular because they are easy to interpret and can handle both categorical and numerical variables. Overall, decision trees are a flexible and powerful tool that can be applied to a wide range of problems.

2.7 Choosing a platform and addressing privacy concerns

Accessibility, which we define as the usability of a tool for as many people as possible, is an important topic to consider when designing a tool that facilitates reflection. If a user is unable to access the tool in the first place, then the tool serves no purpose. Additionally, there is increased sensitivity when designing a tool that makes use of a machine learning approach, especially when it comes to health data where privacy concerns are of utmost importance.

The lightweight nature of decision trees offer a machine learning model that can run on various devices, including mobile phones, tablets, and laptops. Mobile devices in particular are the most accessible, particularly in low-income settings, where they can help bridge the gap in healthcare delivery by providing access to health information and resources, demonstrated in the study by Bird et al. (2013) [7].

Studies have found that people feel uncomfortable sharing their data, especially with developers and corporations. This discomfort is elevated when the data in question is about their health [24] [29]. One of the advantages of using decision trees is that they can keep the data on personal devices, eliminating the need to transmit sensitive health data to external servers. This approach can alleviate users' privacy concerns, particularly when compared to other machine learning models that require transmitting data to remote servers for analysis.

Not only is it important to consider the users' opinion when making choices surrounding the design of a tool that facilitates reflection, but also the legal ramifications that can occur when handling data of this sensitive nature. The European Union's General Data Protection Regulation (GDPR) imposes strict regulations on the use and sharing of health data, requiring explicit consent from the data owner

before sharing [1]. Failure to adhere to these regulations are highlighted by the period tracking personal informatics tool, Flo, that was used in court during the landmark *Roe v. Wade* Supreme Court case [34]. Flo violated users' privacy by sharing data with third parties through software development kits incorporated into its app, despite stating that it would not be sharing data. By utilising the ability of a decision tree approach to keep data locally, the risk of violating such regulations is minimised.

2.8 Utilizing Salience to Encourage Reflection

In a personal informatics system, the reflection tool can often be just one small part of the larger system. This is because reflection is just one of the stages that a personal informatics system must facilitate. This can make it easy for users to overlook or ignore the tool for reflection. One potential solution to this problem is to make the reflection tool more salient, or attention-grabbing. Salience is the degree to which an object or feature stands out and captures attention [8]. Research has shown that making certain features of a product or system more salient can have a significant impact on user behavior. For example, studies on food labels found that using traffic light colors to indicate the healthiness of different foods made it easier for people to make healthier choices [49] [46].

In the context of personal informatics, salience could be used to draw users' attention to the reflection tool and encourage them to use it more regularly. One approach could be to use visual cues such as icons or colors to make the reflection tool more prominent within the app. By making the reflection tool more salient, we can ensure that users are more likely to engage with it to achieve the intended goal of self-reflection.

2.9 Summary

In summary, this chapter explained the purpose of self-tracking data, namely self-insight and self-improvement. It introduced the concept of personal informatics systems as a tool to aid with the journey towards self-improvement, and identified the importance that self-reflection has along that journey. It then highlighted that the current tools used by personal informatics systems for reflection are highly ineffective. We additionally outlined a set of criteria which we deem necessary for facilitating reflection in personal informatics systems and postulate that decision trees are an approach capable of meeting all of these criteria. Furthermore, it gave context for the decisions surrounding platform choice and privacy. It also explored the idea of salience, and ways that it can be utilised to encourage reflection.

Chapter 3

Technological background

In this brief chapter we provide an overview of the main programming languages, packages, and frameworks used in the execution of the project. We use this overview to motivate the rationale behind the selection of each tool and to justify how they contribute to the effective implementation of our project.

3.1 JavaScript, React Native & Expo Go

JavaScript is a popular programming language, used primarily for both client-side and server-side web development. According to the Stack Overflow Developer Survey 2022, JavaScript is the most commonly used programming language by all respondents for the tenth year in a row, with 65.36% of respondents reporting to have used it in the past year [40]. This popularity can be explained partly by the fact that all modern web browsers support it [27]. The popularity of JavaScript has resulted in many libraries, frameworks, and tools being available for developers to use and contribute to. For example, NPM (short for Node Package Manager), a popular registry for JavaScript packages, has over 3 million JavaScript packages at the time of writing, of all sorts of functionality, available for download by developers [38].

One such library, popular amongst JavaScript developers, is React. React is an open-source JavaScript library for building user interfaces, developed by Facebook [16]. It also happens to be the most popular front-end JavaScript library [2]. React is based on the idea of reusable components, which can be composed together to create complex UIs. It is often used in combination with other JavaScript libraries and frameworks, such as Redux and React Native. The latter, React Native, builds on top of React and extends its capabilities to allow for the development of native mobile applications [17]. One of the main advantages of React Native is its ability to use a single codebase for both iOS and Android platforms, which can significantly reduce development time and costs. In terms of popularity, React Native is just behind React, being the third most popular front-end library [2].

Expo Go is a companion app for React Native that enables developers to quickly and easily test and preview their React Native applications on a mobile device [15]. From a developers perspective, Expo Go is incredibly useful as it allows for dynamic development of React Native applications. It is as simple as downloading the Expo Go app and scanning a QR code that is generated by the expo framework. Changes made to the codebase appear automatically within the Expo Go app, avoiding the need to constantly recompile. Expo Go is available for both iOS and Android platforms and can be downloaded for free from the App Store and Google Play. One of the benefits of using Expo Go is that it eliminates the need for complex build and deployment processes, making it easy for developers to share their work with others.

Another benefit of using JavaScript, React Native and Expo Go, unique to this project, was the fact that INTUIT's CHIME app was built using it. Given that this project had access to the CHIME app, it mitigated the need to develop the tracking functionality of a personal informatics system, maximising the time spent on the task of addressing the problems encountered by reflection tools.

3.2 Firebase

Firebase is a popular mobile and web application development platform, used by millions [19]. Acquired by Google in 2014, it offers a variety of tools and services to help developers build high-quality applications [18]. Of particular use to this project are the authentication and analytics features. Firebase

Authentication offers a robust authentication system that comes with functionality to create and authenticate user accounts, directly from an application. Firebase Analytics provides developers with the means necessary to track user behaviour and app performance, even having the ability to track custom metrics. This results in Firebase being an ideal platform to create a secure application for the user, and an observable application for the developer.

Chapter 4

Project Execution

4.1 Introduction

This chapter provides a comprehensive overview of the steps taken to create the decision tree-based mobile application. We cover the development process of the decision tree algorithm, and include experiments conducted using our own algorithm and pre-existing libraries. We then talk about the integration process of the decision tree algorithm into a mobile environment, and test the performance in this setting. Additionally, we explore the UI choices made to best facilitate reflection in the CHIME app. Finally, we describe any extra features implemented with the purpose of encouraging reflection.

4.2 Getting Familiar with Decision Trees

Prior to the development of our own decision tree classifier, we conducted initial experiments with existing models to garner a deeper understanding of their strengths and limitations. We used these studies to investigate the data requirements for a decision tree to begin making accurate predictions.

Using the implementation from SciKit-Learn [41], figure 4.1 depicts the ability of the model to correctly classify instances of both the Iris [13] and Pima [37] datasets, both of which are commonly used in the testing of models to compare industry standards. We show the accuracies they attained based on varying the size of the training datasets used in each case. The method for obtaining these accuracy scores is explored further in Avinash Navlani’s article *Decision Tree Classification in Python Tutorial* [36]. An interesting observation, that aided in our decision making later on in development, was that the biggest increase in classification accuracy for both datasets, happened while increasing the training dataset size from 1 to 10 entries.

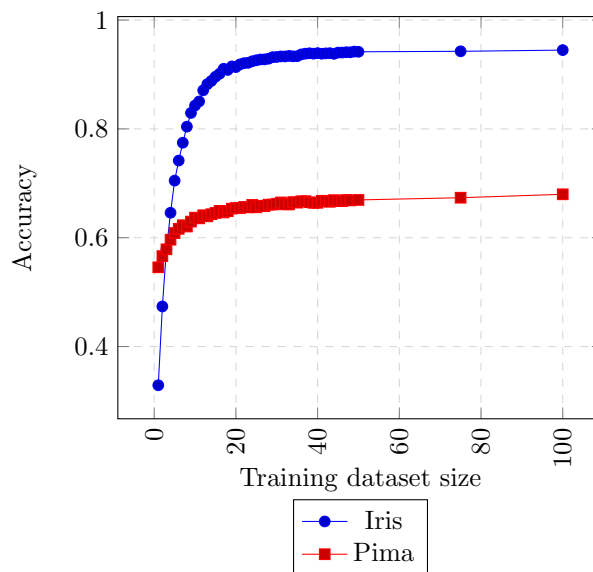


Figure 4.1: Accuracy of classifier for varying test dataset size.

4.3 Developing a JavaScript Decision Tree

In section 2.6 we covered the primary functionality for building a decision tree. The construction of a tree, programmatically, utilises a recursive build function that aims to minimise the Gini impurity at each node, by partitioning the data in a certain way.

The process of developing our own JavaScript decision tree was made easier after watching a Google developer write their own decision tree using Python [12]. Out of all of the key methods and helper functions that we developed for the JavaScript decision tree, it is important to draw attention to the classification function 4.1. This function serves as the foundation for extracting the mood prediction, and eventually the classification rules, used to drive this project.

```
1 function classify(row, node) {  
2   if (node instanceof Leaf) {  
3     return node.predictions  
4   }  
5  
6   if (node.question.match(row)) {  
7     return classify(row, node.true_branch)  
8   }  
9   else {  
10    return classify(row, node.false_branch)  
11  }  
12 }
```

Listing 4.1: Function to classify data.

4.3.1 Extracting Rules

One of the big selling points of using the decision tree approach was the ability to extract human-readable rules explaining the classification process. Currently the classification function traverses the tree and matches the input data to each decision node, to predict the class label. However, it does not keep track of this process. Piotr Płoński (2021) wrote an article describing methods of implementing this extraction process [43]. His implementation for extracting human-readable rules, however, extracted rules for the entire tree, and not one class prediction. This results in rules of the form, “If this...and then this... then predict this”.

For this project, we wanted rules of the form “This was predicted, because this...and then this”. Our method for rule extraction was loosely based off of Płoński’s description, but instead of creating a separate rule extraction method, we included the functionality into the classification function. The implementation was simple (Listing 4.2), an unpopulated list is passed down a recursive function. Each decision node that is encountered, the stored question gets added to the list of rules. Once a Leaf node is reached, the classification function returns, now with a list of rules for the classification (Listing 4.3).

```
1 function classify(row, node) {
2   var rules = []
3
4   function helper(row, node, rules) {
5     if (node instanceof Leaf) {
6       return node.predictions;
7     }
8
9     if (node.question.match(row)) {
10      rules.push(node.question.printTrue());
11      return helper(row, node.trueBranch, rules);
12    } else {
13      rules.push(node.question.printFalse());
14      return helper(row, node.falseBranch, rules);
15    }
16  }
17
18  var classified = helper(row, node, rules);
19
20  return { classified: classified, rules: rules };
21 }
```

Listing 4.2: Function to classify and extract rules for the classification.

```
1 {
2   classified: { 'Iris-virginica': '100%' },
3   rules: [
4     'petalWidth is greater than or equal to 1',
5     'petalWidth is greater than or equal to 1.8',
6     'petalLength is greater than or equal to 4.9'
7   ]
8 }
```

Listing 4.3: Output of classification function, with rule extraction.

