Research Paper Business Analytics

Predicting participant behavior and dropout in a physical activity experiment using machine learning

Author :
Georgios Christos Chouliaras
sn: 2592496

Supervisor :
Bart A. Kamphorst

Vrije Universiteit Amsterdam
Faculty of Sciences
Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

August 30, 2017

This page intentionally left blank.

Predicting participant behavior and dropout in a physical activity experiment using machine learning

by
Georgios Christos Chouliaras (2592496)

Supervisor: Bart A. Kamphorst

Submitted to the Faculty of Sciences
on August 30, 2017, in fulfillment of the Research Paper BA

Abstract

Physical inactivity is identified by the World Health Organization as the fourth leading risk factor for global mortality as it increases the risk of various adverse health conditions. App-based health interventions promise to help increase physical activity levels by enhancing the motivation of the users to exercise on a regular basis. One such app-based intervention which sends tailored coaching messages to help people become more physically active, has been created by the Active2Gether project. A 12-week physical activity experiment was conducted with 92 healthy young adults to study the effect of the Active2Gether app. In this paper, supervised machine learning is applied on the experimental data, in order to detect which factors together best predict behavioral increase and which predict dropout from the experiment. Several tree-based classifiers are fitted and tuned in order to optimize F_1-score. The classifiers are evaluated using *K-Fold* cross validation and furthermore a *modified repeated* cross validation is applied in order to evaluate the robustness of the models. For predicting behavioral increase, we fitted Decision Tree, Random Forest and Extremely Randomized Trees (Extra-Trees). Results show that the Decision Tree outperformed the Random Forest and the Extra-Trees, achieving an F_1-score 0.8 in the 5-fold cross validation and 0.72 in the repeated cross validation. For participant dropout, we compared Decision Tree, Extra-Trees and Extreme Gradient Boosting (XGBoost). Results show that the Extra-Trees performed best with an F_1-score 0.765 in the 10-fold cross validation and 0.68 in the repeated cross validation.

Contents

Preface

The Research Paper BA is a compulsory part of the master programme in Business Analytics at the VU Amsterdam. The aim of this 'thesis' which is credited with 6 ECTS, is to demonstrate the student's ability to describe a problem in a clear manner. In the Research Paper should be emphasized the business-related aspects of the programme and also the more fundamental aspects of mathematics and computer science.

The main motivation for the topic selection were articles such as [1] and [31] that detail the risks of physical inactivity. This problem is on the rise nowadays as the lifestyle becomes more and more sedentary. My supervisor Bart A. Kamphorst along with Michel Klein and Julienka Mollee gave me the opportunity to work on a project that addresses the problem of physical inactivity, by conducting data analysis and machine learning on datasets from an interdisciplinary project called Active2Gether. This project is concerned with developing and evaluating an intervention app that aims to increase or maintain the activity levels of young adults. Data from various sources were obtained through a randomized controlled trial (RCT) for the evaluation of the intervention app. The main purpose of this paper is to apply machine learning techniques on the experimental data, in order to derive the most important factors for behavioral increase and dropout and create models that are able to predict them accurately.

The business aspect of this paper lies with the fact that these kinds of models could be proven useful to businesses and organizations that build physical activity applications and/or conduct evaluation experiments with these applications. By determining the factors that drive behavioral increase, improvements can be made to the applications in order to beneficially affect as many users as possible. Also, by being able to predict dropout in an evaluation experiment, users that have the lowest probability to dropout can be selected at intake, so to maximize the app usability and minimize the costs (in terms of money invested to users that dropped out).

The mathematics part of this research is represented in the various data analysis techniques and the statistical tests applied in order to derive results with statistical rigor. With regards to informatics, several supervised machine learning algorithms, both simple and complex, are presented, tuned and compared. Emphasis is given to evaluating the models, by taking into account specific characteristics of the datasets such as size and class imbalance.

I would like to thank my research supervisor Bart A. Kamphorst for the exceptional cooperation and the constructive suggestions during the development of this research paper. I would also like to thank Julia S. Mollee and Michel C.A. Klein from the Active2Gether team, for their guidance in getting acquainted with the Active2Gether project and its data.

Chouliaras Georgios Christos
July 2017

1 Introduction

Physical inactivity is identified as the fourth leading risk factor for global mortality [4]. It increases the risk of various adverse health conditions, including non-communicable diseases (NCDs) such as coronary heart disease, type 2 diabetes, breast and colon cancer [31]. Physical inactivity however, is a modifiable behavior and there is the possibility for people to increase their activity levels by adopting a more active lifestyle. Interactive technologies, such as smartphones, might play a role in supporting people to change their behavior towards physical activity. This might especially be relevant to a younger demographic, since young ages are more familiar with the emerging technologies [12].

Studies show that app-based health interventions have been effective in increasing the physical activity levels. For example, King et. al mention that 69% of the participants reported that the apps enhanced their motivation to be more physically active and 71% stated that the apps helped them to exercise on a regular basis [27]. Hence, from the point of view of a business analyst it is interesting to study the effect of such intervention apps as well as the determinants for increase in activity levels.

Another aspect that is worthwhile to examine, is dropout (users quit using the app) and what the important factors are that lead people to stop using such apps. Various experiments are being conducted in order to assess the impact that such intervention apps have on people. These kind of experiments are funded by companies and governmental agencies that have an interest in app-based interventions and want to have the largest possible impact on the users. In a business context, identifying the factors that drive dropout and predicting it, could potentially add value to the organization by maximizing the app usability. Moreover, finding the determinants for behavioral increase could help to increase the beneficial effects of the intervention apps.

An app-based intervention has been created and studied by the Active2Gether project [1], which makes use of existing social networks and mobile devices to automatically send tailored coaching messages and feedback to motivate people to become more physically active. Message tailoring is implemented based on an intelligent prediction of the effect of specific suggestions. The Active2Gether team conducted a 12-week experiment on 92 healthy young adults (18 to 30 years old), who were assigned to one of three groups, using a randomization procedure based on gender, type of smartphone and befriended participants. All groups received a Fitbit in order to track the users' progress. The A2G Full group received the full version of the Active2Gether app which provides personalized messages based on a reasoning system. The A2G Light group received a light version of the app that does not send any coaching messages, but has the same functionality and layout as the full version. Finally, the Fitbit group only received an app that links them to the Fitbit website [36].

Until now, the effect of the Active2Gether app has been studied using a top down approach, by setting certain hypotheses before the start of the experiment and checking if the hypotheses are rejected by the end of it. This paper introduces a bottom-up approach in which machine learning techniques are applied in order to derive insight from the experimental data. For the data analysis and the modeling, the Python language is used, due to the fact that it is fast, well documented and also supports a great variety of functions for machine learning.

1.1 Purpose of this Paper

The purpose of this paper is to conduct a data-driven analysis on the experimental data, in order to detect which factors determine behavioral increase and which determine dropout, and build models that are able to predict these two variables accurately. To achieve that, classification is applied, which is a supervised machine learning technique that allows prediction of the outcome of a certain nominal target variable, based on

[1] https://active2gether.few.vu.nl, last accessed at August 30, 2017.

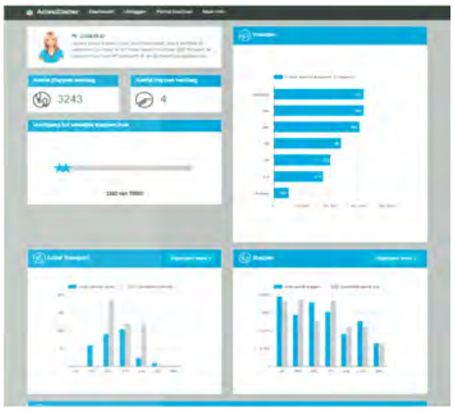
training data. Several classifiers are tuned and fitted for both behavioral increase and dropout. The biggest challenges in this research are the small amount of data and the highly imbalanced nature of the data. Due to the imbalance, evaluation metrics such as F_1 and ROC AUC scores are used as we are mostly interesting in predicting correctly the minority class.

1.2 Paper Overview

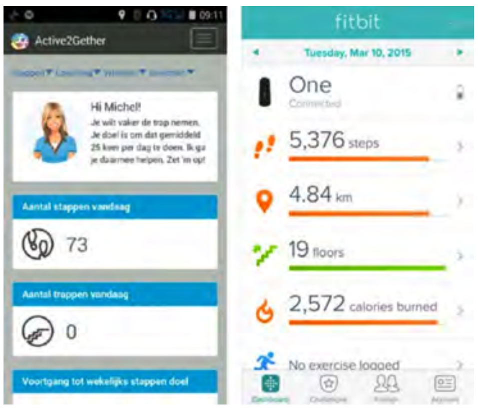
The paper is organized as follows. Chapter 2 introduces the Active2Gether experiment and presents related work. Chapter 3 presents the exploratory data analysis and the pre-processing conducted on the Active2Gether data. It also reports information about the feature extraction of the questionnaire data. Chapter 4 presents the modeling procedure along with results and evaluation of the models, both for behavioral increase and dropout. Chapter 5 provides a discussion about the findings of this paper. Chapter 6 concludes with the most important findings and suggestions for future work.

2 Background & Related Work

The Active2Gether project combines domain knowledge from experts in physical activity interventions with modern mobile technology to design an intervention app that encourages healthy young adults to increase their physical activity levels. An innovative aspect of the system is that it exploits model-based reasoning techniques in order to tailor the coaching to the users' needs, something that until now, has hardly been applied within existing intervention apps [35]. The purpose of the Active2Gether system is to increase or maintain physical activity levels among young adults in the age group of 18 to 30 years. To achieve that it employees the most promising behavior change techniques such as self-monitoring, performance, feedback, goal setting and social comparison [11] [34]. The app performs four main functions: it communicates with the user regarding his/her objectives about physical activity for next week, provides timely and personalized feedback, facilitates self-monitoring based on various collected data sources, and supports social comparison with the help of Facebook friendship relations. The focus of the system is on three types of physical activity: leisure time sports activities, active transport and climbing stairs. Users have the option to be coached on at most one of these three domains at the same time [29]. Screenshots of the Active2Gether dashboard, app and the Fitbit app, can be seen in Figures 1a and 1b. In the next few paragraphs the most important points of the Active2Gether system are explained.



(a)



Hi Michel!
Je wilt vaker de trap nemen. Je doel is om dat gemiddeld 25 keer per dag te doen. Ik ga je daarmee helpen. Zet 'm op!

(b)

Figure 1: (a) Screenshot of an Active2gether dashboard. (b) Screenshot of the Active2gether app (left) and the Fitbit app (right)

New users start by filling out an online intake questionnaire, which includes questions about their daily life (e.g., occupation, significant locations) and about psychological factors underlying their physical activity behavior (e.g., self-efficacy, intentions, perceived barriers). The users then start a one-week assessment, in order to gauge their current physical activity level. For the collection of physical activity data, a *Fitbit One* activity tracker is used, in combination with prompted daily user input about active transport and sport activities. After the assessment week, a categorization is applied to the users based on whether they meed the Dutch physical activity guidelines [5] and whether they think they should be more active. This categorization is repeated every three weeks, in order to tailor the system to the user's latest awareness state and includes the following categories: *education, coaching* or *feedback*. Depending on which awareness category the user is in, he/she receives messages tailored to this category.

Users assigned to the coaching category are suggested to select one of three possible domains (i.e., active transport, climbing stairs or sports activities) to focus on the next week. Then, they are prompted to set a domain-specific goal. If the previous goal for this domain has been met by the user, the system suggests to increase it, otherwise the user is suggested to keep the same goal. Afterwards, the system estimates what types of coaching messages are expected to be most effective for the user based on simulations that use a computational model. With these simulations the system hypothesizes what the effect of improvement in each of the determinants of the behavior would be and the user receives coaching messages based on the selected domain and the three most promising determinants. These messages are filtered to remove any messages that are irrelevant or not applicable to the user. The coaching cycle is repeated on a weekly basis, in order to tailor the coaching to the user's current state and needs.

In order to evaluate how users used the app, a trial has been conducted that is described in [36]. As mentioned in the introduction, participants were assigned to one of three conditions and each condition received a variant of a physical activity app. The experimental conditions, namely A2G Full and A2G Light received a full and a light version of the app respectively. Finally, the Fitbit condition, which is the control group, only received an app that links the users to the Fitbit website. After filling out the intake questionnaire, all the participants received a Fitbit One activity tracker and installed their assigned physical activity app. The participants used the app for a period of twelve weeks or longer, depending on their availability for the final appointment. By the end of the twelve weeks, the users received a link to the final questionnaire, which included questions about their experience with the app.

In [36] it is stated that from 92 participants in total, 24 were assigned to the A2G Full condition, 23 to the A2G Light and 45 to the Fitbit condition. From these 92 users, 35 dropped out from the experiment. Participants were marked as dropouts if they consecutively did not upload any Fitbit data for at least one day before the end of the experiment. It was also found that the mean rank of the number of days that participants used the app differed significantly between the three conditions. Concerning the evaluation of the user experience, the authors performed a factor analysis which revealed four factors, namely satisfaction, user friendliness, perceived effectiveness and professionality. Significant differences in these factors existed between the Active2Gether and Fitbit group, but not between the two Active2Gether conditions. The same differences were also observed for the overall user experience rating.

Until now, machine learning techniques have not yet been applied in the Active2Gether dataset. Hence, given the fact that machine learning provides powerful tools to make predictions and derive insight from data, it is interesting to see how these techniques can help explaining the data from the Active2Gether experiment.

So far, several studies have deployed machine learning methods to deal with problems related to physical activity. As an example, in [14] Ellis et. al, applied machine learning to correctly classify types of physical activity behavior and predict energy expenditure. Physical activity data were obtained by accelerometers worn on the wrist and hip, while energy expenditure and metabolic equivalents (METs) were computed by a portable indirect calorimeter. They created a random forest classifier to predict activity type and a random forest regressor to estimate METs. Using the hip accelerometer they predicted four activity types (household, stairs, walking, running) with an accuracy of 92.3%. For the prediction of the energy expenditure they obtained per minute RMSE of 1.09 METs using the right hip device. The results of this study demonstrate the validity of random forests for physical activity type and MET prediction using accelerometers. In [44] Trost et. al, use decision tree (DT) models for the classification of physical activity intensity from acceleromenter output and Gross Motor Function Classification System (GMFCS) classification level. The models were applied in a total of 57 ambulatory youth with cerebral palsy (CP). The decision trees utilized to identify vertical axis (VA) and vector magnitude (VM) count thresholds, corresponding to sedentery (SED), light (LPA) and moderate-to-vigorous physical activity (MVPA). The classifiers showed relatively high performance, as the average cross-validation accuracy for the vertical axis decision tree was 81.1%, 76.7% and 82.9% for GMFCS levels I, II, and III, respectively. For the vertical magnitude decision trees, the average cross validation accuracy was 80.5%, 75.6% and 84.2% respectively. In [41], Saez et. al, compare various classification algorithms for automatic, cross-person activity recognition under different scenarios that vary in the amount of information available for analysis. They made use of a wide set of classical as well as state-of-the-art classification techniques, such as distance-based methods (k-NN), statistical methods (Gaussian NB, LDA), kernel methods (stochastic gradient descent, support vector machines), decision tree-based methods (CART, Random Forest, Extra-Trees), ensemble learners (boosting, bagging and voting) and deep learning (deep neural network). The results of this research show that the model which performed best is the Extra Trees with an average accuracy 96.1%, while the second best is the Random Forest. Moreover it was found that when the data are limited, methods such as extra randomized trees, random forest and deep neural networks can achieve much higher predictive performance than linear discriminant analysis, Gaussian naive Bayes, k-nearest neighbors, support vector machines or stochastic gradient descent.