4.3.2 Improving the Rule Extraction

A big problem with this implementation is that it does not account for the stacking of rules that are based on the same feature. That is, the fact that the question of decision nodes deeper in the tree may be created from the same feature. Listing 4.3 is a good example of this. The root node and the second node traversed, reference the same feature. When converting this into a human-readable sentence, it makes more sense to say, “Because petalWidth is greater than or equal to 1.8 and petalLength is greater than or equal to 4.9” than it does to say, “Because petalWidth is greater than or equal to 1, petalWidth is greater than or equal to 1.8 and petalLength is greater than or equal to 4.9”.

We solved this problem by using maps as an intermediary step, when tracking the rules (Listing 4.4). Instead of adding the rules directly to a list, we set the map value belonging to that feature, with the threshold value. This results in rules which overlap the same feature as a prior rule, overwriting the previous threshold value. The map values are then able to be converted into a list (Listing 4.5).


```

1 function classify(row, node) {
2     var ruleMap = new Map()
3
4     function helper(row, node, ruleMap) {
5         if (node instanceof Leaf) {
6             return node.predictions
7         }
8
9         if (node.question.match(row)) {
10             ruleMap.set(node.question.printTrueKey(),
11                         node.question.printTrue())
12
13             return helper(row, node.trueBranch, map)
14         }
15         else {
16             ruleMap.set(node.question.printFalseKey(),
17                         node.question.printFalse())
18
19             return helper(row, node.falseBranch, map)
20         }
21     }
22
23     var classification = helper(row, node, ruleMap)
24
25     return {classification: classification,
26           rules: [...ruleMap.values()]}
27 }

```

Listing 4.4: Function to classify with improved rule extraction.

```

1 {
2   classified: { 'Iris-virginica': '100%' },
3   rules: [
4     'petalWidth is greater than or equal to 1.8',
5     'petalLength is greater than or equal to 4.9'
6   ]
7 }

```

Listing 4.5: Output of classification function, with improved rule extraction.

4.3.3 Publishing our Decision Tree

As well our purpose of using decision trees as a tool to facilitate self-reflection, decision trees also have many other uses. It is because of this that we decided to publish our decision tree model on the NPM registry. This allowed us to contribute to the open-source nature of the JavaScript community, and now allows others the ability to incorporate an accurate decision tree model into their JavaScript projects.

To publish our library on NPM, we followed Aman Kumar's tutorial "Creating an npm package from React Component" [39]. Here is a link to the library: [https://github.com/aman-kumar/decision-tree](#) for any curious readers.

4.4 Comparing the Decision Tree with SciKit Learn

Before beginning the integration process of the decision tree into a mobile application, we thought that it was best to conduct some experiments. The aim of these experiments was to explore the differences between our decision tree model and the one produced by the SciKit-Learn library. These experiments were conducted using the iris flower dataset.

In SciKit-Learn’s documentation for decision trees, there is a section about exporting their decision trees for visualisation, using Graphviz [51]. Graphviz is an open-source graph visualization software that allows you to create and display diagrams, networks, and graphs in various formats. It provides a set of tools for drawing and manipulating these graphs, as well as algorithms for layout and rendering [25]. Fortunately, SciKit-Learn’s decision tree class comes built-in with the functionality to export using Graphviz. By following the steps listed in the documentation, we managed to export an image of the decision tree built using the SciKit-Learn library, on the iris flowers dataset. Figure 4.2 shows the exported image of the decision tree.

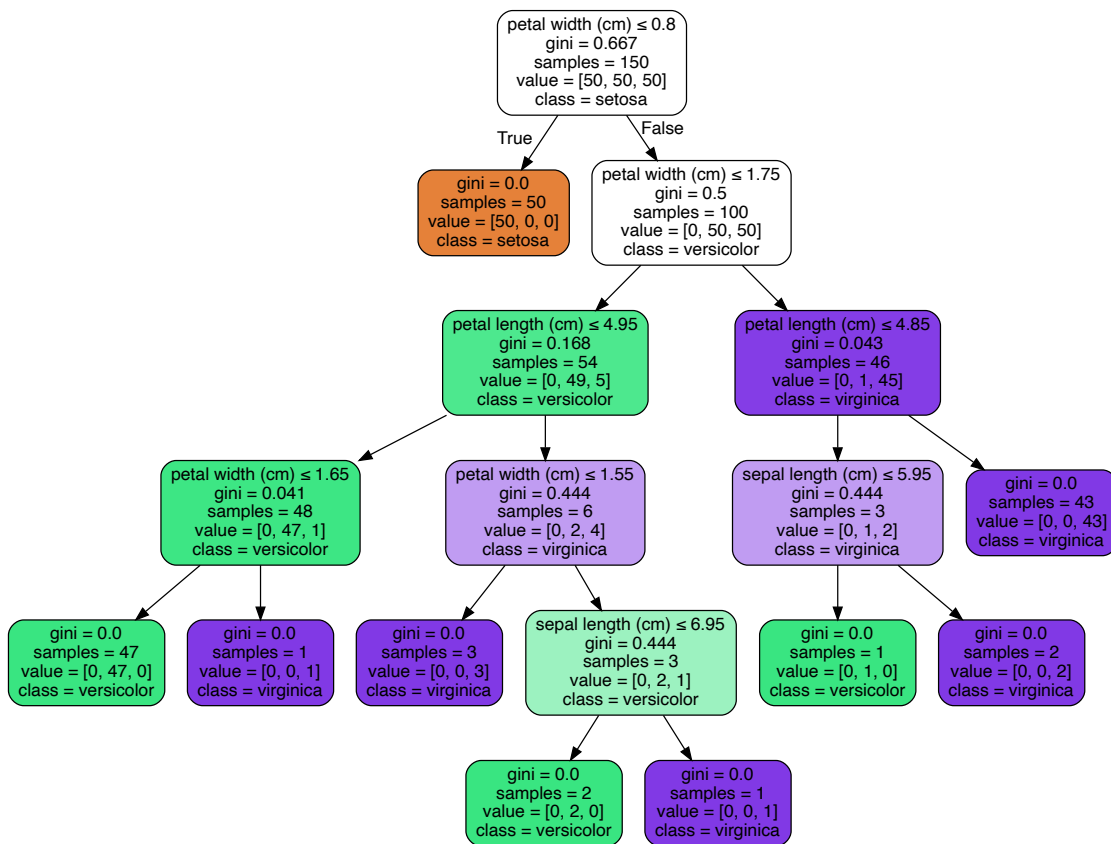


Figure 4.2: Graphviz render of SciKit-Learn’s decision tree.

Our decision tree model, on the other hand, did not have this functionality built-in. To counter this discrepancy, we created a function that would parse our own decision tree, into the same input format that SciKit-Learn does. Due to differences in the programmatic implementations of our model, and SciKit-Learn’s, our resulting Graphviz code was not as complete. For example, our decision tree model only stores class predictions in the leaf nodes of the tree. Therefore, we were only able to display class predictions in the leaf nodes of our Graphviz render. Figure 4.3 shows the exported render of our decision tree.

After a brief glance, it would be safe to assume that both trees are the same, obviously excluding the fact that our decision nodes don’t include class values, nor do we include the value array that is shown in the nodes in figure 4.2. We can be excused for making this assumption, because both trees are of the same depth, and the class labels and partitioned samples appear to be the same for both trees. However, after close inspection, we can see that our decision tree is constructed differently to SciKit-Learn’s. Firstly,

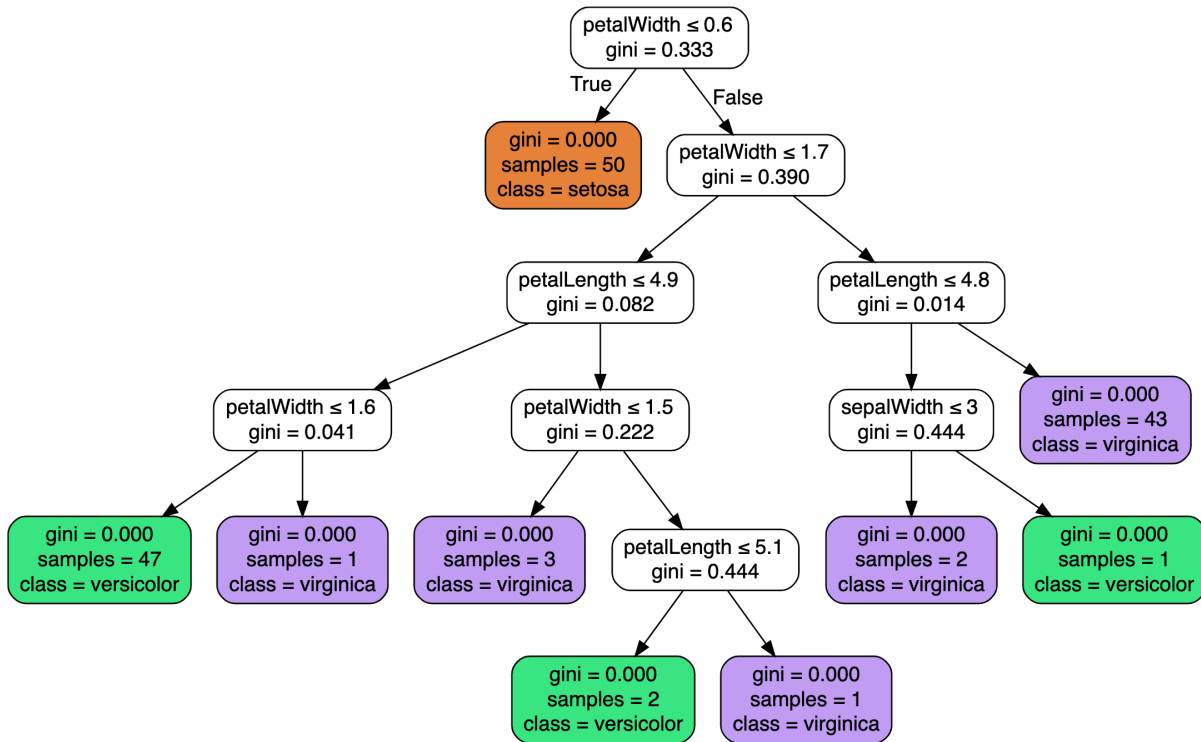


Figure 4.3: Graphviz render of our decision tree.

the question's asked in ours are slightly different. It is also interesting that, if we assume that the root node is of depth 0, the nodes of depth 4, that are furthest right, are switched around in our tree.

Considering that the two trees are different, despite being built from the same data, we thought it would be important to evaluate the effect of these differences on the classification process. We conducted another experiment using the iris dataset, which incorporated the same idea as the previous experiment made in section 4.2. That being, the splitting of the iris dataset into a training dataset and a test dataset. The general idea was to begin with a 1-149 split of training and test data, looping up to a 149-1 split, testing the accuracy of predictions on the test dataset. This was done for both decision tree models. We wanted to see first, whether both trees had the same accuracy despite the differences. Or, if different predictions were made, which tree was the most accurate. Figure 4.4 shows the difference in accuracy between both trees. If both trees were to make the exact same predictions, then the graph would display a straight, horizontal line.

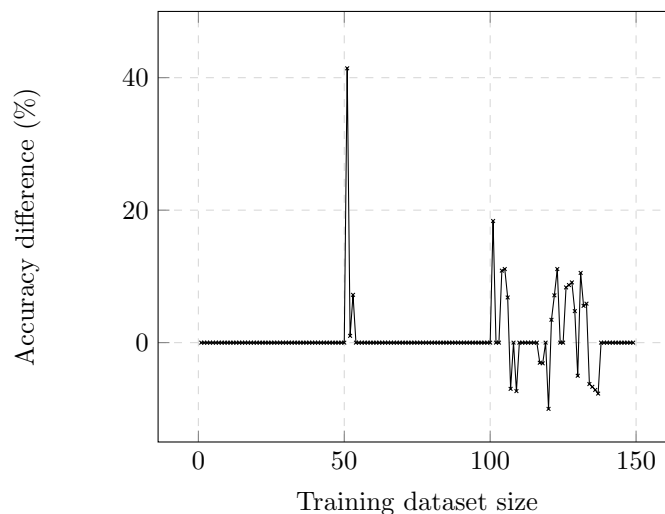


Figure 4.4: Difference in accuracy of our decision tree compared with SciKit-Learn's.

Interestingly, we can see that the differences in the way both trees were built, did in-fact have an effect on the classification process. On the whole, both trees made the same number of accurate predictions. However, on the occasions that the models predicted differently, our model typically resulted in a higher accuracy. Out of the 149 iterations of the loop, the models differed in accuracy 27 times. Our model performed better in 17 of those instances. One of those instances, training dataset of size 51, resulted in our decision tree model having over a 40% increase in accuracy, compared with SciKit-Learn's.

4.5 Mobile Compatibility

For the development of our own decision tree, all code execution was run on a 2019 MacBook Pro. Considering that accessibility is a core goal of the project, it was important to ensure that the decision tree model had a suitable performance on mobile devices.

JavaScript has an API called the **Performance** API that provides access to performance related information and metrics of the device running the code. The *performance.now()* method is part of this API and provides a high-resolution timestamp, typically measured in milliseconds, that can be used to measure the performance of JavaScript code. If necessary, it can represent times as floating-point numbers with up to microsecond precision, allowing for a very high level of precision for performance measuring.

To make use of this API to measure the performance of our decision tree model, on a mobile device, we created an Expo application for the purpose of conducting experiments. We did this by following the Expo documentation on creating an Expo application [14]. Listing 4.6 demonstrates how performance was measured within the Expo application.

```
1 const startTime = performance.now()
2 const decisionTree = new DecisionTree(dataset, features)
3 const endTime = performance.now()
4 console.log(endTime - startTime)
```

Listing 4.6: Measuring Build Time of the Decision Tree.

To get a good understanding of the performance of the decision tree model on a mobile device, we repeated the code in listing 4.6, on an increasing training dataset size. The results of this experiment are shown in figure 4.5. Fortunately, buildtime remained almost instantaneous, with 150 data entries taking just under three hundredths of a second to build.

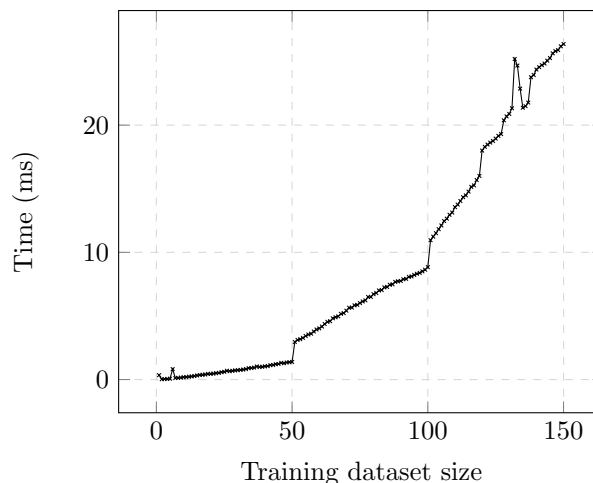


Figure 4.5: Buildtime of decision tree on iPhone 13.

4.6 Tracking data and Presenting Results

When it comes to our mood, it is important to track the mood relevant data, necessary for the decision tree model to make accurate predictions. The general guidance from the NHS lists the following as key, mood influencing factors [35]:

- Sleep
- Diet
- Physical activity
- Social interaction
- Stressors, i.e. workload

As a result, for this project we decided to track sleep, water intake, number of meals, level of exercise, social interaction and whether the user had completed the work necessary of them for the day, as the accompanying data for mood. We also decided to track mood categorically, as either low/OK/good, as these three options have shown to be effective [11]. In the context of personal informatics, data is tracked through an individual tracker. Data is typically either numerical (hours slept), or categorical (low mood, okay mood, good mood).

For our project, we aimed to present the output of our decision tree model, built using the mood-relevant data, in a clear and salient way. The approach that we took was to include a component within the wider context of a personal informatics app that displayed a prediction and explanation, based on the combination of all of the current tracker values.

4.7 Creating a Prediction Card

In this section we talk about the creation of a prediction card component. This prediction card was developed to present the decision tree classification, and extracted rules, in a salient and engaging way in the mobile app. Developing the prediction card as a component allowed us to stay in-keeping with the reusable nature of the React framework.

The prediction card takes the output from the decision tree prediction and presents the classification and rule in an easily digestible format. For added effect, we mapped the mood classification to an emoji, to better convey the prediction. Figure 4.6 is a screenshot of the developed prediction card. As you can see, we also included a speech button. The speech button makes use of the Expo Speech API to read aloud the written prediction. This was a necessity, due to the low levels of literacy mentioned in section 2.4.

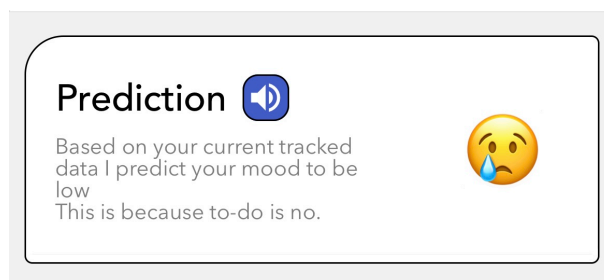


Figure 4.6: Prediction card.

4.7.1 Improving the Readability

A problem with the explanation generated from the extracted rules, was that it wouldn't always make the most sense. You may have noticed already that the extracted rule in figure 4.6, "This is because to-do is no", does not flow as naturally as it could do. We encountered this problem for rules that are based on categorical features. We fixed this inconsistency by mapping the grammatically inconsistent rules, to a more readable version. Figure 4.7 demonstrates the change that this mapping had on the previous rule, now being, "This is because you haven't started your to-do."

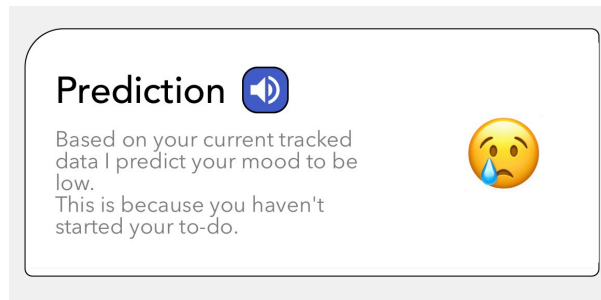


Figure 4.7: Prediction card with improved readability.

4.7.2 Implementing Salient Properties

In section 2.8, we introduced the concept of salience, that is, the extent to which something grabs attention.. We also covered previous research that found that the inclusion of salient, traffic light coloured food labels, resulted in the people making healthier choices when choosing food. Figure 4.8 illustrates the approach we took to make our prediction card more salient. Similarly to the emoji, we mapped the mood prediction to a traffic light colour to use as a border surrounding the prediction card, with the hope that users might make healthier connotations between their behaviour and mood.

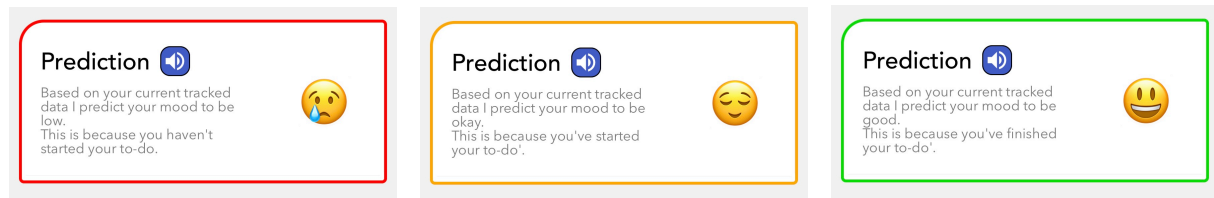


Figure 4.8: Prediction card with traffic-light border.

4.8 Creating a UI for an R2 Level of Reflection

The prediction card gives a prediction for the users mood, and an explanation for that prediction, based on the user's currently tracked data for that day. These two functionalities meet the criteria necessarily for an R1 level of reflection. However, to facilitate an R2 level of reflection, the application must also allow the user to explore the relationship between mood and changes in their behaviour.

To facilitate this higher level of reflection, we came up with the idea of a UI, separate from any tracking functionality, that would load the user's currently tracked data and give them the ability to explore how their mood would change if they changed the values of their currently tracked data. The UI would incorporate the prediction card, and for each tracker, there would be a component that allows the user to easily iterate over different value combinations, while dynamically observing any changes to their mood.

The following subsections list the design iterations that we made when developing the UI to facilitate an R2 level of reflection. We experimented with multiple methods of displaying and changing the tracked values.

4.8.1 Wheel Spinners

Our initial approach was to use wheel spinners, as they offer the ability to change values in one fluid motion. This is a result of their perceived functionality of sliding between values. To implement this design idea, we used the WheelPicker component.

4.8.1.1 Vertical Wheel Spinners

Figure 4.9 shows the first iteration that we tried, using wheel spinners. The idea was to stack each tracker vertically with a respective WheelPicker component. This resulted in the need to scroll back and forth to change behavioural values, and to see the prediction card. Studies have shown that scrolling formats

reduce comprehension of tasks, due to the need to recall information [50], and as a result, this design implementation was deemed ineffective.

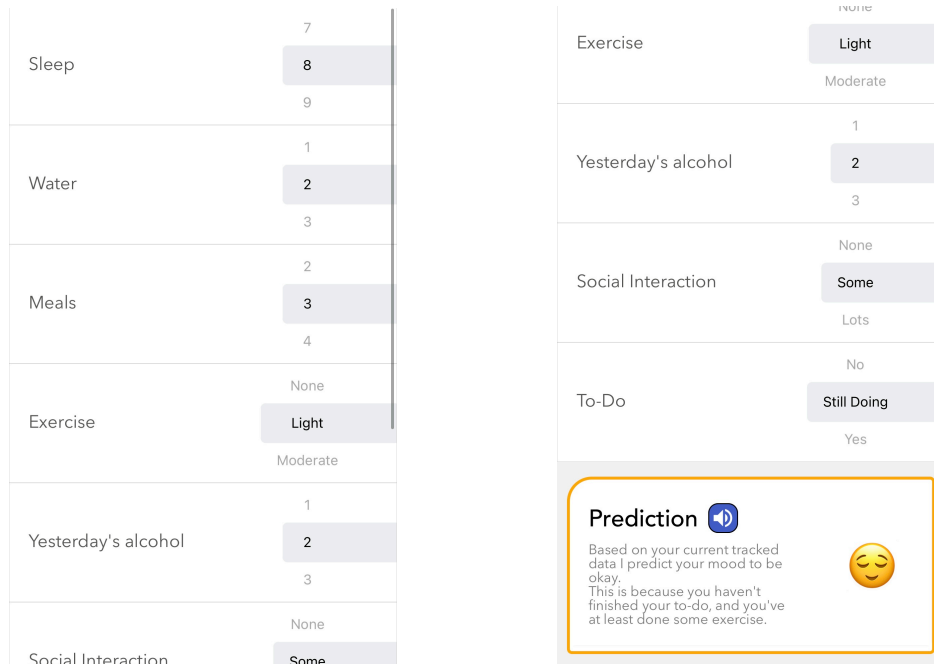


Figure 4.9: Exploration UI using vertically stacked wheel spinners.