Due to the relatively high accuracy scores of the Decision Tree in [44], we choose to utilize it since it is simple and can easily be interpreted. Interpretability is vital in our case, in order to obtain an understanding of the prediction reasoning for both behavioral increase and participant dropout. Moreover, we will compare Random Forest and Extra-Trees classifiers to check which model predicts best. This is due to the fact that these algorithms performed well in the physical activity domain as it was found in [14] and [41]. Another algorithm that has been widely used in Kaggle competitions [15], but not much in the physical activity domain, is the Extreme Gradient Boosting (XGBoost) [10]. We choose to use this technique, since it is quite accurate and fast and it is interesting to check if XGBoost can outperform other ensemble-based methods such as Random Forest and Extra-Trees.

3 Exploratory Data Analysis

This section will present the available datasets. Moreover, it will describe exploratory data analysis conducted in order to create informative plots that will provide a deeper understanding.

3.1 Data Description

The available experimental data, provided by the Active2Gether team, come from various sources and hence, the first challenge is the integration of the data in order to have a unified view of them. A summary and a short description of the data is given below.

- **Fitbit data:** For each user and for each submitted date the number of steps and floors a user took, as well as the amount of calories burned during that day.

- **User data:** For each user the condition group that the user belongs to is reported, the starting and ending date of the experiment and the number of days until dropout (84 if the user did not drop out).
- **Self-reported activities:** Minutes of active traveling (e.g., biking) and sports, as reported by the users in the experimental groups (A2G Full & Light) for all the experiment days.
- **Intake questionnaire data:** The answers of each user in the intake questionnaire, which took place before the start of the experiment. Includes demographic information about the users (gender, age, height, study, job), information about the users' current activity levels, questions about how competitive a user is, statements about attitude, intentions, social norm, self-efficacy, etc.
- **Follow-up questionnaire data:** The answers of the users in the follow-up questionnaire, which took place after the experiment had ended. Includes almost all the questions from the intake questionnaire plus some additional questions about perceived usefulness and usability of the Active2Gether application.

When it comes to data quality, there are only a few missing values in the follow-up questionnaire and some invalid Fitbit data caused by defective devices or devices that ran out of battery. Overall, the data quality is quite high.

3.2 EDA on Fitbit & User data

The first step of exploring the data is to create plots from the Fitbit data, since we have this kind of data for users from all the experimental groups. This will allow for a comparison between the different groups. Since steps, floors and calories are generally correlated features, we first create the correlation matrix between these three quantities:

Table 1: Correlation checking between the three available variables in the Fitbit data. We see that the variables are highly correlated.

	Steps	Calories	Floors
Steps	1.0	0.83	0.62
Calories	0.83	1.0	0.52
Floors	0.62	0.52	1.0

In Table 1 it can be seen that the three available variables have high positive correlations, hence we must select one variable to use in the modeling, as all of them measure physical activity but in different units. We use the step values, since they are more unbiased than calories, which are different for each individual. As an example, 5000 steps are associated with 1971 calories for user 186, while the same number of steps is associated with 1899 calories for user 195. Furthermore, floor values are not a good indicator for measuring the users performance, as not all users were in places with floors every day.

In order to find the distribution of the users' steps during the experimental period we first need to decide which step values will be considered as valid. We take as valid step values, only values larger than 1000, since step data lower than this level usually indicates a problem with the Fitbit device. We also take only step values submitted within the active experimental period of each user (approximately 12 weeks for all users). Figure 2 depicts the number of active users per month of experiment. As can be seen, March is the month with the least number of active users (6) as it is the first month and most of the users have not started the experiment yet, while June is the month with the largest number of active users (79) as it is in the middle of the experimental period.

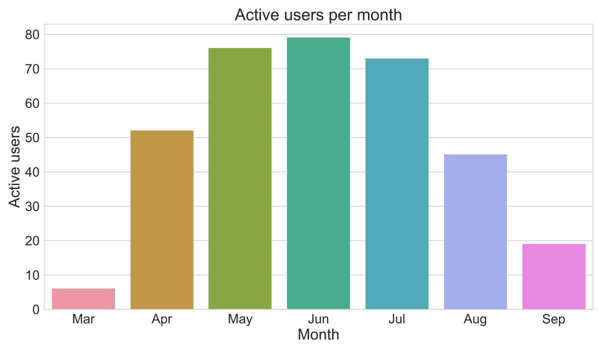


Figure 2: Number of active users per month of experiment.

Figure 3 depicts the mean number of steps per month. It is observed that there is an upward trend until July when the mean number of monthly steps starts moving downwards until September, which is the last experimental month. The total number of the participants is 92, however in Figure 2 it is shown that the maximum number of active users is 79. This is due to the fact that several users dropped out from the experiment. We find that 8 out of 23 users dropped from the A2G Full group, 3 out of 23 users from the A2G Light group and 24 out of 45 users from the Fitbit group, which gives us a total of 35 dropouts. The respective percentages are 33.3 % for the Full group, 13.04 % for the Light group and 53.3% for the Fitbit group, as can be seen in Figure 4.

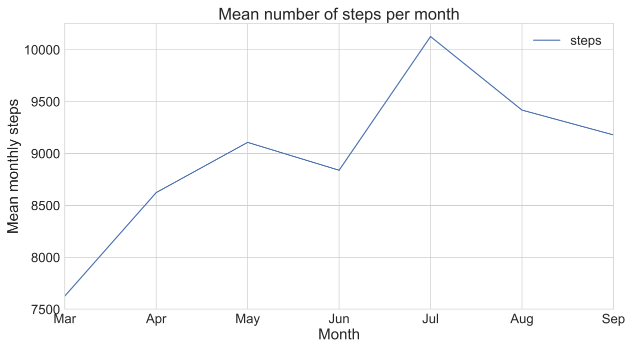


Figure 3: Mean number of steps per month of experiment

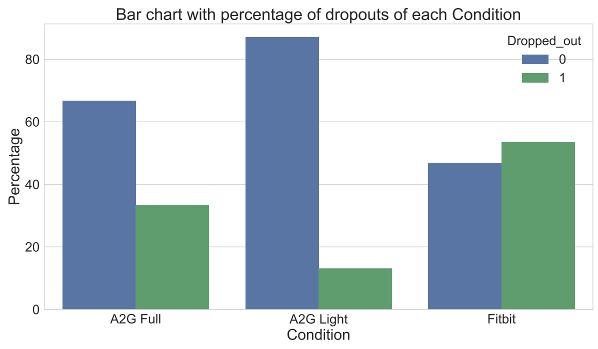


Figure 4: The dropout percentages for each condition

In order to check if the differences in the dropouts between each condition are significant, we apply a Fisher exact test [17] for all the three pairs of the experimental groups. Table 2 shows the results of the Fisher exact tests. It is observed that only the test between the A2G Light and Fitbit groups is lower than 0.05. Hence, we conclude that only those two groups have significantly different dropout percentages.

Table 2: P-values of the Fisher exact test for the significance in the differences of the dropout percentages for each condition.

Conditions	**Fisher P-value**
A2G Full & A2G Light	0.16
A2G Full & Fitbit	0.13
A2G Light & Fitbit	0.001

In order to further investigate the dropouts, we calculate the cumulative count of dropouts in the end of each experiment week. These counts are summarized in table 3, where it can be noticed that 5 users dropped out instantly (before week 1), 7 users had dropped in total after week 1, 9 users after week 2 and finally 35 users had dropped in total after the end of the experiment. It is noted that all the 7 users that dropped out until the end of week 1 belong to the Fitbit group.

Table 3: Cumulative count of dropouts when each of the 12 experiment weeks ends

Week	**Dropouts (cumulative)**
0	5
1	7
2	9
3	10
4	10
5	13
6	15
7	15
8	15
9	20
10	21
11	28
12	35

To explore the differences in the steps between each experimental category we find the mean steps of each non-dropped user and then we plot the distributions of the mean user steps for each experimental category. The results are in Figure 5. Here we take the mean of the steps for each user instead of all the daily step values, in order to have smaller variance in the values. We exclude the dropped-out users, since we want to compare the mean number of steps only for the users that remained active throughout the experimental period, in order for the comparison not to favor groups with smaller percentages of dropouts.

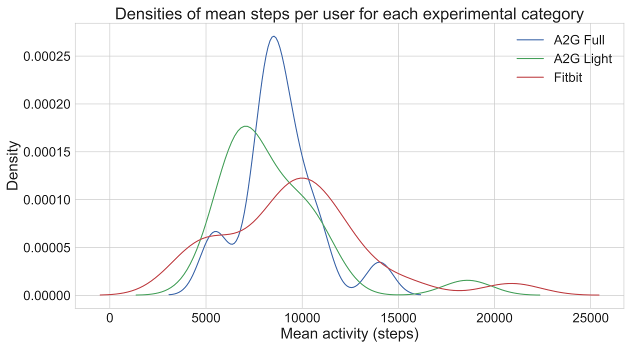


Figure 5: Distribution of the mean number of steps for the users in each group

In order to create the densities of Figure 5, we used Kernel Density Estimation (KDE) [40]. In this figure it can be observed that the distributions of the mean user steps for all the conditions seems to have the form of normal distribution. This is in line with the Central Limit Theorem, which states that the arithmetic means of n independent random variables tend to normal distribution as $n \to \infty$ [39]. To check if the distributions of the experimental conditions are indeed normal, we can apply a Shapiro-Wilk test for normality [42]. By doing so, we find that only the A2G Full and the Fitbit groups have normally distributed mean user steps. In order to verify this visually, we also plot the normal quantile plots of the mean user steps for each condition in Figure 6. It can be seen that the sample and theoretical quantiles of the A2G Full and Fitbit groups are on the diagonal, with only a small number of deviating values, while those of the A2G Light group deviate to a greater extent.

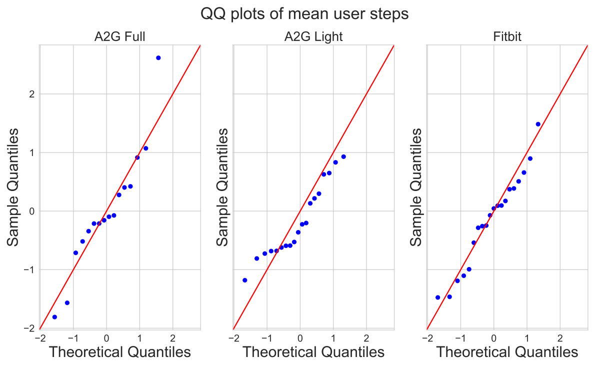


Figure 6: Normal quantile plots for the mean user steps for the experiment groups.

To visually compare the mean user steps for each condition, we provide the corresponding boxplots in figure 7. In this figure, the size of the boxes indicates the number of active users (non-dropped users) in each condition. The A2G Full group has the smallest number of active users (16), the A2G Light group has 20 users, while the Fitbit group has 21 active users. It can be noticed that the Fitbit group has the largest median (9868) and also the largest mean (9697), while the A2G Light group has the smallest median (7667) and the smallest mean (8510). It can also be seen that the Fitbit values are the most widely spread, something that corresponds the fact that the Fitbit group has the largest standard deviation (3927) in the mean step values. This means that the Fitbit group has both very inactive and very active users. The fact that it has very active users can be seen from the largest mean (not statistically significant though) compared to the other two groups. In the boxplots it is also observed that all the groups have at least one outlier.

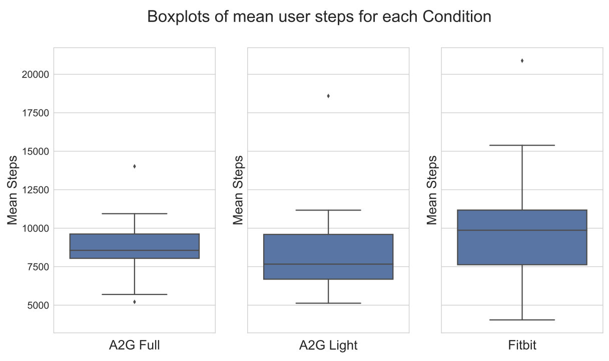


Figure 7: Boxplots of the mean number of steps for the users in each experimental group

In order to find whether the differences in the means between the three conditions are significant or not, we need to apply a statistical test, to test the null hypothesis that the samples originate from the same distribution. As we saw earlier, the mean user steps are not normally distributed for all the groups and hence, we cannot apply a one-way ANOVA test. However, we can use the Kruskal-Wallis test [30] which is the non-parametric equivalent of one-way ANOVA and does not assume a normal distribution of the residuals. The Kruskal-Wallis test can be used for comparison of two or more independent samples. By applying the Kruskal-Wallis we obtain $H = 2.38$ and a p-value 0.303 and hence we do not reject the hypothesis that the samples come from the same distribution. Due to this we conclude that the differences in the mean user steps between the three conditions, are not significant.

3.2.1 Identifying Behavioral Increase

An interesting topic for investigation is to find if there are users who have significantly increased performance levels at the end of the experiment, compared to the levels in the beginning of the experimental period. Ideally, we would use the entire duration of the experiment to check the change in the behavior. However, most of the users have several invalid step values, especially in the last weeks, hence we are not able to use the whole period to identify change in user behavior.

We use the step data to find the monthly progress for the users, by taking for every user the mean steps for each month of the experiment. As an example we provide the progress of two extreme cases of users, one that shows a downward trend and one that shows an upward trend. User 121 in Figure 8, with experiment period 6/4/16 - 7/7/16 is in the A2G Full group, while user 179 with experiment period 10/5/16 - 5/9/16 is in the A2G Light group and neither of them dropped out. From the plot we can see that user 121 started and finished the experiment one month earlier than user 179.

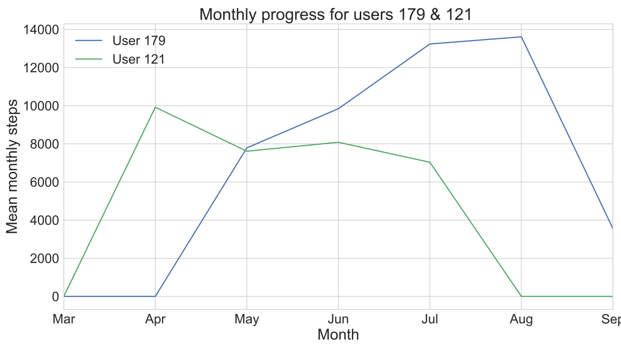


Figure 8: Monthly progress of users 121 and 179

In order to identify which users showed a significant increase in the behavior during the time that the experiment took place, we must specify starting and ending intervals in which we will check if the differences in the means of the step values are statistically significant. There are two parameters that need to specified:

1. the *length of the interval* with which to determine the mean of the steps
2. the *point in time* that signals the end period.

For our results to be valid we want to set our second measurement as latest as possible and at the same time to include as many valid users as possible. Next, we examine the distribution of the number of valid days for the non-dropped users. For all the users that did not drop out we find the number of days with a valid number of submitted steps and we plot their distribution using KDE as it is shown in Figure 9. We find that the valid number of days with step data for the non-dropped users has mean 63.78 and standard deviation 12.46. This means that on average we have almost 2 months of valid step data for each user.

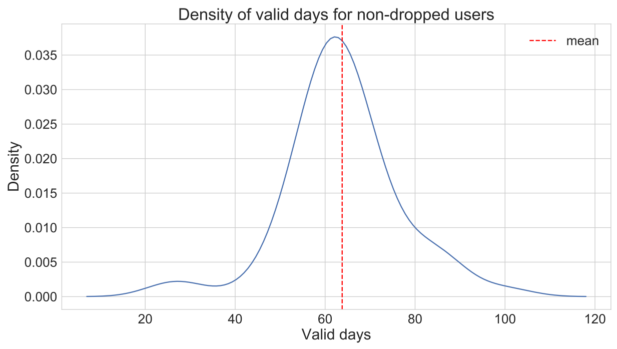


Figure 9: The density of the number of valid days for the non-dropped users

When it comes to the appropriate interval to consider, the options are 1, 2, 3 or 4 weeks. We reject the period of 1 week since 7 step values are not enough for the Wilcoxon signed rank test to provide valid results (we use this non-parametric test since most of the data do not follow a normal distribution). Moreover, due to monitoring effect the users might do more steps in the first 7 days and hence these 7 step values might not be representative of their actual activity levels. We also reject the period of 4 weeks, since it is too large, as the experiment lasted only 3 months. Moreover, many of the users have invalid values in their last weeks, something that leaves less than 3 months of valid step values. In order to check if a period of 3 weeks is appropriate we find the number of users that have less than 21 valid days (days with valid step data) for several possible cases. The results can be seen in Table 4. In this table it is shown that if we take as ending period, the weeks 10-11-12 (and as starting period weeks 1-2-3) then the number of users that have less than 21 valid days in this ending interval is 40. By taking into account that the number of users that did not drop out is 57, then 40 out of 57 is a very large proportion of invalid users and hence this is not an appropriate interval to check for behavioral increase. It can be seen that in order to get a relatively low number of invalid cases we should take weeks 8-9-10, however in this case we start our final measurement earlier than 2 months of active experiment, something that does not allow us to use the maximum possible interval.

A similar pattern of behavior is observed when we examine two-week intervals. We find that by taking as ending period weeks 11-12, again 40 out of 57 cases are invalid, while by taking weeks 10-11, 8 cases are invalid. Finally, if we take weeks 9-10 we see that only 3 instances are identified as invalid, which is the minimum so far, and at the same time the measurement is after 2 months of experiment.

Table 4: Number of invalid non-dropped users for several periods. A user is invalid if he has less than 21 valid step data for a 3 week period and less than 14 valid step data for a 2 week period.

	Ending Period	**Invalid cases**
3 week period	Weeks 10-12	40
	Weeks 9-11	17
	Weeks 8-10	6
2 week period	Weeks 11-12	40
	Weeks 10-11	8
	Weeks 9-10	3

Based on this analysis we conclude that the most appropriate interval to consider is 2 weeks and most specifically weeks 9-10, the first two weeks after two months of active experiment, which is an interval that fits well with the observation that for the non dropped users we have 2 months of step data on average.

After determining the proper interval and the points in time that we will check for behavioral increase, we find and store for each user the valid step values of weeks 1-2 and weeks 9-10. We create a variable `prog_increase` which is 1 if the mean of the weeks 9-10 is numerically larger than weeks 1-2 and 0 otherwise. Subsequently, to determine if the difference in the mean is significant, we apply for each user a Wilcoxon Signed Rank Test. This is a non-parametric alternative to the paired t-test for dependent samples, when the population is not assumed to be normally distributed [47]. For each user we create a binary variable `significance` denoting that if difference in the means of the starting and ending period is significant or not. Then, the users that showed significantly increased behavior are those that have both variables `significance` and `prog_increase` equal to 1. These users are presented in Table 5 along with the p-value of the Wilcoxon test, the difference in the means of the weeks 9-10 and weeks 1-2, the group they are in and a variable showing if they dropped out or not. As can be seen, for all the users who show a large positive difference in the means of the first and last weeks, there are 3 users from the A2G Full group, 1 user from the A2G Light group and 1 from the Fitbit group. Moreover, none of the users dropped out.

Table 5: Users with significantly increased behavior after a period of 2 months. The p - value here refers to the Wilcoxon signed rank test.

A2G id	**P - value**	**Difference in means**	**Condition**	**Dropped-out**
179	0.001	8000	A2G Light	0
190	0.035	3784	Fitbit	0
195	0.001	6594	A2G Full	0
199	0.030	5081	A2G Full	0
208	0.011	5659	A2G Full	0

In order to find the percentages of behavioral increase for each condition we create a cross tabulation for the 3 conditions and the variable `behavioral_increase` which is 1 if both `significance` and `prog_increase` are 1 and 0 otherwise. We find that from the Full group, 3 out of 21 users had significant behavioral increase, from the Light group 1 out of 22 had significant behavioral increase and from the Fitbit group 1 out of 45 showed significant behavioral increase. These proportions can be seen in figure 10. Note that in these calculations we include all the users, both dropped and non-dropped. If we exclude dropouts, then A2G Full has again the largest proportion (0.187), while the A2G Light and Fitbit have almost the same percentages (0.05 and 0.047 respectively).

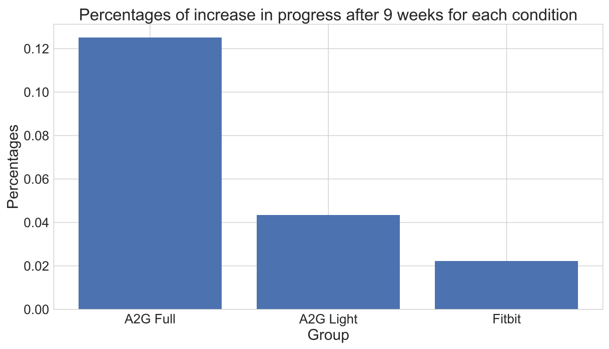


Figure 10: The percentages of the users that showed significant behavioral increase after 2 months, for each condition

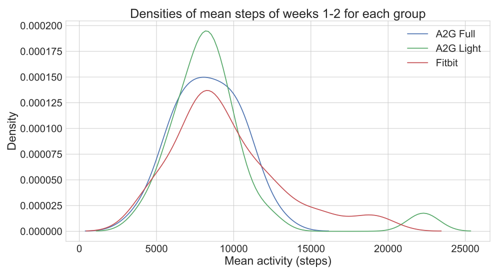
In order to check if the differences in the percentages for each condition are significant, we apply 3 times the Fisher exact test for all the possible pairs of the conditions. We find that all the p-values are much larger than 0.05 (0.6 between conditions Full and Light, 0.11 between conditions Full and Fitbit and 1 between conditions 2 and 3) and hence, we conclude that the differences in the proportions are statistically insignificant.

For exploratory purposes only, we applied the same methodology to find the users that had significant behavioral increase after 1 month (weeks 5-6). The results can be seen in Table 6. We observe that the only common user between the two tables is user 208, which is a user from the A2G Full experimental group.

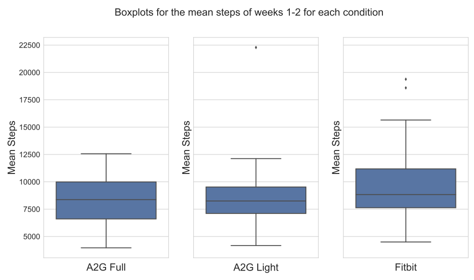
Table 6: Users with significantly increased behavior after a period of 1 month. The p - value here refers to the Wilcoxon signed rank test.

A2G id	P - value	Difference in means	Condition	Dropped-out
131	0.015	2085	A2G Light	0
184	0.006	5397	A2G Full	0
208	0.005	3352	A2G Full	0
210	0.003	8398	Fitbit	0

Next, we would like to check if there are any significant differences between the means of weeks 1-2 and weeks 9-10 between the three conditions. For the weeks 1-2 we include all the users except those that dropped out within the first 2 weeks. Figure 11a depicts the densities of the mean steps of weeks 1-2 for each condition. A Shapiro-Wilk normality test shows that only the mean steps for the Full group are normally distributed ($p = 0.904$ for the A2G Full, $p = 3.148 \cdot 10^{-8}$ for the A2G Light, $p = 0.013$ for the Fitbit), something that can also be observed visually. Figure 11b shows the corresponding boxplots and can be observed that all the conditions have almost the same levels of mean steps in weeks 1-2.



(a) Densities of mean steps for weeks 1-2



(b) Boxplots for mean steps for weeks 1-2

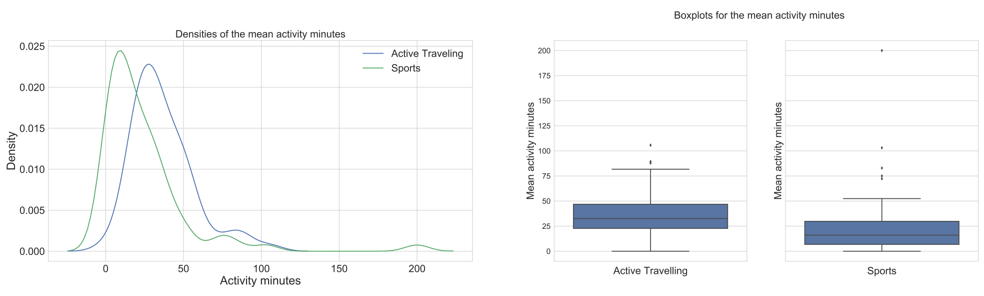
Figure 11: Densities and boxplots for weeks 1-2 for each condition

A Kruskal-Wallis test produced $H = 1.41$ and p-value $= 0.49$ and hence, there are no significant differences in the means between the three groups. This finding shows that the participants were indeed assigned randomly in one of the three groups, as none of the groups showed any significantly different performance in the first two weeks. The same procedure was also applied for weeks 9-10 and the results were the same, none of the groups showed any significantly different performance (in terms of mean user steps) after two months of experiment.

3.3 EDA on self-reported activity data

The self-reported activity dataset includes the minutes of active traveling and sports, as reported from the users in the A2G Full and Light groups throughout the experiment. The purpose of the exploratory data analysis in this dataset, is to apply a similar methodology as before in order to check if there are users with significantly increased activity levels after two months of experiment. Since this dataset concerns only the users in the experimental conditions, we have data for 46 users (in total there are 47 users in these groups but there were no data for the self-reported activity for one user).

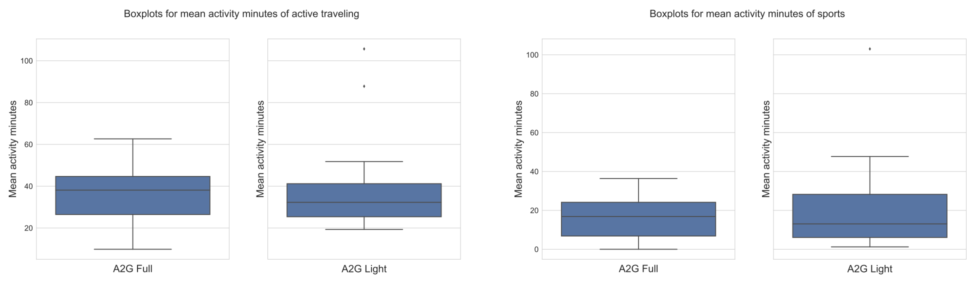
Firstly, we conducted a comparison between the mean activity minutes for active traveling and sports. We take the mean activity minutes for all the users and for both categories, and we plot their distributions as can be seen in Figure 12a. We observe that both distributions are right skewed with long right tails. A Shapiro-Wilk normality test reported p-values $1.4 \cdot 10^{-4}$ for active traveling and $2.9 \cdot 10^{-11}$ for sports, indicating that the densities are not normal, as expected. The comparison is more clear in Figure 12b where can be seen that the active traveling minutes have in general larger values than sports. Active traveling minutes have a mean of 36.8 minutes (std: 20.09), while sports have mean 24.05 minutes (std: 29.9). To check if the differences in the means are statistically significant we apply a Mann-Whitney U test which is a non-parametric test for independent groups and ordinal responses. Unlike a t-test, the Mann-Whitney does not require the assumption of normal distributions. The Mann-Whitney test reports $U = 3428$ and $p = 1.19 \cdot 10^{-6}$ and hence we conclude that the differences in the means are statistically significant. This indicates that the users spent more time traveling than doing sports.



(a) Densities of mean activity minutes *(b) Boxplots of mean activity minutes*

Figure 12: Densities and boxplots of mean activity minutes for active traveling and sports

Next, we conduct a comparison in the mean activity minutes for active traveling and sports between the experimental groups A2G Full & A2G Light, using only the non-dropped users. In the boxplots of Figure 13 can be seen that while the median of the A2G Full group is slightly larger in both cases, the groups have almost the same range of values. This is verified by conducting Mann-Whitney test for both cases, something that resulted in p-values much larger than 0.05. Hence, there are not significant differences in the mean activity minutes of active traveling and sports between the two groups.



(a) Active traveling minutes comparison *(b) Sports minutes comparison*

Figure 13: Boxplots of mean activity minutes for active traveling and sports, for the non-dropped users of the experimental groups

In order to check if there are users with significantly increased activity in weeks 9-10 (we use the same interval as with the step data) compared to weeks 1-2 we apply the following procedure. For each user (both dropped and non-dropped) we create an activity score that combines minutes of active traveling and minutes of sports. Since doing sports is typically more intense than active traveling, we want to apply a higher weight to minutes of sports and hence, we use the following score function [2]:

$$score = 1 \cdot travel\_minutes + 2 \cdot sports\_minutes \tag{1}$$

For each user we create a score of total activity using the score function (1). Then we create a variable `act_increase` which is to 1 if the mean of activity score for the weeks 9-10 is larger than the one of weeks 1-2. Afterwards, for each user we apply a Wilcoxon Signed rank test in order to check if the difference in the means of weeks 1-2 is statistically significant from weeks 9-10 and we create a variable `significance` which is set to 1 if the difference in the means is significant and to 0 otherwise. The users with significantly increased activity are those who have both `act_increase` and `significance` equal to 1 and can be seen in Table 7. Note that the user 152 is a dropout, however since he/she dropped out after 76 days of experiment and our last measurement is after 2 months of experiment (60 days), we consider him/her as valid.

[2]Note that for the weight of sports minutes, we also tried to use 3 instead of 2, but without any change in the results.