4.8.1.2 Horizontal Wheel Spinners

Figure 4.10 shows our second implementation, using wheel spinners, stacking the trackers horizontally. This implementation successfully mitigated the need to scroll to see the prediction card, however the user still needed to scroll to change different tracker values. This ultimately resulted in a less severe version of the previous problem, where the task of behavioural exploration relied on the user recalling the values that they had changed.

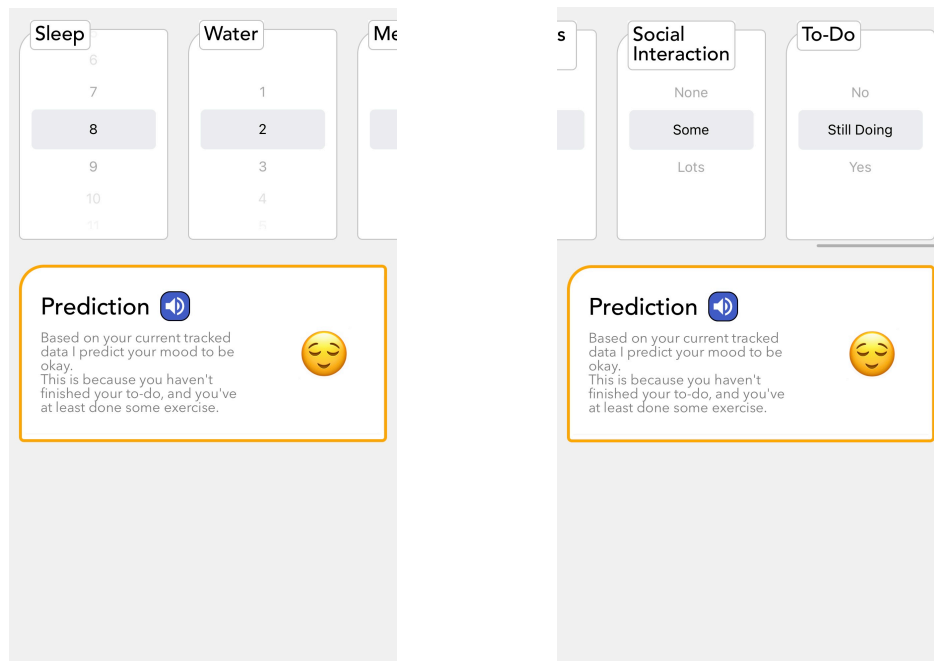


Figure 4.10: Exploration UI using horizontally stacked wheel spinners.

4.8.2 Click Spinners

An alternative approach using wheel spinners, was to use a clickable spinner that can either increase or decrease the value with a each click. At the cost of decreased fluidity when moving between different values, the clickable spinner solves the problem of screen space, encountered by the wheel spinner.

To facilitate this implementation, we made use of the InputSpinner component. The previous, WheelSpinner, component required the us to input the whole selection array as a parameter, which, for numerical input was very tedious. Whereas, the InputSpinner component only took a numerical input, and allowed for the specification of an increase/decrease precision value. This meant that, unlike the WheelSpinner, we would not have to generate an array of hundreds of decimal place values between each integer value. And instead, pass in the tracked value, and a value to increase/decrease it by. The InputSpinner also had the added capability of allowing the user to input a numerical value directly into the component.

However, there was a problem with this component. The InputSpinner component only took numerical input, and our dataset made use of both numerical and categorical data. To remedy this issue, we created a StringSpinner component that replicated the look and feel of the InputSpinner, and that worked on our categorical data. Figure 4.11 shows the UI design implementation using both the InputSpinner component, and our own StringSpinner component. As you can see, all trackers are visible, as well as the prediction card.

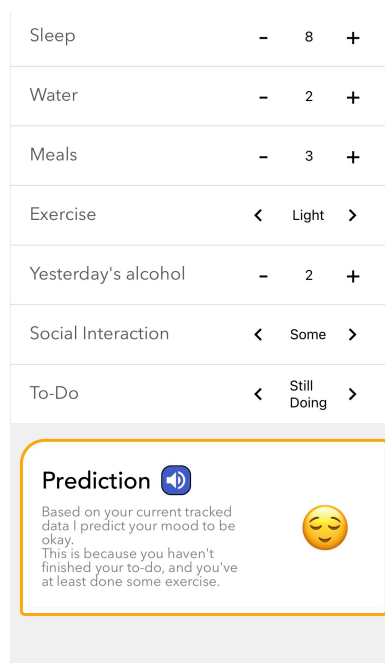


Figure 4.11: Exploration UI using click spinners.

4.9 The CHIME app

The prudent choice of using JavaScript and the other technologies mentioned in the background, resulted in a seamless integration of the decision tree library, and the UI components that had been previously developed, into the CHIME app. Once we had been granted access to the codebase for the CHIME app, the biggest obstacle that we encountered was the outdated Expo SDK version that had been used initially for the CHIME app. At the time of our development, Expo GO only supported Expo SDK versions 46, 47 and 48. The CHIME app, however, was using Expo SDK version 38. This meant that we were unable to run the CHIME app application until we had fully upgraded the codebase to use a supported SDK version. After resolving the issue with Expo by upgrading the codebase to use Expo version 46, transition into the CHIME app was as simple as importing the decision tree package from NPM, and copying the UI components into the codebase.

4.9.1 Incentivising Data Collection and the Use of the Exploration Page

To increase the accuracy of the decision tree model, we blocked the prediction card component from displaying any predictions until the user had successfully input values for all trackers for that day. This ensured that there were no missing values while collecting data.

Also, to maximise the exposure that the user had to the exploration page, we edited the prediction card shown in the tracking screen of the app (Figure 4.12) to force the user onto the exploration page to view their prediction for that day. The user still had the ability to go back to previous days in the data tracking screen, with the prediction card functioning normally, displaying the usual data.

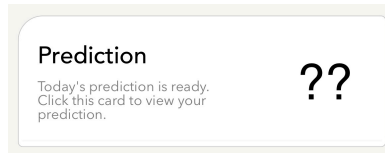


Figure 4.12: Prediction ready card.

4.9.2 Authentication

Due to the sensitive nature of the data collected by the app, we wanted to ensure that no unauthorised access was allowed. Also, for the sake of the study, we wanted to only allow a select few people that had agreed to participate, to use the app. To meet these requirements, we used Firebase's Authentication SDK, which supports account creation and user authentication into the app.

The CHIME app already had authentication through Firebase integrated, however it was making use of a Firebase project that we did not have access to, and also was no longer functioning. Fixing this was a matter of switching the Firebase configuration to our own. This allowed for the fundamental features of account creation and user authentication, through the use of our own Firebase project (Listings 4.8 & 4.9). However, we still had the issue of anybody being able to create an account. One potential fix for this was to keep a local list of verified email addresses that the application could query during the sign-up process. However, this would involve storing the list of our participant's personal emails on every device, a security decision that we weren't prepared to make. Even if we were to disregard this privacy concern, we would still have to re-publish the application with each new verified email address. Firebase comes with another feature, named Cloud Functions. This feature allows for the calling of functions that are stored remotely. Separating the email verification function from the local device, not only eliminated the previous privacy concerns, but also allowed for the addition of new verified emails, without the need to re-publish the application (Listing 4.7).

4.9.3 Analytics

As a part of our three week user study, we planned to collect user analytics to go alongside the user feedback. We wanted to gather these analytics to develop our understanding of how users used the features in the app. For example, we aimed to collect when users tracked their data, their interactions with the prediction card and also explore how they used the UI for reflection. Firebase Analytics used the same Firebase configuration that Firebase Authentication did. Listing 4.10 gives an example for how user events were logged and collected.

```
1 export const validEmail = (email) => {
2   return new Promise((resolve, reject) => {
3     firebase
4       .functions()
5       .httpsCallable("verifyEmail")({ email: email })
6       .then(({ data }) => {
7         if (data) {
8           resolve(data.message);
9         } else {
10          reject(
11            "This is not a valid study email address. Please make sure
you have consented to the study first."
12          );
13        }
14      })
15      .catch((err) => {
16        console.log(err);
17        reject(
18          "This is not a valid study email address. Please make sure you
have consented to the study first."
19        );
20      });
21    });
22  };
```

Listing 4.7: Call to the Cloud Function, to verify an email address.

```
1 await validEmail(email);
2
3 await firebase
4   .auth()
5   .createUserWithEmailAndPassword(
6     email,
7     localState.inputValues.password
8   );
```

Listing 4.8: Account creation with Firebase Authentication & Cloud Functions.

```
1 await firebase
2   .auth()
3   .signInWithEmailAndPassword(email, localState.inputValues.password);
```

Listing 4.9: Account creation with Firebase Authentication & Cloud Functions.

```
1 await Analytics.logEvent(event.name, {})
2   .then(() => {
3     })
4   .catch((err) => {
5     console.log(err);
6     });
7 ;
```

Listing 4.10: Event logging using Firebase Analytics.

4.9.4 Changing the Default Voice

In section 4.7, we highlighted the need to include text-to-speech functionality within the app, to aid any users that had difficulty reading the data presented. One irritating consequence of the way the Expo Speech API was designed is that it is up to the developer to specify the voice that is used. The default voice is very robotic sounding, and we believed if given the choice, users might want the option to change it. The Expo Speech API does not come with built-in voice options, and instead, makes use of voice profiles preinstalled on the device. Considering that each device can have different voice profiles installed, we added a settings page (Figure 4.13) so that the user was able to tweak the voice used by the Expo Speech API, if they wanted a change from the default.

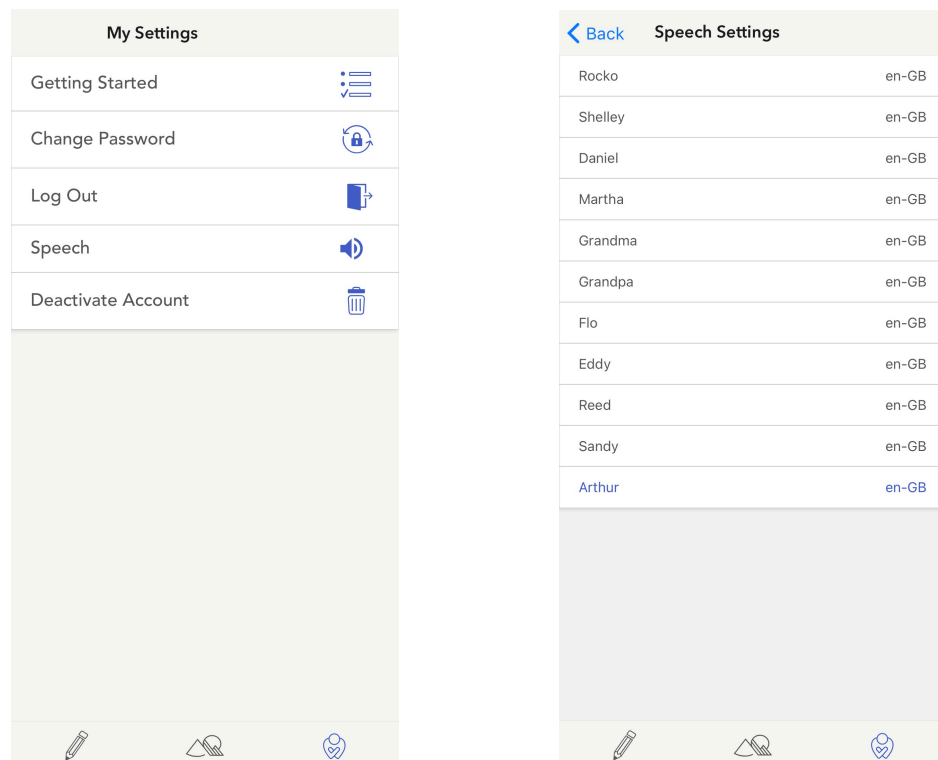


Figure 4.13: Settings page to change the default voice.