From the table we observe that both users are in the A2G Full group, however none of the users is common with the users in Table 5 or Table 6.

Table 7: Users with significantly increased activity score after a period of 2 months. The p - value here refers to the Wilcoxon signed rank test.

A2G id	P - value	Difference in means	Condition	Dropped-out
152	0.00	29	A2G Full	1
222	0.03	55	A2G Full	0

We also applied Mann-Whitney test for the difference in the mean activity scores between A2G Full & A2G Light, which produced $U = 149$ and p-value $= 0.73$, hence the difference in the mean scores is not significant.

4 Modeling

In this section, supervised machine learning algorithms are used to create predictive models for behavioral increase as well as dropout. Supervised techniques are used when there is available a training set of correctly identified observations. Since both the target variables are categorical, classification algorithms are utilized. Classification belongs to the field of pattern recognition and deals with the problem of assigning a category to each input variable vector, based on a series of features used for training.

In order to create models that predict the behavioral increase and the dropout, we first need to extract useful features about the users. To achieve that, we use the intake questionnaire (filled in before the start of the experiment) which includes questions about demographics, physical activity levels of the users, their intention to be physically active in the next weeks and psychological questions related to their motivation to increase their physical activity levels. After translating the questionnaire from Dutch to English, in total 90 features are extracted. The vast majority of the features are categorical, such as gender, job and sports (whether the user does one sport, several sports or no sports at all). However, there are several numerical features too, such as the Body Mass Index (BMI), the motivation of a user to move (in a scale from 0 to 10) and the number of days the user had at least 30 minutes intensive exercise in the week before the questionnaire. A complete list of the extracted features can be found in Appendix A.

Since all the users answered all the questions in the intake questionnaire, there are no missing values in the extracted dataset. The next step is the integration of the extracted features in the user dataset. To achieve that, we keep from the user dataset only the columns `Dropped_out` and `behavioral_increase`. We then merge these two columns with all the 90 features extracted from the intake questionnaire to create a combined dataset that will be used to train and test our algorithms. Note that from the user dataset we drop variables created by the Fitbit data, such as the mean number of steps in weeks 1-2, since these data are not available before the start of the experiment and hence we could not use them to predict in new data. This is due to the fact that we want to create models that are able to predict behavioral increase and dropout, before a new experiment starts.

Each of the following subsections includes theoretical background about the machine learning algorithms, information about the parameter tuning procedure as well as the validation of the models. In 4.1 we present the theory, the hyper-parameter tuning and the tree visualization for each created model for predicting behavioral increase. More specifically, subsections 4.1.1 - 4.1.3 include information about the Decision Tree, the Random Forest and the Extra-Trees. Subsection 4.1.4 reports the most important features of the created models and also provides an explanation about why these features are identified as important by the classifiers. Subsection 4.1.5 introduces the evaluation metrics and reports the results of the 5-Fold cross validation and a modified version of repeated cross validation. The classifiers are compared and the one with the largest F_1-score is identified. Section 4.2 concerns the prediction of dropout. More specifically, subsections 4.2.1 - 4.2.3 include information about the Decision Tree, the Extra-Trees and the XGBoost. Subsection 4.2.4 provides an explanation of the Decision Tree and reports the most important features of the classifiers. Each important feature is further investigated to understand why it is important for the corresponding model. Lastly, in subsection 4.2.5 the classifiers are evaluated and compared using 10-Fold cross validation and a modified version of repeated cross validation.

4.1 Predicting Behavioral increase

In order to predict behavioral increase, we create three tree-based classifiers: Decision Tree, Random Forest and Extremely Randomized Trees. Since the users who showed significantly increased behavior are 5 out of 92, the percentage of the minority class is only 0.54% and hence, the data are highly imbalanced. Due to this, accuracy cannot be used as an evaluation metric, since a model that identifies all the instances as the majority class, will give a very high accuracy but will be of no practical use. A metric that is commonly used in case of imbalanced data is the area under the receiver operating characteristic (ROC) curve. However, in [25] several metrics were evaluated and it was found that when the data are skewed, the area under the ROC curve may mask poor performance. Since in our case the data is highly skewed, we will tune our classifiers in order to optimize the F_1 *score* which is the harmonic mean between *precision* and *recall*: $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$.

4.1.1 Decision Tree

Decision trees belong to a group of rule-based classification methods. This kind of classifier inspects the raw data for rules and structures that are common in a target class and then these rules become the basis for decision making. The cause of a certain decision can be reconstructed and is interpretable without statistical knowledge. The great interpretability and the fact that they require little data preparation (no dummy variables need to be created as for example in Logistic Regression or in XGBoost) were the main reasons for choosing this classifier.

A well-known decision tree algorithm is CART (Classification And Regression Tree) which is a binary splitting tree, where each non-leaf node has exactly two outreaching branches and was first introduced by Breiman et. al in [8]. Since the dropout response vector consists of categorical data we apply a classification tree algorithm which finds the binary splits as follows. For each numerical predictor, it orders its values from smallest to largest and looks at the $N-1$ possible splits between adjacent values. In our case N refers to the number of users for which we have data. For each categorical predictor, it evaluates all possible $2^{K-1}-1$ splits of the category nodes into two groups, where K refers to the number of categories in each predictor. The best split is then chosen to minimize the *impurity* of the two resulting subsets that are formed. A decision tree classifier in Python [3] supports two impurity criteria, namely Gini index and Entropy, which are given by the following formulas:

$$Gini = 1 - \sum_j p_j^2 \qquad\qquad Entropy = -\sum_j p_j log_2 p_j \tag{2}$$

where p_j is the probability of class j. Gini index is an impurity measure and describes the dispersion of a split. It reaches its maximum when the data in the node are equally distributed across the categories, and its minimum if all data belong to the same category in the node [45]. Entropy quantifies the homogeneity of categories into a node

[3]Created using the `DecisionTreeClassifier` from the *scikit-learn* python package (v. 0.18.1).

and reaches its maximum if all categories in a node segment are equally distributed. The selection of the best criterion to use is determined in the parameter tuning of the classifier.

In order to find the best features for the decision tree, we first tune a decision tree classifier by applying random search [4], and we try N combinations for the specified ranges in each parameter of the decision tree. The best parameters are found by applying 5-fold cross validation. After finding the best parameters for this first decision tree (First DT), we fit the model to the whole dataset and we use the `SelectFromModel` function to find the features with the highest importance weights assigned from the fitted classifier. Note that if we choose to optimize F_1-score in this first classifier, only 1 feature is identified as important from the `SelectFromModel` function and the subsequent score results are low. Due to this, we optimize this first classifier in terms of ROC AUC by trying a large number of different parameter combinations (we tried 2000 combinations) and the best parameters can be seen in Table 8. Note that *max features = None* means that the model considers N features when looking for the best split, where N is the number of available features in the data set, in this case 90. In order to handle the class imbalance we use the `class_weight = 'balanced'` parameter in the `DecisionTreeClassifier` routine. This mode uses the values of the response vector to automatically adjust weights inversely proportional to class frequencies in the input data . Note that we also used Synthetic Minority Over-sampling Technique (SMOTE) [9] to handle class imbalance, however no better results were observed. Moreover, in order to be able to equally compare the classifiers that we build, we use in all the models the seed parameter `random_state = 99` (arbitrary value). This parameter affects the randomness in the classifier and by using a constant value it is guaranteed that we will always obtain the same results.

Table 8: Hyperparameter tuning of the Decision Tree for the behavioral increase

Hyperparameters	**First DT**	**Best DT**
criterion	entropy	entropy
max depth	4	4
max leaf nodes	7	7
min samples leaf	1	1
min samples split	3	3
max features	None	4

Using these settings, 5 out of 90 features are identified as important by the decision tree: `challenge_feeling,max_cycle_time,sports_look_better,app_resources` and the `friends_think_exercise` feature. We then create a data set which consists only of these features, and we use again random search with 2000 iterations, to find the parameters that optimize F_1-score in this smaller dataset. Alternatively, we can use a grid search, to check all the possible parameter combinations, however this procedure is very computationally expensive. The best parameters for the final decision tree (Best DT) can be seen in Table 8. It is observed that the parameters are the same as in the First DT, with the only difference that the *max features* parameter is set to 4.

In order to find the rules that the Decision Tree uses to predict behavioral increase, we plot its structure in Figure 14. For the given features, the Decision Tree evaluates all the possible splits and finds those that minimize the impurity criterion (in this case entropy). This can be seen in Figure 14, as all the leaf nodes have 0 entropy.

[4]Using the `RandomizedSearchCV` routine from the *scikit learn* python package (v. 0.18.1).

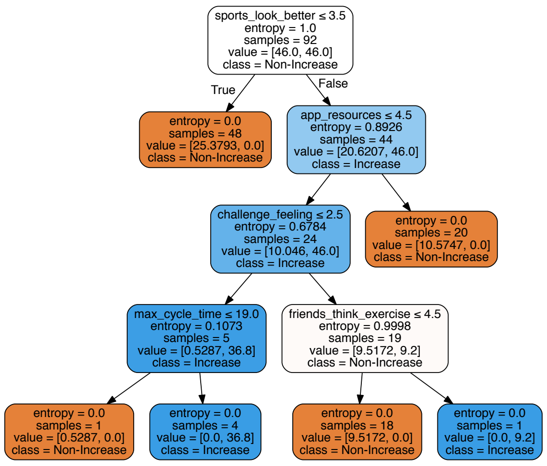


Figure 14: Visualization of the Decision Tree with the best parameters (Best DT)

4.1.2 Random Forest

The next algorithm to be used for predicting behavioral increase is the Random Forest, also known as Random Decision Forest [23]. Random Forest is an ensemble method that constructs a collection of decision trees and combines them in order to achieve higher predictive performance than the individual trees. In the case of classification, the ensemble of the decision trees vote for the most popular class. For regression problems, the responses of the trees are averaged in order to estimate the dependent variable [8]. Besides the fact that the algorithm is generally simple, it is also recognized for its accuracy as well as for its ability to handle high-dimensional feature spaces and samples of small size [7]. We choose to use this classifier for two reasons. Firstly, we want to check if this model can achieve higher performance than the individual Decision Tree which we built earlier. The second reason is that decision trees tend to overfit in the training set and hence random forests can be used in order to avoid this [22]. This is due to the fact that the Random Forest algorithm repeatedly selects a random sample with replacement of the training set and then uses these samples to fit the sub-trees. This bootstrapping procedure decreases the model's variance without increasing the bias, something that typically results in better predictive performance [8].

A Random Forest in Python [5] allows for a lot of parameter tuning and also supports the `class_weight='balanced'` parameter which we use to handle the class imbalance. In order to create a dataset with the most important features to fit our classifier, we first use the `SelectFromModel` function in the same way as we did with the Decision Tree. However, this function identifies 36 features as important, something that results to poor predictive performance (in terms of F_1-score) even after parameter tuning. This is due to the fact that many of the selected features are highly correlated, as can be seen in the correlation matrix of Figure 15. In order to tackle the problem of highly correlated variables in a random forest, Gregorutti et. al [19] use an algorithm called recursive feature elimination (RFE). This technique recursively eliminates the features and selects those that contribute the most to predicting the target attribute [21]. RFE can be performed in Python using the `RFE` function from the `feature_selection` package in *scikit-learn* (v. 0.18.1). In this function, we can set the number of the features to be selected. Several possible values were tried (5, 10, 15 and 20) and the best results were obtained by choosing 10 features.

[5]Created by the `RandomForestClassifier` routine from the *scikit-learn* package (v. 0.18.1).

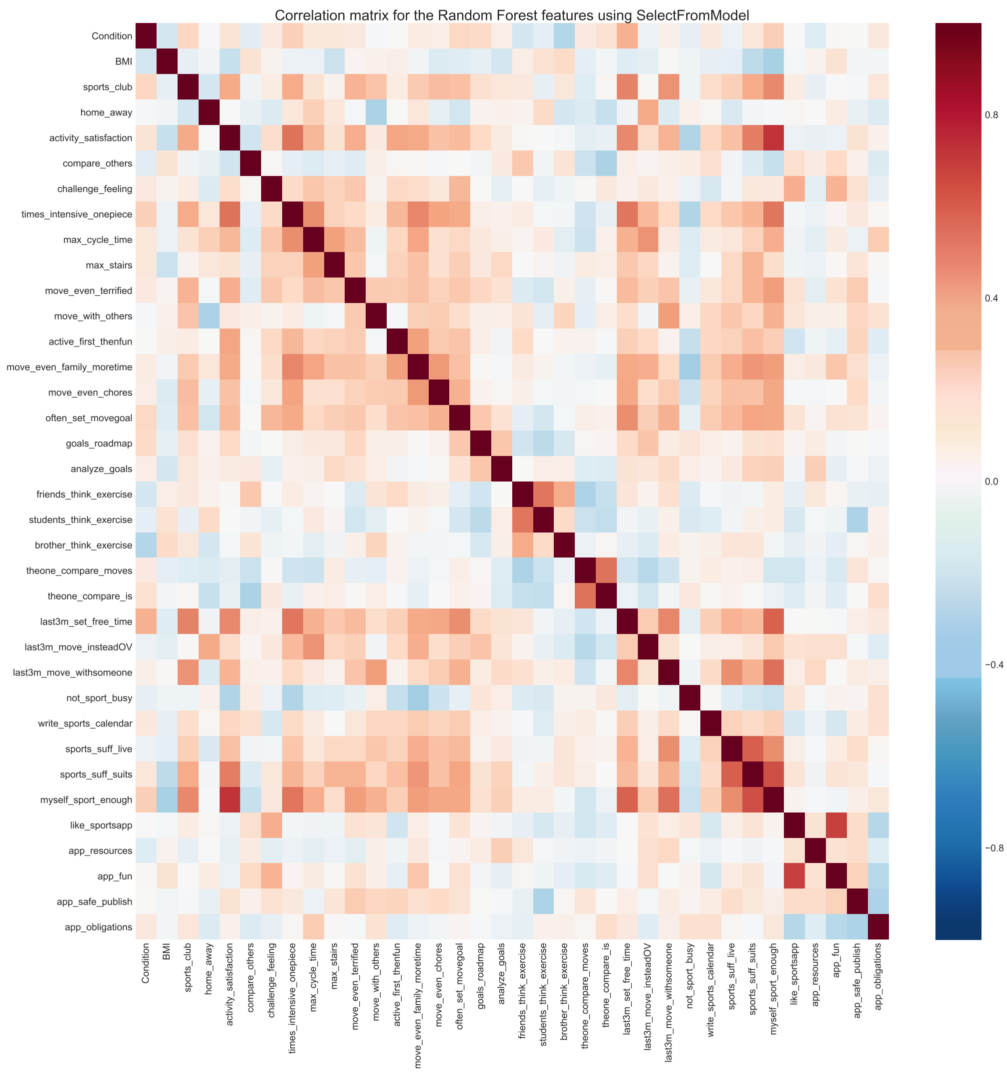


Figure 15: Correlation matrix of the features identified by the SelectFromModel routine

In Figure 16 we plot again the correlation matrix of the 10 best features identified by RFE. It can be observed that in this case, the majority of the variables have low correlations. After creating a dataset containing only the 10 important features, we tune the parameters of the random forest by applying random search with 2000 iterations and we find the best identified parameters. Alternatively, we can use a grid search to try all the possible parameter combinations, however this is very computationally expensive due to the large number of possible combinations. We need to mention that due to the fact that we have only 5 positive instances and a small dataset, 2000 combinations of parameters are adequate in order to maximize F_1. We also tried to apply random search with more than 2000 combinations, however no further score improvements were observed. The best parameters for the Random Forest can be seen in Table 9. Note that the number of estimators indicates the number of trees in the forest. In this case, the Random Forest consists of 46 Decision Trees.

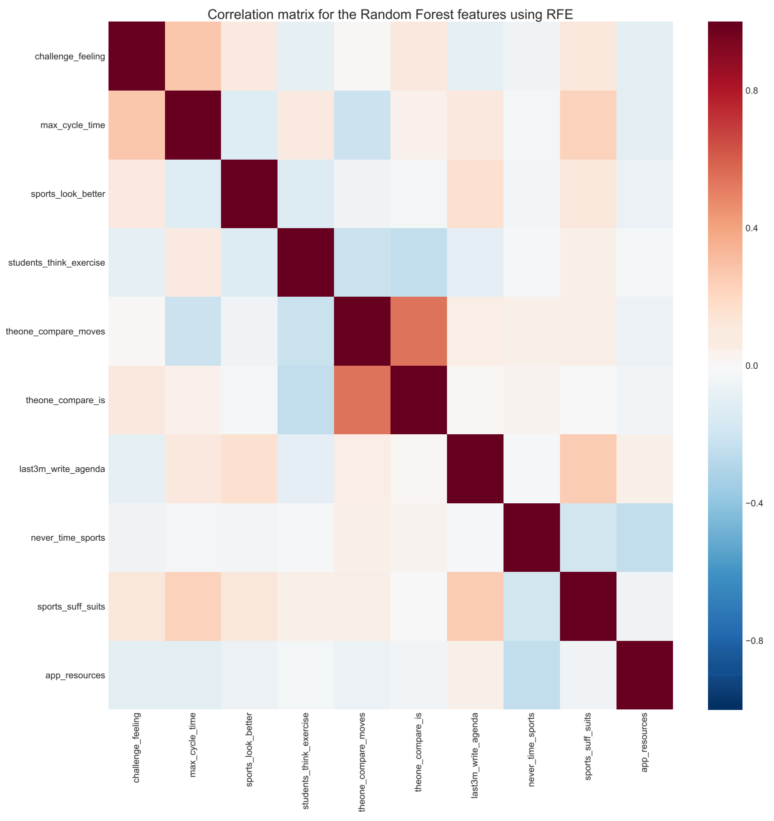
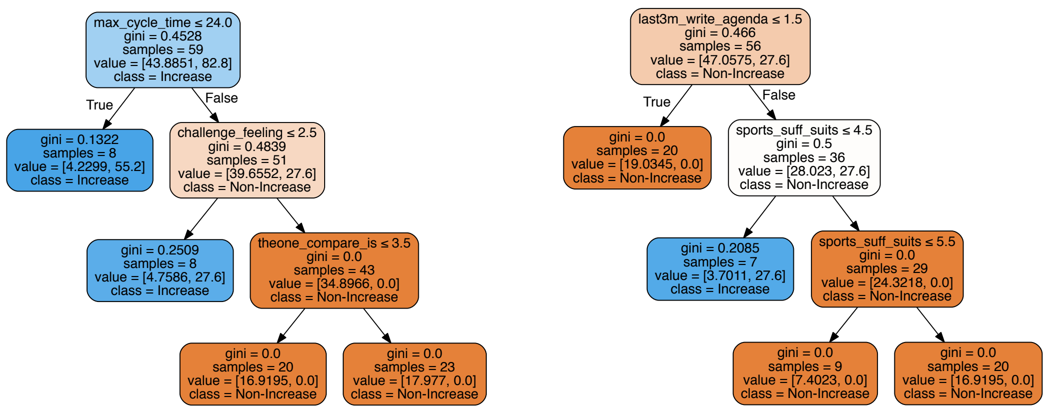


Figure 16: Correlation matrix of the features identified by the Recursive Feature Elimination for the Random Forest

Table 9: Hyperparameter tuning of the Random Forest for behavioral increase

Hyperparameters	**Value**
criterion	gini
number of estimators	46
max depth	7
max leaf nodes	14
min samples leaf	7
min samples split	16
max features	2

To get an idea of how the trees in the Random Forest look, we plot the first 2 (out of 46) Decision Trees in Figure 17. In this figure can be seen that the trees use features that have low correlations according to Figure 16. The same goes for the majority of the trees in the Random Forest. The features `theone_compare_moves` and `theone_compare_is` which are the most correlated, were used in the same tree only twice. Moreover, in Figure 17 can be seen that each of these trees uses different features. This is done by the algorithm in order to create uncorrelated trees, since the lower the correlation between the trees, the lower the error rate [8].



(a) *(b)*

Figure 17: The first two Decision Trees of the Random Forest

4.1.3 Extremely Randomized Trees

The last algorithm that we use to predict behavioral increase is the Extremely Randomized Trees (Extra-Trees). This is another tree-based ensemble technique, used both for classification and regression. There are two main differences between the Extra-Trees and the Random Forest. While the Random Forest evaluates all the possible splits and finds the best one, the Extra-Trees choose cut-points fully at random. Moreover, Extra-Trees use the whole learning sample to train the trees, while on the other hand Random Forests use a sample taken by bootstrapping. The reasoning behind the explicit randomization of the cut-point is that it should be able to reduce variance to a greater extent, compared to methods that use weaker randomization schemes. The motivation behind the usage of the full original learning sample instead of a bootstrap replica is to minimize bias. The Extra-Trees algorithm is considered to be more computationally efficient than other tree-based ensemble methods, due to the simplicity of the node-splitting procedure. Moreover, in many cases, Extra-Trees seem to outperform Random Forests and single CART trees, in terms of average error [18].

The Extra-Trees classifier in Python [6] has almost identical parameters with the Random Forest classifier with the only exception that in the latter, the default value for the `bootstrap` parameter is *True* (since Random Forests use bootstrap samples to fit the trees), while in the Extra-Trees classifier the default value is *False*. To deal with the class imbalance we use once again the parameter `class_weight= 'balanced'`. In order to select the best features to fit our classifier, we first use the `SelectFromModel` routine. However, this function identifies 26 features as important and the performance results in terms of F_1-score are relatively low. To achieve better results, we use the RFE technique as we did with the Random Forest. We tried several possible values for the number of features to be selected by the RFE (5, 10, 15 and 20) and the best results were obtained by selecting 5 features. The features identified as important by the RFE are: `home_away`, `sports_look_better`, `more_than_1goal`, `theone_compare_is` and `app_resources`. We plot the correlation matrix between those 5 features in Figure 18, where can be seen that the features have low correlations between each other. The largest (negative) correlation is observed between the features `theone_compare_is` and `home_away` (-0.23) while the rest of the correlations are less than 0.11 (absolute value).

[6]Implemented using the `ExtraTreesClassifier` function from the *scikit-learn* python package (v. 0.18.1).

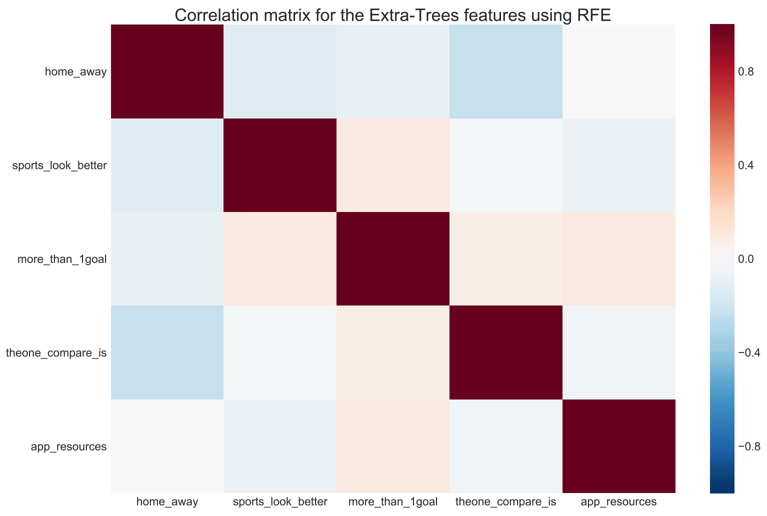


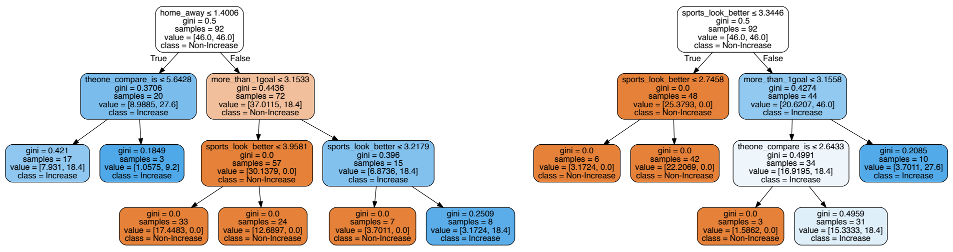
Figure 18: Correlation matrix of the features identified by the RFE routine for the Extra-Trees

After fitting the model to the dataset containing these 5 features, we tune the parameters by applying random search with 2000 iterations, as we did with the Decision Tree and the Random Forest. The best parameters found by this procedure can be seen in Table 10.

Table 10: Hyperparameter tuning of the Extra-Trees for behavioral increase

Hyperparameters	**Value**
criterion	gini
number of estimators	96
max depth	12
max leaf nodes	15
min samples leaf	3
min samples split	9
max features	2

In Figure 19 we plot the first two trees of the Extra-Trees classifier. We observe that while those two trees use the same features, their prediction reasoning is different, in order for the trees to have the least possible inter-correlation. In this figure it is also visible the fact that the Extra-Trees choose the cut-point for the split completely at random. This can be seen from the decimal values in each splitting criterion. By doing a comparison between the trees of the Extra-Trees and those of the Random forest in Figure 17, we see that in the latter the splitting criteria have either integer values or one decimal of 0.5. This is because the Random Forest checks all the possible splits and chooses the best one.



(a) *(b)*

Figure 19: The first two trees of the Extra-Trees classifier

4.1.4 Feature Importance & Tree explanation

Firstly, we provide an explanation of the Decision Tree in Figure 14 in order to understand the reasoning behind its predictions. By starting at the top node of the tree, we see that the first split criterion is *sports_look_better* $\leq$ *3.5*. Since this is the first node, the samples included are 92, equal to the total number of users in the dataset. The parameter `value = [48.0, 48.0]` indicates that this condition is true for 48 users and false also for 48 users. This is why this node has entropy equal to 1. The condition *sports_look_better* $\leq$ *3.5* is true if a user answered "do not agree at all", "do not agree" or "agree" and false otherwise. If this condition is false, then we move to the right node, which checks the `app_resources` feature. This condition is true if the user answered anything but "totally agree" for the statement about the app resources. In this case, we move to the left node and the Decision Tree checks the `challenge_feeling` variable. The condition is true if the user replied that he or she "never" or "rarely" gets a challenging feeling when people at work or study perform better than they do. If this is the case, then we move to the left node, where the `max_cycle_time` variable is being checked. This condition checks if a user answered that he or she is prepared to cycle (instead of taking the public transportation) if the maximum cycling time is less than 19 minutes. In the case that this condition is true, then we move to the left node and the user is classified as *non-increase* user. We observe that this is a pure leaf node, as there is only 1 user that satisfies all the aforementioned criteria, and this user has not shown an increase in his or her behavior.

In order to obtain an understanding about the most important features for the decision tree, we plot the feature importances of the Best DT in Figure 20. The importance of a feature is computed as the total reduction of the criterion (in this case entropy) brought by that feature. We see that the most important feature is the `sports_look_better` which indicates if the users agree with the fact that physical activity helps them to look better. The high importance of this feature is due to the fact that all the users who showed significant increase in behavior, answered that they totally agree with the statement that physical activity helps them to look better, as it is shown in Figure 21. This pattern shows that the improvement of the physical appearance was one of the motivations for the users to increase their activity levels. Moreover, all the 5 users that had significantly increased behavior, answered that they "somewhat agree" with the fact that they have the necessary resources to use a sports app as it is shown in Figure 22. This seems reasonable since it is essential for a user to have the necessary resources to use the app, in order to show an increase in his or her behavior. This pattern explains why `app_resources` is the second most important feature for the decision tree. The features `friends_think_exercise, challenge_feeling` have also relatively large importances while `max_cycle_time` has the lowest feature importance.

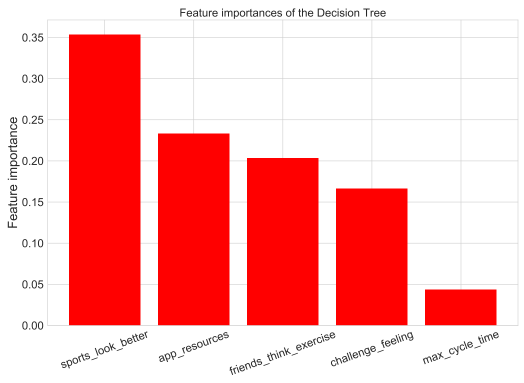


Figure 20: Feature importances of the decision tree with the best parameters (Best DT)

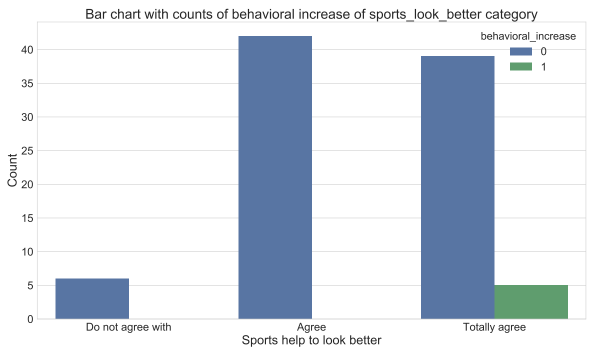


Figure 21: Bar chart showing the relation between behavioral increase and the sports_look_better feature.

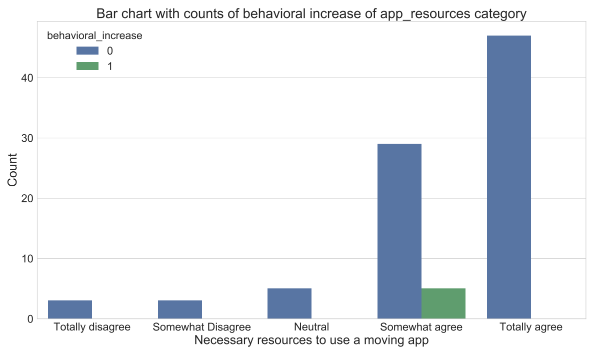


Figure 22: Bar chart showing the relation between behavioral increase and the app_resources feature.