4.10 Summary

The execution of this project was a long, meticulous process. The result is a combination of careful planning, in-depth experiments and multiple design iterations. This chapter had a lengthy initial focus on the functionality and development of an accurate decision tree classifier. It then saw the integration of that decision tree into a mobile setting, where multiple design decisions were considered to ensure the proper execution of an effective reflection tool. Finally, it documented the transition from the test mobile environment, into a fully functional personal informatics application, providing justifications for the choice of data tracked.

Chapter 5

Critical Evaluation

5.1 Introduction

In this chapter, we will evaluate the effectiveness of the decision tree-based, mobile personal informatics app. More specifically, its ability to facilitate self-reflection. To aid in this evaluation, two studies were conducted. The goal of these studies was to measure the usefulness, usability and performance of the app, and to also gather user feedback. The first study was carried out during the individual projects poster fair day, with the aim of establishing user opinion on the current standard of using graphs as a means for reflection, compared to the app developed for this dissertation. The results of this study provided valuable insights into user preference when choosing a tool for reflection, with an overwhelming majority of people choosing to use the decision tree-based app over the graphs. In the second study, we conducted a a three week in the wild evaluation [45], with a small group of participants. This longer study allowed us to gather rich data on the app’s usability, user engagement, and its overall effectiveness in facilitating reflection on self-tracked data. In the following sections, we will discuss the methodologies, results, and implications of these two studies, providing a comprehensive evaluation of the effectiveness of using decision trees to facilitate self-reflection in a personal informatics app.

5.2 Poster fair study

During this project, we were given the opportunity to present our work at a poster day. The poster day served as an opportunity for people to get to know the project, and ask any questions that they had. This event was open to everyone, whether they were students of any year/course, academic staff, industry colleagues and even people unfamiliar with the field of computer science. Having access to this stream of people made for the perfect environment to conduct a study on the general population’s opinion on the app developed for this project.

This study aimed to answer, firstly, whether people would choose to use the decision tree-based app over the standard approach of using graphs to reflect on personal data. And secondly, whether they were able to reflect effectively with their chosen method. Participants were told that they would be analysing some self-tracked data in the study and were asked whether they wanted to do this with paper-based graphs (printouts of the reflection tool shown in Figure 2.2) or using the decision tree-based reflection tool. They were then asked to complete 3 activities. The first activity involved predicting a mock user’s mood, given the mock user’s tracked data for the day and their previously tracked data. The second activity was to explain why they had made this prediction. The final activity was to have the participants suggest what the mock user could do to improve their mood. These activities were designed to test whether a participant demonstrated an R2 level of reflection, described in section 2.5.

To help with this study, a mock dataset was created using real user data. However, it was modified to emphasize exercise and to-do list as major factors influencing the mock user’s mood. It is important to note that these modifications resulted in a completed to-do status having a greater weighting on mood, in comparison to level of exercise. This approach allowed for an easier evaluation of the effectiveness of the decision tree-based reflection tool in identifying and explaining the key factors contributing to the mock user’s current mood. A list of values for the mock user’s current day was also created. For consistency, this mock day entry was created using real user data, but the mock user’s exercise was set to none, and to-do status set to incomplete.

Overall, 16 people agreed to participate in this study. These 16 people mainly consisted of computer science students and academic staff. To gather the results for this study, participants were asked to complete a questionnaire by scanning a QR code.

The questions asked were as follows:

1. Did you use the graphs or the app?
2. Do you feel like you successfully identified how the user might currently be feeling?
3. Do you feel like you can explain why the user might be feeling this way?
4. Do you feel like you were able to identify what the user could do to increase their mood?

A screenshot of the aforementioned questionnaire is shown in Appendix A.

5.2.1 Results

The majority of participants chose to use the decision tree-based application over the graphs. Figure 5.1 shows that 75% of participants chose the decision tree-based app. Interestingly, some of the 25% that chose to use the graphs, hinted at the fact that they were doing so as a challenge for themselves. Additionally, one participant was under the impression that the decision tree was a generalised model, and chose to use the graphs as they felt like they could create a more tailored prediction.

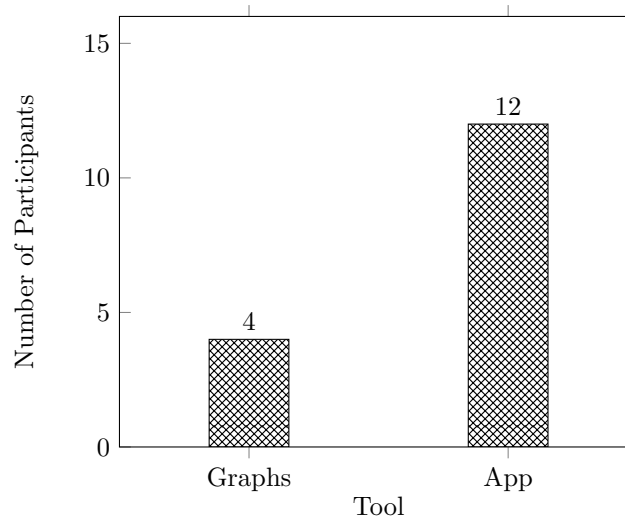


Figure 5.1: Choice of reflection tool when presented with the option of paper-based graphs or the decision tree-based app

The importance of this study is highlighted by the three figures below. Figures 5.2 and 5.3 show that the criteria for an R1 level of reflection was met by 11/12 participants (91.7%) using the app compared to 1/4 participants (25%) using the graphs. In other words, they were able to predict (Figure 5.2) that the mock user would be feeling low as a result of the exercise level and state of the to-do list. Figure 5.4 shows that a further 10/12 app users (83.3%) met the criteria for an R2 level of reflection, as they suggested that the mock user could slightly improve their mood through exercise, and greatly improve it by completing their to-do list.

On the other hand, the results were less favourable for the participants who used graphs as a tool for reflection. Only one of the participants was able to meet the criteria necessary for an R2 level of reflection (Figure 5.4). Interestingly, one of the participants using graphs, who had made it clear that they were using this medium as a challenge for themselves, got frustrated and gave up on the activities.

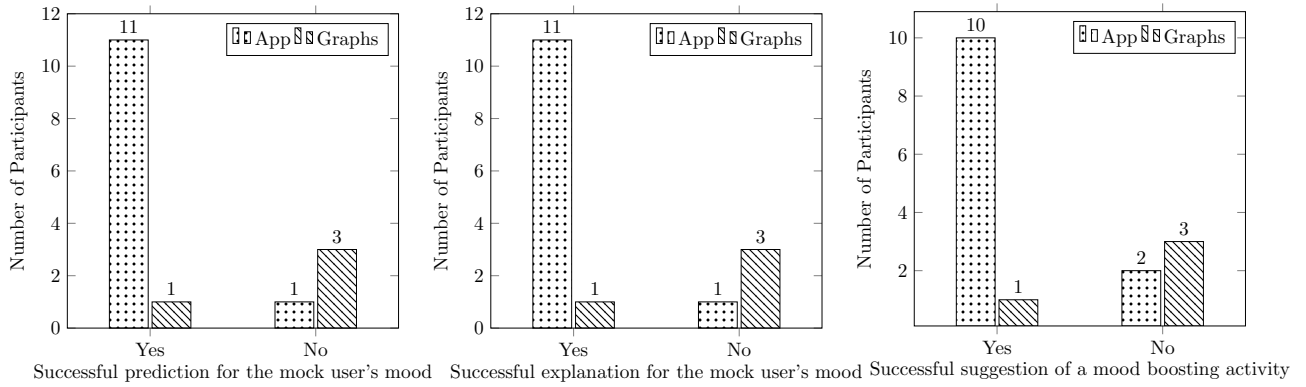


Figure 5.2: Ability to predict a user's current mood

Figure 5.3: Ability to explain a user's mood

Figure 5.4: Ability to suggest a mood boosting action

5.2.2 Poster fair study summary

In summary, the poster fair study demonstrated that the decision tree-based application was the preferred and most effective tool for reflection in comparison with the paper-based graphs. It also showed that an overwhelming majority of the participants who used the decision tree-based application were able to meet the criteria necessary for an R2 level of reflection. In contrast, it showed that the participants who used the graphs were unable to meet the same criteria, and gives additional evidence for this approach not being as effective as a tool for reflection.

5.3 User Evaluation in the Wild

The second user study aimed to have participants use the decision tree-based app over a three week period. This allowed the participants the opportunity to track their own data, and potentially reflect on it. While the poster day study suggested that the reflection tool could facilitate reflection, it was important to see whether participants would engage with the app over an extended period of time in the wild.

The study aimed to collect both quantitative and qualitative data about how participants used the app. The app used Firebase Analytics features to record how the participants engaged with the app and whether and how they used the different functionality. When implementing the speech settings in section 4.9.4, the API that we used to determine the user's language code, was not supported by Expo version 46. To resolve this, we upgraded our project to Expo version 48. At the time of development, the Firebase Analytics SDK had not been fully integrated with Expo version 48, and as a result, events were not logged correctly. Unfortunately, we realised this too late. Due to the failure to accurately generate the user analytics over the user's three week study period, we decided to omit all user analytics from this evaluation.

Despite this shortcoming, the qualitative user feedback that we received when interviewing the participants at the end of the study was invaluable in highlighting the positives and negatives of the decision tree-based application.

5.3.1 Participants

Participants were recruited opportunistically from our network of friends and family. The participants had the following demographics and previous experience with self-tracking:

P1: 21, Male, Student. Only ever tracks steps.

P2: 51, Female, Working. Tracks data automatically by wearing a FitBit, also tracks food and water in a weight app. Also uses paper and pencil to track all behaviour related to migraines, for the purpose of finding patterns.

P3: 19, Female, Student. Tracks her period, running and step count.

P4: 21, Female, Student. Tracks her mood and other related data alongside her period.

After agreeing to take part in the study, the participants were asked to download the Expo Go application. We then sent them a QR code that Expo generated to use for distribution, shown in appendix B.1. During this initial process, we maintained direct communication with each participant, guiding them through the set-up of the app. For P4, who used an Android device, the set-up process was seamless and they were up and running instantly. For P1, P2 and P3 this was not the case. Apple had blocked the anonymous distribution of Expo applications, and as a result, these three users were asked by us to make Expo accounts, so that we could officially add them to the project.

Once the user set-up was complete, participants were then asked to track their data over the three week period. We explained to them the scope of the project and described the features that we had implemented. Once all questions were out of the way, we stepped aside and allowed them to make the most of the app. We did not intervene during this three week period.

5.3.2 End of Study Interviews

At the end of the three week study period, participants were asked to take part in an interview about their experience using the app. P1's interview was carried out over FaceTime, while the other three were carried out in person. The following questions were asked:

1. Did you use the app?
2. Tell me about your experience using the app?
3. How often did you use the app?
4. How did you use the app?
5. Did you look at the prediction?
6. How accurate did you find the prediction?
7. Did you explore how changes in your data might effect your mood?
8. When it was inaccurate, how did you feel?
9. Did using the app lead to you learning about yourself?
10. What did you learn?
11. Did using the tool lead you to change your habits?
12. What other data would you want to track?
13. Do you have any final suggestions?