Concerning the 10 best features identified by the Random Forest, we find and plot the feature importances in Figure 23. The most important feature for the Random Forest is the `challenge_feeling` variable which indicates how often a user gets a challenging feeling when people at work/study perform better than they do. The reason for the high importance of this feature can be seen in Figure 24. It is shown that 4 out of 5 users with increased behavior answered that they rarely have this challenging feeling, while 1 user with increased behavior answered that he/she always has this feeling. This finding indicates that the majority of the users with increased behavior are not sensitive to social comparison.

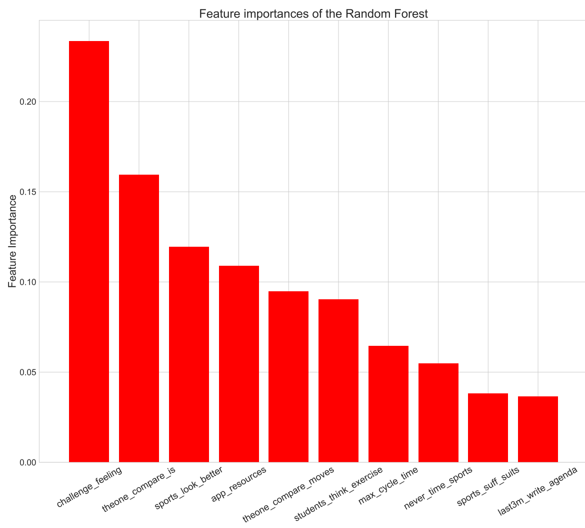


Figure 23: Feature importances of the Random Forest with the best parameters

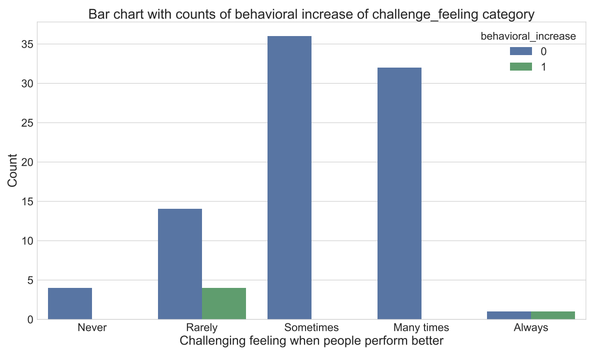


Figure 24: Bar chart showing the relation between behavioral increase and the challenge_feeling feature.

The second feature in terms of importance, is the `theone_compare_is` which shows if the users compare themselves with people that are less or more healthy than they are. In Figure 25 can be seen that 1 user replied that he/she compares himself/herself with healthier people, 2 users compare with much healthier people and 2 users did not provide an answer in this question. We observe that the majority of the users with increased behavior compare themselves with people with higher health levels, something that might enhanced their motivation to increase their activity levels.

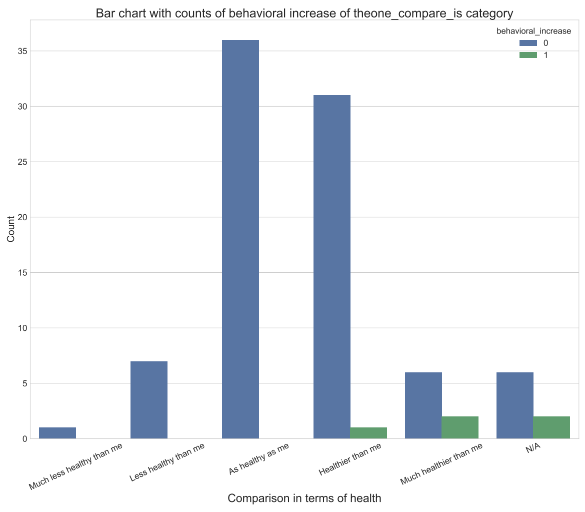


Figure 25: Bar chart showing the relation between behavioral increase and the theone_compare_is feature.

The rest of the features have lower importances, with `last3m_write_agenda` to be the least important of the ten important features. By comparing Figures 20 and 23 we see that the following features are common between the Decision Tree and the Random Forest: `challenge_feeling, sports_look_better, app_resources` and `max_cycle_time`.

When it comes to the important features of the Extra-Trees, we also plot the feature importances of the classifier with the best parameters in Figure 26. We observe that the most important feature according to the Extra-Trees is the `sports_look_better`, which also was the most important for the single Decision Tree. By comparing the feature importances of the three classifiers, we see that the features `sports_look_better` and `app_resources` are identified as important by all the classifiers. This is due to the strong patterns that observed for these two variables in Figures 21 and 22. Moreover, the feature `theone_compare_is` is important for both the Random Forest and the Extra-trees.

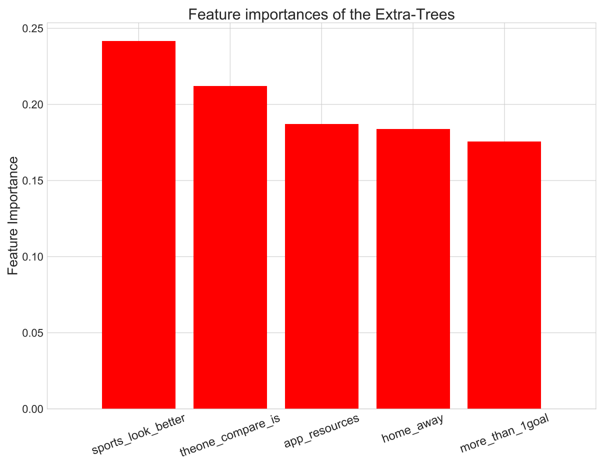


Figure 26: Feature importances of the Extra-Trees with the best parameters

4.1.5 Results & Evaluation

For the evaluation of the classifiers' performance, we will make use of several evaluation metrics that are appropriate for binary classification problems. We check both the F_1-score and the ROC AUC, however we will compare the classifiers in terms of F_1, as the data are highly imbalanced and in this case ROC AUC may mask poor performance as mentioned in Section 4.1. In order to understand what each evaluation metric represents, a brief explanation of the metrics is provided below.

- *Precision* is the number of true positives (TP), which indicate the number of items correctly labeled as belonging to the positive class, divided by the total number of elements labeled as belonging to the positive class:

$$Precision = \frac{TP}{TP + FP}$$

- *Recall* is the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives):

$$Recall = \frac{TP}{TP + FN}$$

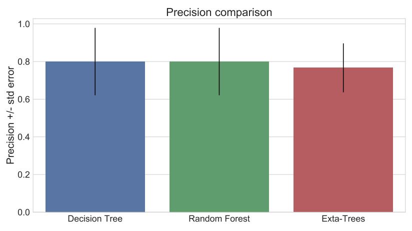
- F_1*- score* is the harmonic mean of precision and recall:

$$F_1 = 2\frac{Precision \times Recall}{Precision + Recall}$$

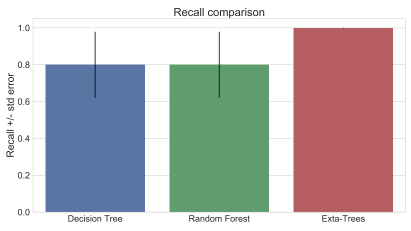
- *ROC AUC* is the area under the receiver operating characteristic (ROC) curve. This curve can be created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various thresholds.

In order to compare the performance of the classifiers, we use *K-Fold cross validation.* During this procedure, we randomly split the whole data set into k equal sized folds, and we use one of these folds for testing the model and $k - 1$ folds for training. We repeat this procedure k times and each of the k folds is used as validation data exactly once. After this procedure, we average the k results in order to produce a single estimation of the model's predictive performance. While it is common practice to use 10-fold CV [33], in our case we can only use 5-fold CV since we only have 5 positive instances [7]. In the cross-validation function we can set among others, the number of folds for the cross validation and also the evaluation metric. In Figure 27 can be seen the results of the cross validation for various evaluation metrics along with the standard error computed as $SE = \frac{\sigma}{\sqrt{n}}$, where σ is the standard deviation obtained by the cross validation and n is the number of folds, which in our case is 5. Note that the results of Figure 27 determined the choice of the hyper-parameters mentioned in the earlier subsections.

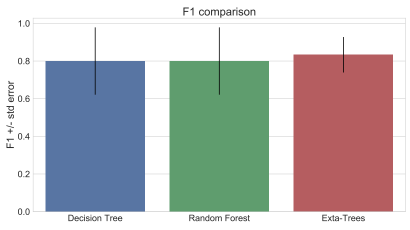
[7]We can apply K-Fold CV in Python using the `cross_val_score` routine from the *scikit-learn* package (v. 0.18.1)



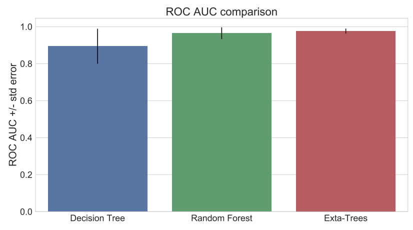
(a) Precision



(b) Recall



(c) F_1-score



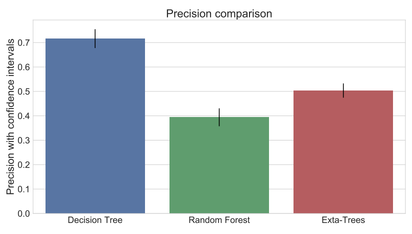
(d) Area Under the ROC Curve

Figure 27: 5 - Fold cross validation results for 4 metrics

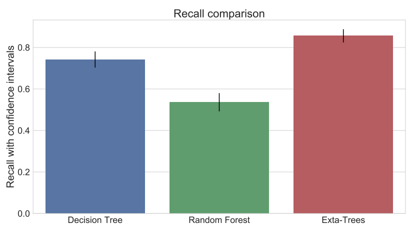
In Figure 27a can be seen that both the Decision Tree and the Random Forest achieved an average precision of 0.8, while the Extra-Trees scored 0.76. However, in terms of recall, Figure 27b shows that the Extra-Trees algorithm outperformed both the Decision Tree and the Random Forest, as it achieved a recall of 1.0. When it comes to the F_1-score, in Figure 27c it is depicted that the Extra-Trees achieved slightly larger F_1 (0.83) compared to the other two classifiers (0.8). Finally, as can be seen in Figure 27d the Extra-Trees also achieved a slightly larger ROC AUC (0.976) compared to the Random Forest (0.964) and the Decision Tree (0.894). By comparing the ROC AUC with the F_1 values, it is observed that the former are much larger. This is due to the fact that the F_1 score is affected by the skewness in the data, while the ROC AUC is not [25]. Due to the small differences in the scores, Mann-Whitney U tests showed that none of the differences between the models is statistically significant.

However, due to the fact that we used a specific random seed (99) when we built our classifiers, it is important to check the stability of the models by cross-validating with different seeds. A generalizable model should be robust in the choice of seed. To check the models' stability, we apply a modified version of the *Repeated 5-Fold Cross-Validation*. In the original *repeated* cross validation [46], the seed was changed in order for the CV procedure to have a different random partitioning of our data set in each iteration of the K-Fold cross validation. This procedure is used due to the high variance of the result of a single cross validation. In our modified version, we change the `random_state` in the classifiers and we obtain the 5-fold CV results in a recursive way. More specifically, we apply 5-fold cross validation in all three classifiers using 100 different seeds (from 90 to 190), something that results in 500 scores for each model. By doing this, we see how the classifier changes its predictive performance when we apply slight changes in the model training. The idea for this procedure came through a webinar about combining machine learning and optimization [2], in which they change the seed of the classifier in order to create a range of predictions to simulate the behavior of their solution. The reason why we used this modified version of repeated cross validation instead of the original one, is due to the small amount of available data. Since the models are trained only in a few instances, they are prone to overfit on one specific seed. This means that with one seed they might produce high F_1-score but with another seed the results could be much worse. Hence, with this modified version of repeated cross validation we are able to check how the models perform with a large number of different seeds, something that gives us a more realistic indication of their predictive performance.

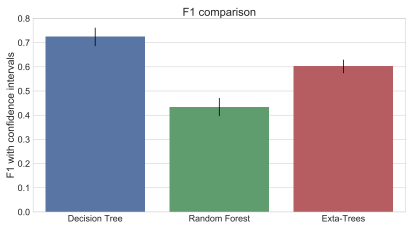
Using the scores of this modified repeated cross validation procedure, we compute bootstrap confidence intervals [8] for the mean of each metric. Here we use bootstrap confidence intervals as the distributions of the scores are not normal, hence we cannot use *normal* or t quantiles. The results of the modified repeated 5-fold cross validation along with the bootstrap confidence intervals can be seen in Figure 28.



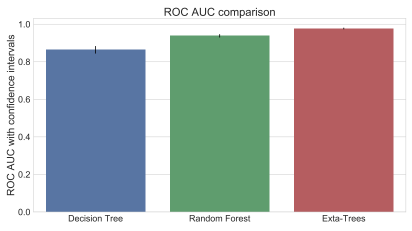
(a) Precision



(b) Recall



(c) F_1-score



(d) Area Under the ROC Curve

Figure 28: Modified repeated 5 - Fold cross validation results for 4 metrics

From this figure can be observed that the results change drastically when we modify the seed in the classifiers. The largest change in the scores is observed for the Random Forest which in this case achieves a precision 0.39 and recall 0.53, something that results in an F_1-score less than 0.5. This means that the classifier performs worse than random and hence should not be used for prediction without further parameter tuning. The Extra-Trees classifier achieves an F_1-score 0.601 due to the fact that with this procedure, its precision dropped to 0.5. However, it still achieves the highest score in terms of recall (0.85). On the other hand, the Decision Tree seems to be the most stable model under this setting, as its recall and precision dropped only slightly. This resulted to a slight drop of the F_1 score to 0.72, which is still relatively high. In Figure 28d can be seen that despite the drop in the other scores, the ROC AUC is still high. This is another indication that when the data are highly imbalanced, the ROC AUC score can mask poor performance. This is particularly observable for the Random Forest, which achieves ROC AUC 0.93, but the F_1-score is only 0.43. A Mann-Whitney U tests showed that the differences in the F_1-scores between the Decision Tree and the Extra-Trees is statistically significant (P $= 5.89 \cdot 10^{-18}$). This test also shows that the Extra-Trees achieve significantly larger recall than the Decision Tree ($P = 6.94 \cdot 10^{-6}$). These differences in the means can also be observed visually, since the confidence intervals for the Decision Trees and the Extra-Trees do not coincide.

Through this procedure, we observe that the most stable model in the case of predicting behavioral increase is the Decision Tree. It also achieves a higher F_1-score than both the Random Forest and the Extra-Trees. It is typical for Decision Trees to perform worse than ensemble tree methods, however there are cases that ensemble methods do not outperform single decision trees [24]. An intuitive explanation of why the Decision Tree performs better in this case, is due to the simplicity of the problem. Since there are only 5 positive instances, a simple model that captures the common patterns that

[8]Using the `bootstrap` routine from the *scikits.boostrap* package (v. 0.3.2).

are shared between these 5 positive cases, may perform better than more complex models (in our case Random Forest and Extra-Trees). Also, since the three models used different features, it might be the case that the Decision Tree captured the features with the strongest patterns. By comparing the Extra-Trees and the Random Forest, the extra randomization in the Extra-Trees seems to lead to better predictive performance. However, the number of trees in the Extra-Trees classifier is almost two times larger than the Random Forest, something that might plays a role in the robustness in small changes.

4.2 Predicting Dropout

In order to predict dropout, we built three different tree-based classifiers, namely Decision Tree, Extremely Randomized Trees and Extreme Gradient Boosting (XGBoost). The models are tuned in order to maximize the F_1*-score* and then the classifiers are compared in terms of predictive performance. Attention is paid in the imbalanced nature of the data, as the percentage of dropouts is 38% which means that the class `Dropped_out = 1` is the minority class.

4.2.1 Decision Tree

In order to find the best features for the decision tree, we first tune a decision tree classifier by applying random search [9], and we try N combinations for the specified ranges in each parameter of the decision tree. In our case we try $N = 2000$ combinations and we obtain the best parameters in terms of F_1-score, by applying 10-fold CV. The best parameters for this first decision tree (First DT) can be found in Table 11. Then, we fit our model to the whole dataset and we use the `SelectFromModel` function to find the features with the highest importance weights assigned from the fitted classifier. From a total of 90 features, 8 features are identified as important by the decision tree: `Condition, sports_club, plan_weekly_sports , max_cycle_time, move_less_tv, set_date_goal, compare_others` and `last3m_move_insteadOV`. By plotting the correlation matrix of these features in Figure 29, we see that the majority of the features have low correlations, except of the `sports_club` and `plan_weekly_sports` which have correlation 0.69. The next larger correlation is between `max_cycle_time` and `last3m_move_insteadOV` (0.44), while the rest of the features have correlations lower than 0.21.

[9]Using the `RandomizedSearchCV` routine from the *scikit-learn* python package (v. 0.18.1).

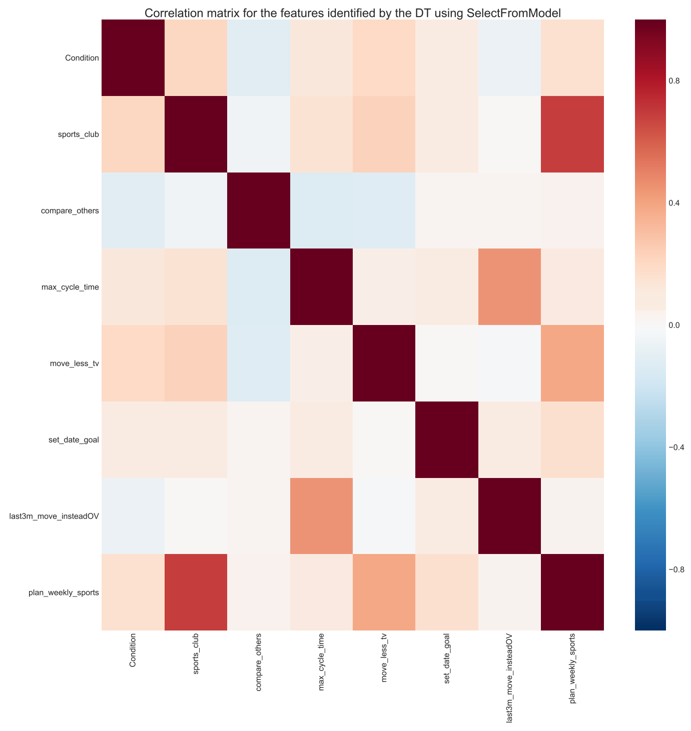


Figure 29: Correlation matrix of the features identified by the Decision Tree using the SelectFromModel function.

The next step consists of tuning and fitting a decision tree in a data set containing only the 8 most important features. We tune the parameters using random search with a large number of iterations (150000) and we try to find the parameters that optimize the F_1-score. Note here that in the First DT we only used 2000 iterations as we need this model only to identify the most important features and not to predict, hence 2000 iterations are adequate [10]. However, for the Best DT that will be used for prediction, we need to find the best possible parameters and this is the reason why we use a much larger number of iterations. Moreover, we should mention that in the case of predicting behavioral increase, we used 2000 iterations in all the models, as the problem was simpler (5 positive instances) and more iterations did bring any improvement in the results. The parameters of the decision tree that resulted to the maximum F_1 can be seen in Table 11. In order to deal with the class imbalance we again use the `class_weight = 'balanced'` parameter in the `DecisionTreeClassifier`. As with predicting behavioral increase, SMOTE was used again to handle the class imbalance but without better results. We also use `random_state=99` in all the classifiers in order be able to compare them in equal terms.

Table 11: Hyperparameter tuning of the Decision Tree for dropout

Hyperparameters	**First DT**	**Best DT**	**Balanced DT**
criterion	gini	gini	entropy
max depth	3	7	3
max leaf nodes	9	17	10
min samples leaf	4	1	1
min samples split	14	3	5
max features	None	2	2

[10]We also tried more than 2000 iterations, however the same features were identified as important.

Despite the fact that this model achieved the maximum F_1-score (0.755), by plotting the structure of the Decision Tree in Figure 30, it can be seen that the tree is quite large and also some of the bottom right leaf nodes have only 1 sample. This indicates that the model overfits in the specific dataset, which means that it does not generalize the data well.

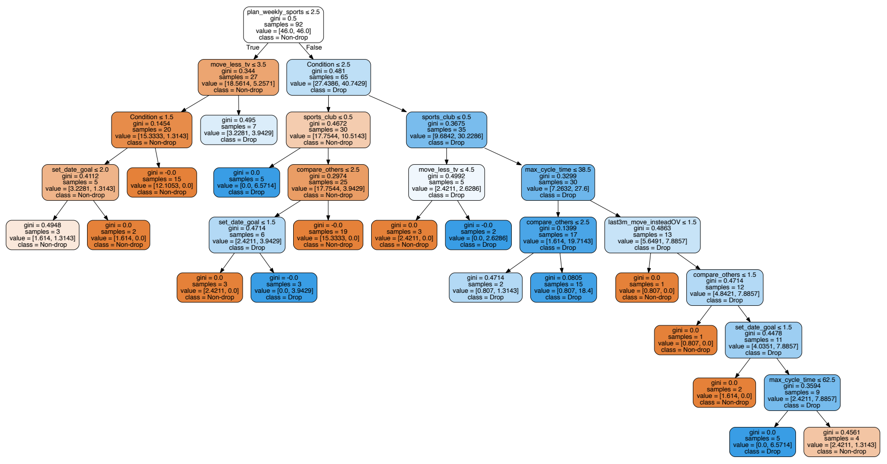


Figure 30: Visualization of the Decision Tree with the best parameters (Best DT)

Typically, in order to prevent that, we must prune the tree after we construct it. However, the *scikit-learn* package does not support post-pruning, so we can prevent the tree from overfitting, by changing the parameters that concern the maximum depth of the tree (max depth) and the minimum number of samples required to split an internal node (min samples split). To achieve that, we apply again random search and choose the best possible parameters that have lower `max_depth` and larger `min_samples_split` than the Best DT. The identified parameters for this more balanced Decision Tree (Balanced DT) can be seen in Table 11. Of course, since there is a bias-variance trade-off, this decision tree scores slightly lower F_1 (0.749), however the structure of the tree in this case is simpler, as can be seen in Figure 31. Since we want the tree to be more generalizable, rather than predict better in this specific dataset, we will use this Balanced DT for modeling.

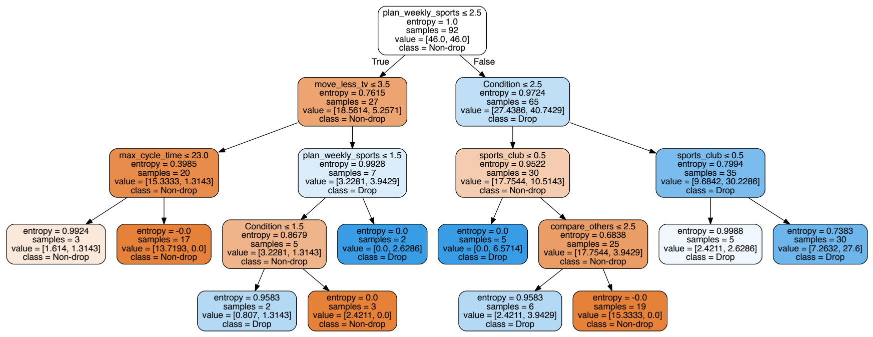


Figure 31: Visualization of the Decision Tree with the balanced parameters (Balanced DT)

4.2.2 Extremely Randomized Trees

The next algorithm we will use to predict dropout is the Extremely Randomized Trees (Extra-Trees) algorithm. We use this technique in order to obtain more generalizable predictions than from the decision tree as the latter tend to overfit [22]. We choose this algorithm over a Random Forest, as in predicting behavioral increase the Extra-Trees outperformed the Random Forest [11].

To identify the best features to fit our model, we first tried the `SelectFromModel` procedure as with the decision tree. This function identified 29 features as important and their correlation matrix can be seen in Figure 32.

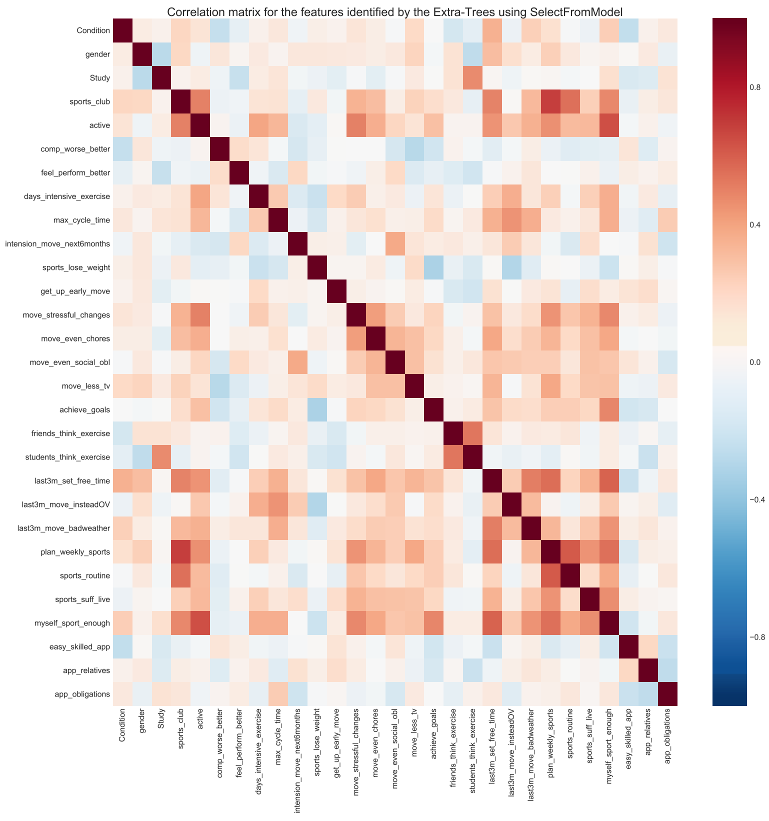


Figure 32: Correlation matrix of the features identified by the Extra-Trees using the SelectFromModel function

However, the performance results (in terms of F_1) of the Extra-Trees using these 29 features were relatively low even after parameter tuning, probably due to the large number of features used, and the high correlations that are observed in Figure 32. We also tried to use Recursive Feature Elimination with various values for identified features (5, 10, 15, 20), although the best predictive performance was observed by using the 8 best features identified by the Decision Tree (Figure 29). We use the subset of the 8 best features, to tune our classifier using random search. We apply 25000 iterations and we use 10-fold cross validation to find the parameters that maximize the F_1-score. Alternatively, we can apply a grid search to try all the possible parameter combinations, however this procedure is very computationally expensive [12]. The best parameters obtained by this procedure are depicted in Figure 12. The number of estimators indicates that the Extra-Trees classifiers consists of 44 trees.

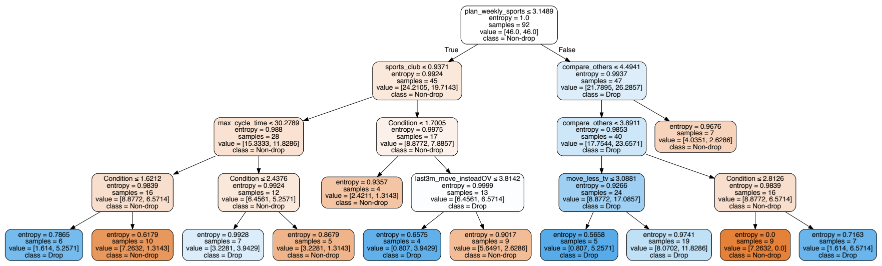
[11] Note that we also tried to predict dropout using a Random Forest, although with worse results than the Extra-Trees.

[12] The random search (which is considered faster than the grid search) needs almost 5 hours for 25000 iterations.