We chose to ask these questions as they allowed the interview to remain as open-ended as possible, while still providing us with the relevant feedback necessary to determine the acceptability and feasibility of the decision tree-based app. All interviews were recorded and transcribed.

5.3.3 Results

This dissertation was primarily focused on facilitating self-reflection with the help of an app. However, the evaluation questions were important to create a complete overview of the user's experience.

Questions 1, 2, 3 and 4 were asked in order to develop an understanding of how the participants used the app and their general experience with it. Questions 5, 6, 7 and 8 were asked to determine the accuracy of the decision tree model, and the usability of the tools that we developed for self-reflection. Questions 9, 10 and 11 were asked to gain insight into whether the app facilitated self-reflection and behaviour change. Finally, questions 12 and 13 were asked in order to outline any criticisms and suggestions that the participants had while using the app, to better inform us for future development.

In the subsections below, we have collated the user feedback and highlighted all of our key findings.

5.3.3.1 App Experience

When asked whether they had used the app, every participant responded saying that they had. The general consensus was that the app was easy to use, with P3 describing the experience as “really easy to use on a daily basis”.

Due to the freedom that users had while using the app, i.e. the ability to track data whenever they wanted, it was interesting to hear how each participant used the app. For example, P1 stated that “I fell into a sort of a dual routine where in the morning I wake up and track my sleep and yesterday’s alcohol, then wait for the end of the day to track the other factors”. While on the other hand, the other three participants tracked all of their data at the end of the day.

One shortcoming, mentioned by every participant, was the lack of a notification system. P4 brought it to our attention that on some days they would forget to log their data, because of the lack of notifications. Similarly, P3 mentioned that “it was easy at the beginning to forget about logging data, but once it became part of a routine, that did change”. P1 made a personal solution to this problem, by setting a daily reminder to track their data.

With regards to the tracking of data, each participant found the UI easy to use. However, there were mixed opinions on the actual data being tracked. The most frequent cause of data tracking related problems was water intake. P2 and P4 found it difficult to track their water intake, as they did not have a good understanding of the quantities that they were consuming, without measuring. Despite finding it difficult to track, they both reported that the tracking of their water intake resulted in an increased awareness, allowing them to make healthier choices surrounding water consumption. P1 and P2 reported that they did not find tracking the water intake difficult, but they both explained that this was because they always drink from containers of known volume.

Another notable issue with the data tracking, raised by P2, P3 and P4, was the exclusion of period tracking. They all reported that their menstrual cycle was an important variable to track, and would have liked to have the ability to do so within the app. More generally, each participant suggested that they would have found increased functionality from the app if they had been given the option to add their own trackers.

Overall, the app was reported to be easy to use. After an initial period establishing the app as part of their daily routine, participants were successful in being able to self-track their data. The app would have greatly benefited from the inclusion of a notification system, however this did not greatly impact user data collection for the duration of this study.

5.3.3.2 Mood Predictions & Exploring Relationships

When asked about the prediction card, every participant answered that they had seen it, and made use of it. P3 stated that “the prediction card was fun to see and it motivated you to fill in the tracked data”. Additionally, P2 asked whether they would still have access to the app now that the study period was officially over, as they really enjoyed seeing their mood prediction each day.

The accuracy of the predictions varied somewhat for each participant. However, it was reported by P1, P2 and P3 to be mostly correct. P2 gave us access to their data to explore the model accuracy further, which we cover in section 5.4. When asked about discrepancies in accuracy, a general theme was made apparent. P1, P2 and P3 all observed that it was on days where their behaviour differed greatly from their daily routine, that the prediction became inaccurate. For example, P2 is a migraine sufferer. They stated that “it still might be that everything else was consistent like the sleep, the drinking of the water, even exercise or social activity might still all be consistent, but just because of maybe time of the month or, you know, what other things trigger my migraines, it may well be that that made me feel rubbish”. In a similar vein, P3 said that, “when I’d had a more typical day in comparison to the data that had been logged, it was more accurate with predictions”. On the contrary, P4 did not find the model to be accurate and said “I think that’s strictly because of the actual data it was collecting”, suggesting that the app was not tracking enough data about them to make an accurate prediction.

The use of the exploration tool was a lot more varied than the use of the prediction card. Both P2 and P4 reported that they did not make any use of the exploration features. P2 explained that this was because they was unaware that this was something that they could do. In contrast, P1 and P3 both made positive use of the exploration tool. P1 found that “On low days, typically, I hadn’t drank as much water”. They said that this prompted them to explore the relationship between their water intake, using the exploration tools, and found that “if I increased that count it would increase my mood”. Likewise, P3 found similar use. They stated, “There was one day where I’d put my mood in as like, happy, and the prediction was mid or low, I can’t remember which. And I was like, oh that’s strange, I wonder what

caused that. Once I changed, for example, the social interaction, or my sleep, I was able to see potential changes in my mood.”

Despite the overall success of the predictions for the majority of participants, it was also interesting to observe how incorrect predictions affected the participants. A self-observation made by P1 was that “If I was in a bad mood and it told me I was in a good mood, I would strive to pick myself up a bit. Whereas, if I was in a good mood and it told me I was in a bad mood, I was just like pfft, okay”. This highlights the short term effect of that the app had on P1’s behaviour, mainly that false positive predictions resulted in positive behaviour change for them. Similarly to P1, P3 found that their mood was subconsciously affected by the prediction, noting that, “I think that when it was wrong, positively wrong, I think that would kind of, that had a positive effect because you think, oh actually maybe I shouldn’t be feeling so bad”. In contrast to P1 however, P3 also noticed the opposite effect when the mood prediction was negatively wrong.

For P2 and P4, the incorrect mood predictions had a different effect on them altogether. P2 stated for the mood prediction that, “If it got it right, I was like yeah, cool, this is working. And if it got it wrong, it would make me think, oh, why’s that not quite got the same as me there. What didn’t it know that would’ve affected...If it got it right, it was a bit like having your homework marked correctly, it was like yay”. Similarly, P4 said “If it got it wrong, it would make me think, ah, why is that not quite got the same as me there?”. For both P2 and P4, wrong predictions prompted them to reflect on the parts of their day that the decision tree model was unaware of, which would have contributed to discrepancies in their mood.

5.3.3.3 Self-Reflection & Behaviour Change

When asked whether using the app had resulted in the participant learning about themselves, there was unanimous agreement that it had, irrespective of their prior self-tracking experience. P1 stated that “Generally, it’s made me aware of my trends, and aware of my bad habits in particular”.

P2, by using the app, was able to learn that on days where they logged a good mood, typically they had done more exercise. This insight allowed them to reflect deeper, and they explained to us that they believe that it wasn’t necessarily the exercise that was boosting their mood, but on days that they were able to do more exercise, were days that they were not in work.

P1, P3 and P4 were all able to learn that they often hydrate too little, and that water intake has a noticeable effect on their mood outcome.

An interesting discovery that we made was how the difference in prior self-tracking history affected any change in behaviour resulting from this app. P2, for example, explained that they meticulously self-track their data, and have been doing so for quite some time. P2 did not report any behaviour change from using the app. P1, on the other hand, had very minimal self-tracking experience prior to this study. P1 stated that, “I learned that I probably consume a bit too much alcohol per week, but I certainly began cutting that down... For a while I’ve needed that kick in the arse, to realise that”. They then continued by saying, “I certainly drink more water”. For P1, using the app resulted in positive behaviour change in multiple areas.

P1 was not the only person to report positive behaviour change either, P3 and P4, also mentioned behaviour changes that they had made as a result of using the app. Both P3 and P4’s self-tracking experience lies somewhere in the middle of P1 and P2, due to their prior use of period tracking apps. P3 and P4 both reported that the increased awareness around their water intake, resulted in them increasing their water intake to a healthier quantity.

5.3.3.4 Suggestions

Although suggestions and criticisms were given throughout each interview, questions 12 and 13 were asked to have the participants summarise these opinions.

The most notable criticism, mentioned by every participant, was the lack of a notification system. Despite P1’s temporary solution of setting reminders each day to use the app, the app would have greatly benefited from the implementation of a notification system, to aid in the data tracking process.

Of equal importance, was the exclusion of period tracking features. Due to the impact that their menstrual cycle has on their mood, every female participant suggested adding period tracking as an option, to help with the accuracy of the decision tree model.

Finally, there was a general theme of wanting more control over the data that they tracked. If given the option, participants stated that they would have found increased use from the app if they were able to add and customise trackers as they see fit.

5.3.4 Evaluation in the Wild Summary

The three week in the wild user evaluation provided invaluable information on the decision tree-based application as a tool for self-reflection. Firstly, to backup the results found in the poster day study, the participants of the study found that they were able to self-reflect at an R2 level of reflection, that is, they were able to explain what factors influenced their mood and explore what changes they could make to improve it. We were also able to learn about features that would enhance the functionality of this application. Mainly the inclusion of a notification system, and also providing the users with the ability to control what they track.

Finally, and arguably most importantly, this study demonstrated that using the app had a positive impact on the participants. The aim of a personal informatics systems, after all, is to help the user on their journey towards self-improvement. In their interview, P1 identified that they might be drinking too much alcohol, and as a result has reduced their alcohol intake. Also, P1, P3 and P4 have all been able to identify that they do not drink enough water, which can contribute to their low mood. As a result of using the app they have now started drinking more water.

5.4 Evaluating the Performance of the Decision Tree

To help evaluate the accuracy of the decision tree predictions, P2 was happy to share the data that they had tracked over the three week study period, for the purpose of analysis. The data collection, the building of the decision tree and the mood prediction for each day, were replicated with a script. This was done to analyse the accuracy of predictions that User Two received during their time using the app.

With the initial seven day training period, and the rebuilding on each subsequent day, User Two's decision tree had an accuracy of 71.42% for the two weeks that they received predictions. Multiple sources deem a machine learning model accuracy of 70% and above to be good [3] [4]. Figure 5.5 shows the difference in true and false predictions made by the decision tree-based application.

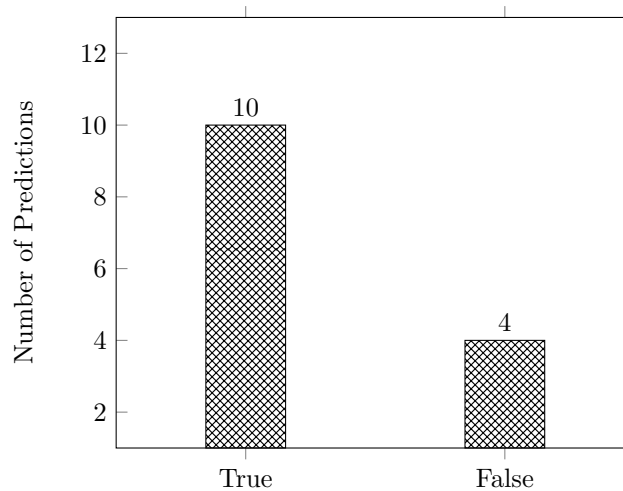


Figure 5.5: True and False Predictions

Considering that the decision tree rebuilds with each new day, we wanted to evaluate the evolving nature of the model. Figure 5.6 shows the cumulative accuracy of the model with each new day. As we can see there is a general trend of increasing accuracy, however there is a sudden decrease in the final few days of User Two's trial period. However, User Two did explain that when the model got their mood completely wrong, they were suffering from a hormonally related migraine, factors that were not considered by the decision tree model.

Due to the ternary nature of the classification, it is important to not only measure whether the model was right or wrong, but how wrong it was and when it was wrong. Figure 5.7 shows the distance of the prediction from the ground truth for each day. For example, if the prediction was "Good" and the ground truth was "Good", then the distance would be zero. On the other hand, if the prediction was "Good" and the ground truth was "Low", then the distance would be two.

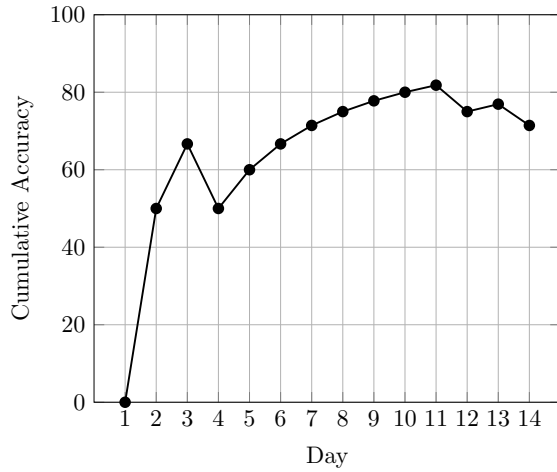


Figure 5.6: Cumulative accuracy of the decision tree over two weeks

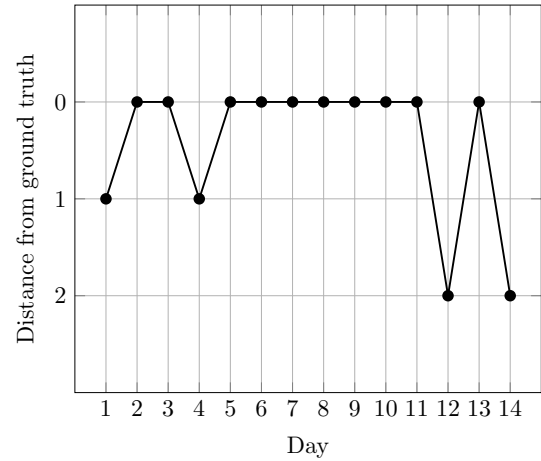


Figure 5.7: Distance of the prediction from the ground truth

5.5 Summary

This section will summarise the findings of our studies that evaluated the effectiveness of the app to facilitate reflection and behaviour change. Section 5.4 provided a quantitative overview of decision tree performance versus the ground truth, highlighting the model's ability to predict users' moods over time. The studies outlined in sections 5.2 and 5.3 both demonstrated the success of the app in both the long and short term. In particular, the interviews conducted as part of the three week in the wild study highlighted specific cases where users changed their habits as a result of using the app: P1 reduced their alcohol intake and P1, P3 and P4 increased their water intake. While the question of whether these changes would be sustained over a long period of time remains unanswered by our study, it is encouraging that users reported the application having a positive effect on their daily lives. This points strongly towards the efficacy of the application in facilitating positive behaviour change.

Chapter 6

Conclusion

6.1 Overview

This dissertation investigated whether decision trees can facilitate user reflection on their self-tracked data. In this chapter, we will discuss our successes in this task, and highlight the achievements made during this project. We will re-summarise the steps that were taken to reach this stage, and emphasise the conclusions we have drawn from our studies. The chapter will finish by exploring opportunities for future work that can be done to enhance reflection through the use of decision trees within a personal informatics system, for which our work has served as a proof-of-concept.

6.2 Achievements Summary

This dissertation has resulted in the development of a personal informatics tool that facilitates reflection through the use of decision trees. This claim is supported by the data collected during the poster day study and the user feedback from the user evaluation study.

The success of this project is, firstly, a result of the development of a lightweight decision tree model that is able to build and classify on a mobile device. Not only was this decision tree algorithm able to perform well on the most accessible computational device, but it was also developed using a popular programming language, which allowed us to make it widely available for other researchers to use, by publishing it on the NPM registry.

Through the results we described in chapter 5, we can confidently conclude that decision tree-based applications succeeded in facilitating self-reflection at an R2 level. Participants were able to meet an R2 level of reflection as they were able to explain what factors influenced their mood and explore what changes they could make to improve it. Furthermore, for three of the participants in the three week in the wild study, using the app led to positive behaviour changes, specifically, P1 reduced their alcohol intake and P1, P3 and P4 increased their water intake.

By addressing the weaknesses and avenues for improvement of existing work and methodologies, we have produced an application which serves as a proof-of-concept to facilitate future work, which we describe in section 6.3. The application we produced was used in user studies which resulted in qualitative data that demonstrated the efficacy of the app to provide predictions about users' moods, and also identified how the app could be improved in future design iterations.

The execution of this project did not run without any issues. Notably, we encountered issues with using Firebase to collect analytics in the three week in the wild study which meant we did not have quantitative data about how and how often participants used different features of the app. However, the rich qualitative data from the interviews with participants demonstrated that the app facilitated self-reflection at the R2 level and led to three of the four participants changing their habits in order to improve their mood.

Although the second study lasted three weeks, it would have been interesting to see how participants would use the app over an even longer period of time. Furthermore, having more participants would improve the validity of the results.

The interviews provided evidence on how the app can be improved and we discuss this in the next section.

6.3 Future work

While our application has served as a successful proof-of-concept, the longer study carried out in section 5.3 brought to our attention avenues for improvement commonly suggested by participants. The most notable of these suggestions was for the inclusion of a notification system; our current solution leaves it up to users themselves to remember to track their data for the day. We saw that this conclusively leads to gaps in data and hence misrepresentation of trends, which ultimately could result in misclassifications by our model. Future work would therefore include a daily notification system to remind users to track their data.

Currently, the application only performs predictions of a user's mood and leaves users to find mood boosting solutions themselves. It was therefore suggested by an academic at the poster fair mentioned in section 5.2 that the tree structure of decision trees could be exploited in the creation of a path finding algorithm to suggest the shortest path from one label to another. In this way, whenever the algorithm predicts a negatively ranked label the app could automatically suggest to the user the easiest action they could take to improve their mood. It could also be explored to what extent increasing the interactivity of the app in this way could encourage continued use in users.

In order to substantiate our study findings, there is scope for supplementary research to quantitatively explore user-analytics which our longer study failed to. For example, exploration into how users typically engaged with the app over a prolonged period of time could elicit changes in how users are encouraged to engage. It would also be of interest to research how the predictions are influenced by users adding custom parameters when logging events. If it is the case that predictions are vastly improved when users have created custom parameters, further research could be done into supporting greater customisation.

Bibliography

- [1] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation), 2016.
- [2] Sachin Agarwal and Michael Rambeau. The state of javascript 2020. <https://2020.stateofjs.com/en-US/>, 2020. Accessed: 2023-05-02.
- [3] Obviously AI. How to evaluate the performance of a machine learning model. <https://www.obviously.ai/post/machine-learning-model-performance>, 2022.
- [4] Stephen Allwright. What is a good accuracy score? <https://stephenallwright.com/good-accuracy-score/>, 2022.
- [5] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.
- [6] Jong-Myon Bae. The clinical decision analysis using decision tree. *Epidemiology and health*, 36, 2014.
- [7] Jon Bird, Peter Byass, Kathleen Kahn, Paul Mee, and Edward Fottrell. A matter of life and death: practical and ethical constraints in the development of a mobile verbal autopsy tool. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1489–1498, 2013.
- [8] Pedro Bordalo, Nicola Gennaioli, and Andrei Shleifer. Salience. *Annual Review of Economics*, 14:521–544, 2022.
- [9] Kate Crawford, Jessa Lingel, and Tero Karppi. Our metrics, ourselves: A hundred years of self-tracking from the weight scale to the wrist wearable device. *European Journal of Cultural Studies*, 18(4-5):479–496, 2015.
- [10] Daniel W G Davies. Manderley. Bachelor’s thesis, University of Bristol, 2022.
- [11] Daniel W. G. Davies. Manderley. Bachelor’s thesis, University of Bristol, 2022.
- [12] Google Developers. Let’s write a decision tree classifier from scratch - machine learning recipes #8. <https://www.youtube.com/watch?v=LDRb09a6XPU>, 2017.
- [13] Dheeru Dua and Casey Graff. UCI machine learning repository, 2019. URL: <http://archive.ics.uci.edu/ml/datasets/Iris>.
- [14] Expo. Create your first app, 2021. URL: <https://docs.expo.dev/tutorial/create-your-first-app/>.
- [15] Expo.io. Expo. <https://expo.io/>, 2023.
- [16] Facebook. React - a javascript library for building user interfaces. <https://reactjs.org/>, 2023.
- [17] Facebook. React native. <https://reactnative.dev/>, 2023.
- [18] Firebase. Firebase. <https://firebase.google.com/>.
- [19] Firebase. What’s new at google i/o. <https://firebase.blog/posts/2022/05/whats-new-at-google-io/>, 2022.

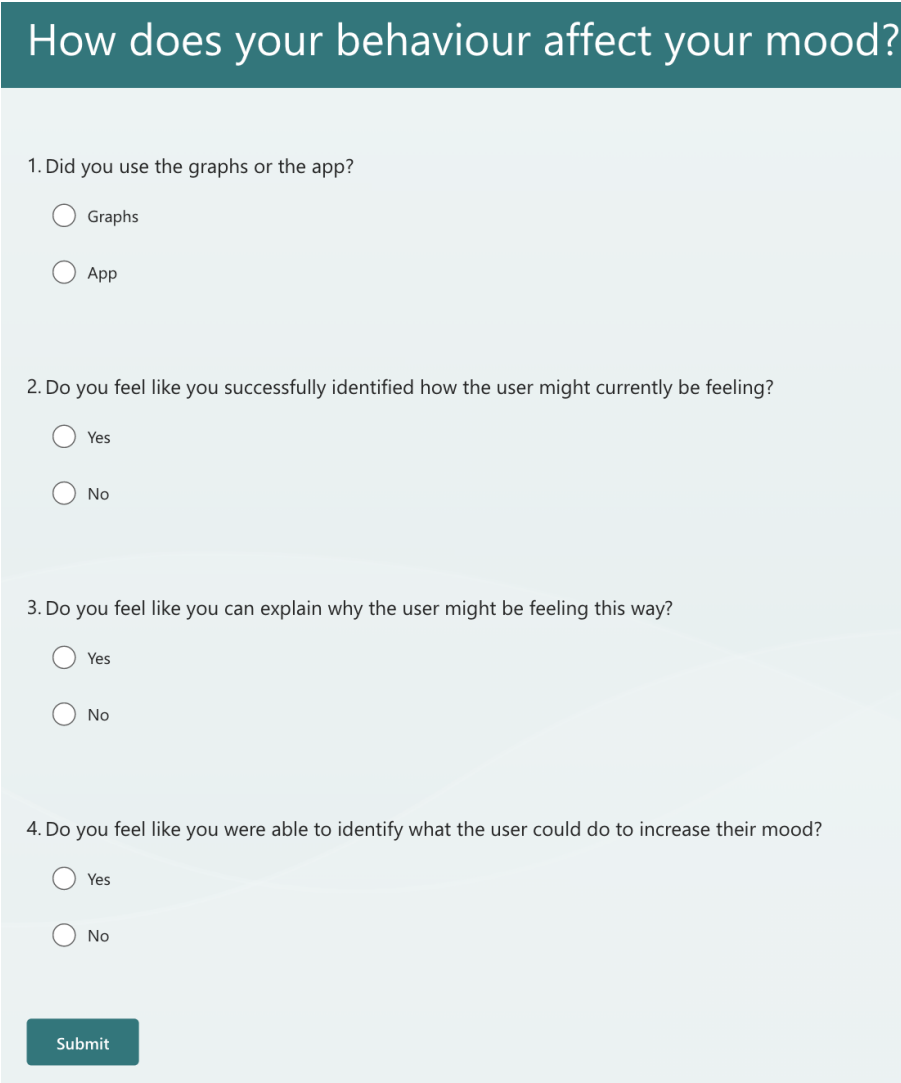
- [20] Rowanne Fleck and Geraldine Fitzpatrick. Reflecting on reflection: framing a design landscape. In *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction*, pages 216–223, 2010.
- [21] Mirta Galesic and Rocio Garcia-Retamero. Graph literacy: A cross-cultural comparison. *Medical decision making*, 31(3):444–457, 2011.
- [22] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. ” O’Reilly Media, Inc.”, 2022.
- [23] Henner Gimpel, Marcia Nißen, and Roland Görlitz. Quantifying the quantified self: A study on the motivations of patients to track their own health. 2013.
- [24] David Grande, Nandita Mitra, Raghuram Iyengar, Raina M Merchant, David A Asch, Meghana Sharma, and Carolyn C Cannuscio. Consumer willingness to share personal digital information for health-related uses. *JAMA Network Open*, 5(1):e2144787–e2144787, 2022.
- [25] Graphviz Developers. Graphviz - Graph Visualization Software, 2021. URL: <https://graphviz.org/>.
- [26] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
- [27] Ecma International. EcmaScript language specification. <https://www.ecma-international.org/publications/standards/Ecma-262.htm>, 2022.
- [28] Elisabeth T Kersten-van Dijk, Joyce HDM Westerink, Femke Beute, and Wijnand A IJsselstein. Personal informatics, self-insight, and behavior change: A critical review of current literature. *Human-Computer Interaction*, 32(5-6):268–296, 2017.
- [29] Tae Kyung Kim and Moon Choi. Older adults’ willingness to share their personal and health information when adopting healthcare technology and services. *International journal of medical informatics*, 126:86–94, 2019.
- [30] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.
- [31] Ian Li, Anind Dey, and Jodi Forlizzi. A stage-based model of personal informatics systems. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 557–566, 2010.
- [32] Deborah Lupton. *The quantified self*. John Wiley & Sons, 2016.
- [33] Saida Mamedova and Emily Pawlowski. Adult literacy in the united states. 2019.
- [34] Nicole McCarthy. Period tracking apps used in court case could be tipping point for data privacy. *Forbes*, 2019. URL: <https://www.forbes.com/sites/nicolemartin1/2019/05/24/period-tracking-apps-used-in-court-case-could-be-tipping-point-for-data-privacy/?sh=77631f6b38f6>.
- [35] National Health Service (NHS). Feelings and symptoms. <https://www.nhs.uk/mental-health/feelings-symptoms-behaviours/feelings-and-symptoms/>, accessed 2023. Accessed on May 4, 2023.
- [36] Avinash Navlani. Decision tree classification in python tutorial. <https://www.datacamp.com/tutorial/decision-tree-classification-python>, 2023.
- [37] Robert G Nelson, William C Knowler, Matthias Kretzler, Kevin V Lemley, Helen C Looker, Michael Mauer, William E Mitch, Behzad Najafian, and Peter H Bennett. Pima indian contributions to our understanding of diabetic kidney disease. *Diabetes*, 70(8):1603–1616, 2021.
- [38] Inc. NPM. npm. <https://www.npmjs.com/>, 2023.
- [39] OnlyOneAman. Creating an npm package from react component, 2020. URL: <https://onlyoneaman.medium.com/creating-an-npm-package-from-react-component-ee5b0ba0cd49>.

- [40] Stack Overflow. Stack overflow developer survey 2022. <https://survey.stackoverflow.co/2022/>, 2022.
- [41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [42] Vili Podgorelec, Peter Kokol, Bruno Stiglic, and Ivan Rozman. Decision trees: an overview and their use in medicine. *Journal of medical systems*, 26:445–463, 2002.
- [43] Piotr Płoński. How to extract rules from a decision tree?, November 2021. URL: <https://mljar.com/blog/extract-rules-decision-tree/>.
- [44] Margreet Riphagen, Marco van Hout, D Kritjnen, and Gijs Gootjes. Learning tomorrow: visualising student and staff’s daily activities and reflect on it. *ICERIE2013*, 2013.
- [45] Yvonne Rogers and Paul Marshall. Research in the wild. *Synthesis Lectures on Human-Centered Informatics*, 10(3):i–97, 2017.
- [46] Qëndresa Rramani, Ian Krajbich, Laura Enax, Lisa Brustkern, and Bernd Weber. Salient nutrition labels shift peoples’ attention to healthy foods and exert more influence on their choices. *Nutrition Research*, 80:106–116, 2020.
- [47] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.
- [48] Carol D Ryff and Burton H Singer. Know thyself and become what you are: A eudaimonic approach to psychological well-being. *Journal of happiness studies*, 9:13–39, 2008.
- [49] Gary Sacks, Mike Rayner, and Boyd Swinburn. Impact of front-of-pack ‘traffic-light’ nutrition labelling on consumer food purchases in the uk. *Health promotion international*, 24(4):344–352, 2009.
- [50] Christopher A Sanchez and Jennifer Wiley. To scroll or not to scroll: Scrolling, working memory capacity, and comprehending complex texts. *Human Factors*, 51(5):730–738, 2009.
- [51] scikit-learn developers. scikit-learn: Decision trees. <https://scikit-learn.org/stable/modules/tree.html>, 2021. [Online; accessed 30 April 2023].
- [52] Melanie Swan. The quantified self: Fundamental disruption in big data science and biological discovery. *Big data*, 1(2):85–99, 2013.
- [53] Ian H Witten and Eibe Frank. Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, 31(1):76–77, 2002.

Appendix A

Poster Fair Study

A.1 Questionnaire



How does your behaviour affect your mood?

1. Did you use the graphs or the app?

☐ Graphs

☐ App

2. Do you feel like you successfully identified how the user might currently be feeling?

☐ Yes

☐ No

3. Do you feel like you can explain why the user might be feeling this way?

☐ Yes

☐ No

4. Do you feel like you were able to identify what the user could do to increase their mood?

☐ Yes

☐ No

Submit

Figure A.1: Screenshot of questionnaire

A.2 Results

ID	Start time	Completion time	Did you use the graphs or the app?	Do you feel like you successfully identified how the user might currently be feeling?	Do you feel like you can explain why the user might be feeling this way?	Do you feel like you were able to identify what the user could do to increase their mood?
1	4/19/23 13:30:07	4/19/23 13:33:54	App	Yes	Yes	Yes
2	4/19/23 14:10:06	4/19/23 14:10:30	App	Yes	Yes	Yes
3	4/19/23 14:16:47	4/19/23 14:17:03	App	Yes	Yes	Yes
4	4/19/23 14:25:46	4/19/23 14:26:00	Graphs	No	No	Yes
5	4/19/23 14:50:29	4/19/23 14:51:30	App	Yes	Yes	Yes
6	4/19/23 15:21:42	4/19/23 15:22:07	App	Yes	Yes	Yes
7	4/19/23 15:31:48	4/19/23 15:32:06	App	Yes	Yes	Yes
8	4/19/23 15:31:54	4/19/23 15:32:09	App	Yes	Yes	Yes
9	4/19/23 15:31:54	4/19/23 15:32:10	App	Yes	Yes	Yes
10	4/19/23 15:33:09	4/19/23 15:33:21	App	Yes	Yes	Yes
11	4/19/23 15:34:39	4/19/23 15:38:40	App	Yes	Yes	Yes
12	4/19/23 15:38:41	4/19/23 15:38:58	Graphs	No	No	No
13	4/19/23 15:38:48	4/19/23 15:39:02	Graphs	No	No	No
14	4/19/23 15:39:01	4/19/23 15:39:27	App	Yes	Yes	Yes
15	4/19/23 15:46:50	4/19/23 15:47:26	App	Yes	Yes	Yes
16	4/19/23 16:12:42	4/19/23 16:14:00	App	Yes	Yes	Yes

Table A.1: Poster fair raw data

Appendix B

Evaluation in the Wild

B.1 Distribution

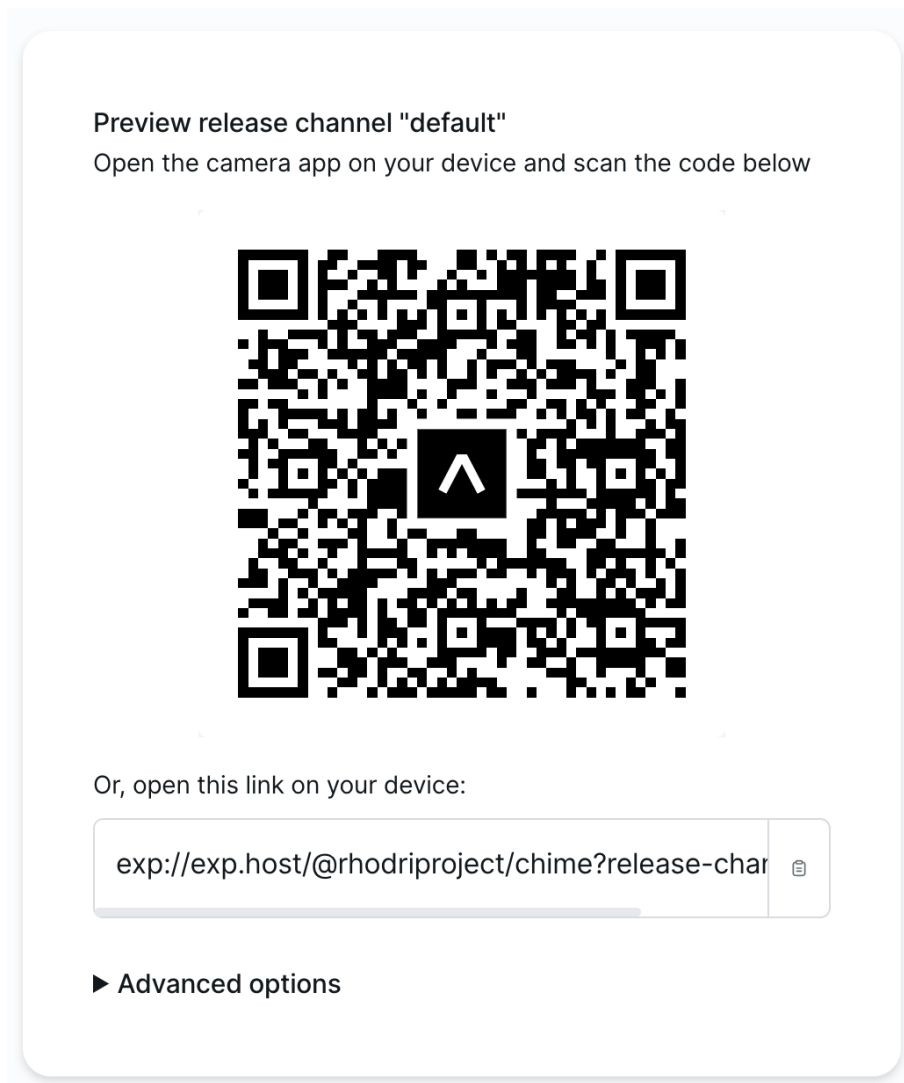


Figure B.1: Screenshot of the Expo Go QR code used for distribution