Table 12: *Hyperparameter tuning of the Extra-Trees for dropout*

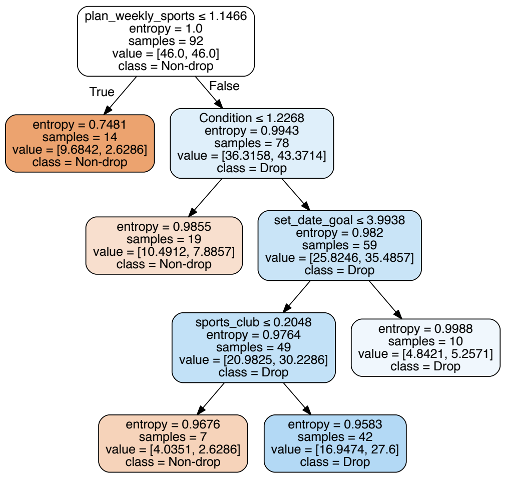
Hyperparameters	**Value**
criterion	entropy
number of estimators	44
max depth	3
max leaf nodes	13
min samples leaf	4
min samples split	5
max features	2

To get an indication of the created trees in the Extra-Trees classifier, we plot two (out of 44) of the trees in Figure 33. In Figure 33a is depicted a quite complex tree, while in Figure 33b a simpler one. Note that most of the trees in the classifier were quite complex, as the one in Figure 33a. Moreover, we observe that the majority of the nodes of the trees have large impurities, indicating *weak* individual trees. However, when all the trees are combined together, the ensemble becomes strong.



plan_weekly_sports ≤ 3.1489 entropy = 1.0 samples = 92 value = [46.0, 46.0] class = Non-drop
True
False
sports_club ≤ 0.9371 entropy = 0.9924 samples = 45 value = [24.2105, 19.7143] class = Non-drop
compare_others ≤ 4.4941 entropy = 0.9937 samples = 47 value = [21.7895, 26.2857] class = Drop
max_cycle_time ≤ 30.2789 entropy = 0.988 samples = 28 value = [15.3333, 11.8286] class = Non-drop
Condition ≤ 1.7005 entropy = 0.9975 samples = 17 value = [8.8772, 7.8857] class = Non-drop
compare_others ≤ 3.8911 entropy = 0.9853 samples = 40 value = [17.7544, 23.6571] class = Drop
entropy = 0.9676 samples = 7 value = [4.0351, 2.6286] class = Non-drop
Condition ≤ 1.6212 entropy = 0.9839 samples = 16 value = [8.8772, 6.5714] class = Non-drop
Condition ≤ 2.4376 entropy = 0.9924 samples = 12 value = [6.4561, 5.2571] class = Non-drop
entropy = 0.9357 samples = 4 value = [2.4211, 1.3143] class = Non-drop
last3m_move_insteadOV ≤ 3.8142 entropy = 0.9999 samples = 13 value = [6.4561, 6.5714] class = Drop
move_less_tv ≤ 3.0881 entropy = 0.9266 samples = 24 value = [8.8772, 17.0857] class = Drop
Condition ≤ 2.8126 entropy = 0.9839 samples = 16 value = [8.8772, 6.5714] class = Non-drop
entropy = 0.7865 samples = 6 value = [1.614, 5.2571] class = Drop
entropy = 0.6179 samples = 10 value = [7.2632, 1.3143] class = Non-drop
entropy = 0.9928 samples = 7 value = [3.2281, 3.9429] class = Drop
entropy = 0.8679 samples = 5 value = [3.2281, 1.3143] class = Non-drop
entropy = 0.6575 samples = 4 value = [0.807, 3.9429] class = Drop
entropy = 0.9017 samples = 9 value = [5.6491, 2.6286] class = Non-drop
entropy = 0.5658 samples = 5 value = [0.807, 5.2571] class = Drop
entropy = 0.9741 samples = 19 value = [8.0702, 11.8286] class = Drop
entropy = 0.0 samples = 9 value = [7.2632, 0.0] class = Non-drop
entropy = 0.7163 samples = 7 value = [1.614, 6.5714] class = Drop

(a)



plan_weekly_sports ≤ 1.1466 entropy = 1.0 samples = 92 value = [46.0, 46.0] class = Non-drop
True
False
entropy = 0.7481 samples = 14 value = [9.6842, 2.6286] class = Non-drop
Condition ≤ 1.2268 entropy = 0.9943 samples = 78 value = [36.3158, 43.3714] class = Drop
entropy = 0.9855 samples = 19 value = [10.4912, 7.8857] class = Non-drop
set_date_goal ≤ 3.9938 entropy = 0.982 samples = 59 value = [25.8246, 35.4857] class = Drop
sports_club ≤ 0.2048 entropy = 0.9764 samples = 49 value = [20.9825, 30.2286] class = Drop
entropy = 0.9988 samples = 10 value = [4.8421, 5.2571] class = Drop
entropy = 0.9676 samples = 7 value = [4.0351, 2.6286] class = Non-drop
entropy = 0.9583 samples = 42 value = [16.9474, 27.6] class = Drop

(b)

Figure 33: *Two trees of the Extra-Trees classifier with the best parameters*

4.2.3 Extreme Gradient Boosting

The last algorithm implemented for the prediction of dropout is the Extreme Gradient Boosting (XGBoost) [10]. This technique which can be used both for regression and classification problems, belongs to a family of boosting algorithms that convert weak learners into strong ones. At its core, it uses the gradient boosting technique, which creates a prediction model in the form of an ensemble of weak prediction models (typically decision trees). The models are built sequentially and each new model uses the Gradient Descent method to minimize a differentiable loss function. The key idea of gradient boosting is to create new learners which have the maximum correlation with the negative gradient of the loss function, which is associated with the whole ensemble [38].

The XGBoost algorithm was created by Tianqi Chen and as he states, it differs from gradient boosting in the sense that it is a more regularized model formalization to control overfitting, something that results in a higher predictive performance. Despite its better performance, the algorithm is fast and it allows for a lot of parameter tuning. Due to these benefits, the algorithm has become popular, especially in Kaggle competitions [15]. In order to find the best features to fit our model, we first tried to use the `SelectFromModel` routine in the whole dataset. This procedure identified 32 features as important, something that resulted in poor predictive performance, even after parameter tuning. Moreover, we tried Recursive Feature elimination with several values for the number of the selected features (5, 10, 15 and 20). However, the best results (in terms of F_1) were obtained by using the 8 best features identified as important by the Decision Tree. Note that since XGBoost can only handle numerical features, it is important to convert all the categorical variables that are used, to binary. In order to find the best parameters for the model, we create a subset of the whole dataset that uses only the 8 features from the Decision Tree and we tune our model by applying random search with 150000 iterations and maximizing F_1. Alternatively, we can apply a grid search, in order to try all the possible combinations of parameters, however this is extremely computationally expensive given the large number of the parameters allowing tuning. Note that the reason why we used 150000 iterations for the XGBoost instead of 25000 for the Extra-Trees, is due to the fact that the XGBoost is much faster algorithm than the Extra-Trees, something that allows for extra iterations in the random search as the models are trained faster. More specifically, the Extra-Trees needs almost 5 hours for 25000 iterations, while the XGBoost, achieves 150000 iterations in less than 3 hours [13]. The best identified parameters for the XGBoost can be seen in Table 13.

Table 13: Hyperparameter tuning of the XGBoost for dropout

Hyperparameters	**Value**
colsample by tree	0.4
learning rate	0.26
max depth	9
min child weight	1
number of estimators	114
subsample	0.6
scale pos weight	1.62
gamma	0.4

Note that the XGBoost implementation in Python [14] does not support the `class_weight = 'balanced'` parameter. However, `scale_pos_weight` parameter can control the balance of positive and negative weights in the case that the data is unbalanced. For this parameter we use the typical value:

$$scale\_pos\_weight = \frac{sum(negative\_cases)}{sum(positive\_cases)}$$

which results in the value 1.62 that can be seen in Table 13. Moreover, the number of boosted trees that are used in the model is determined by the number of estimators, which in this case is 114.

[13]The exact running time depends on the computational power.

[14]Created by the `XGBClassifier` from the *scikit-learn* python package (v. 0.18.1).

4.2.4 Feature Importance & Tree explanation

In order to understand the reasoning of prediction in the Decision Tree with the balanced parameters, we observe its structure in Figure 31. One possible path of the tree is the following. Starting from the top node, we see that the tree firstly checks the `plan_weekly_sports` feature. If the answer in this question is *sometimes, often* or *very often*, then the tree checks the condition of the user. If the user is in one of the A2G conditions, then the tree checks if the user is a member of a sports club. If the user is indeed a member of a sports club (`sports_club = 1`), then the `compare_others` feature is being checked. If the user replied that he or she compares to others *sometimes, many times* or *always*, them the user is identified as non-dropped. We see that this leaf node is pure, as all the 19 users who satisfied the aforementioned criteria are all non-dropped users.

To further investigate the important features for the Decision Tree, we plot the feature importances in Figure 34. It can be seen that only 6 out of 8 fitted features have non-zero importances. The sports club membership is identified as the most important feature.

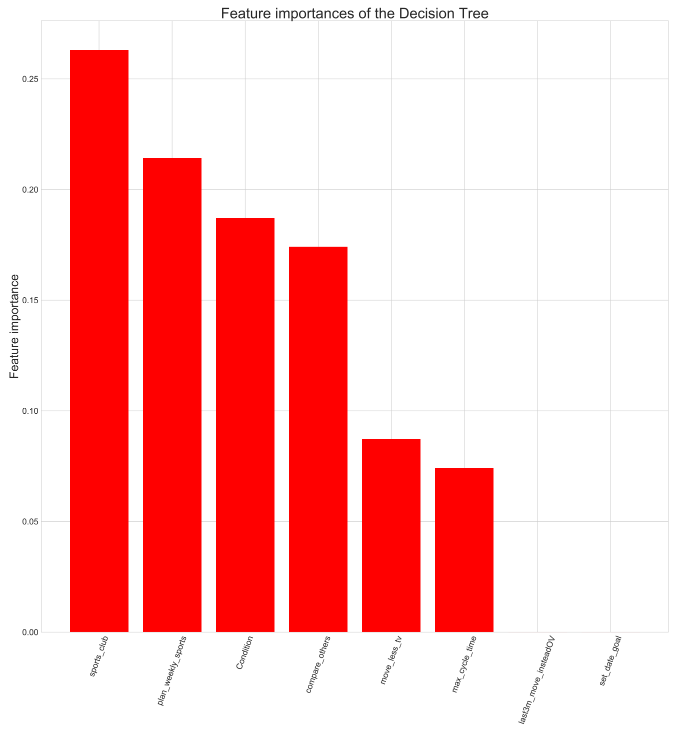


Figure 34: Feature importances of the decision tree with the best parameters (Best DT)

Figure 35 depicts the dropout percentages for members and non-members of a sports club. It can be seen that the difference in the percentages is slightly larger for the club members. However, the reason why this feature is important, is due to the difference in the percentages of dropout for the club membership of each condition separately. As can be seen in Figure 36a users that belong to the A2G Full group and are not members of a sports club, have larger percentage than the non-dropped users. Similarly, users from the Fitbit group who are club members, have larger percentage than the non-dropped club members (Figure 36c). On the other hand, users from the A2G Light who are not club members, have 0 dropout percentage (Figure 36b).

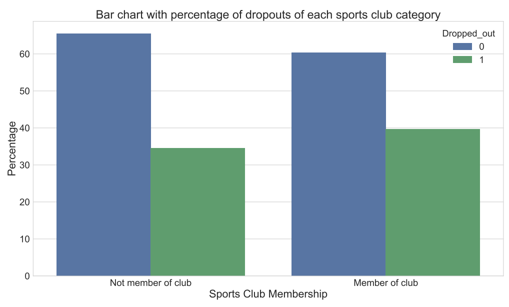
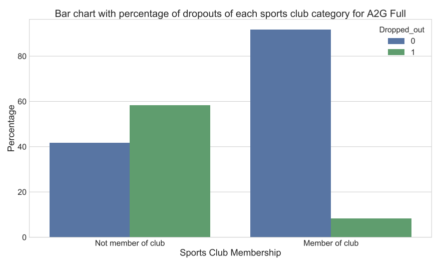
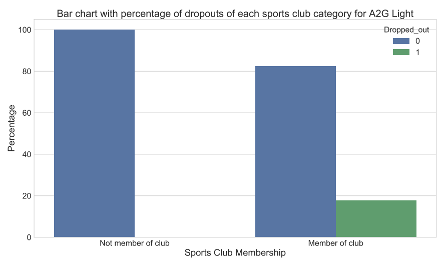


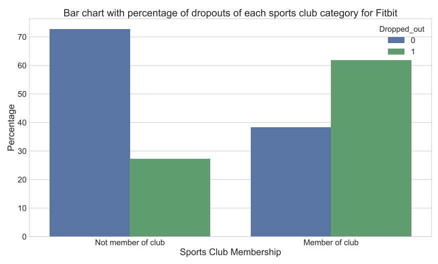
Figure 35: Bar chart showing the relation between dropout and the sports_club feature



(a) A2G Full



(b) A2G Light



(c) Fitbit

Figure 36: Bar charts showing the relation between dropout and the sports_club feature for each condition.

By applying a Fisher exact test for the differences in the percentages of dropout between members and non-members of a sports club for each condition, it is found that only the difference in the A2G Full group is significant, as can be seen in Table 14. Moreover, for the Fitbit group, since the p-value is slightly larger than 0.05, it means that there is a trend towards significant difference in the percentages. This finding suggests that the variable `sports_club` affects significantly the dropout rate of the A2G Full users and in a smaller extent those in the Fitbit group.

Table 14: P-values of the Fisher exact test for the significance in the differences of the dropout percentages between members and non-members of a sports club for each condition.

Condition	**Fisher P-value**
A2G Full	0.02
A2G Light	0.53
Fitbit	0.08

The second most important feature, is the one which concerns the weekly planning of sports. In Figure 37 we observe that the users who answered that they plan sports on weekly basis *sometimes* or *often*, have larger dropout percentages than the other categories. Fisher exact tests showed that these differences in the percentages are statistically significant, as is depicted in Table 15.

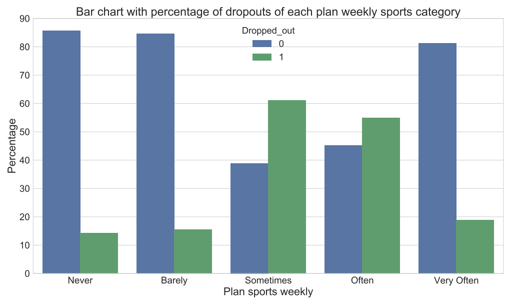


Figure 37: Bar chart showing the relation between dropout and the plan_weekly_sports feature

Table 15: P-values of the Fisher exact test for the significance in the differences of the dropout percentages between the answers in the plan_weekly_sports feature.

Answers	**Fisher P-value**
Sometimes & Never	0.01
Sometimes & Barely	0.02
Sometimes & Very Often	0.01
Often & Never	0.02
Often & Barely	0.02
Often & Very Often	0.02

The third most important feature is the condition of the user. This is due to the highest dropout percentage for the Fitbit group which was observed in Figure 4. The next most important feature is the one which concerns how often a user compares himself or herself with others. From Figure 38 is observed that users who *never* compare themselves to others have the largest dropout percentage (50%), while users who *always* compare themselves to others have the lowest percentage of dropout (27.27%). However, Fisher exact tests applied between the percentages of the answers in this feature, showed that the differences are not statistically significant, as can be seen in Table 16. The pattern in Figure 38 indicates that social comparison might have a negative effect on the dropout percentage of this specific dataset, however the pattern is not strong enough due to the statistical insignificance of the differences in the percentages. The rest of the features have importance lower than 0.1.

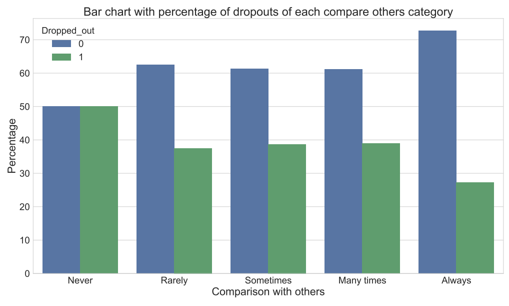


Figure 38: Bar chart showing the relation between dropout and the compare_others feature

Table 16: P-values of the Fisher exact test for the significance in the differences of the dropout percentages between the answers in the compare_others feature.

Answers	**Fisher P-value**
Never & Rarely	1.0
Never & Sometimes	0.66
Never & Many Times	0.67
Never & Always	0.60
Always & Rarely	1.0
Always & Sometimes	0.71
Always & Many Times	0.72

In order to find which features are the most important for the Extra-Trees, we plot the feature importances of the classifier in Figure 39. We observe that while the Extra-Trees uses the same features as the Decision Tree, the ranking of the feature importances is different (compared to Figure 34). More specifically, the Extra-Trees identify `Condition` and `plan_weekly_sports` as the most important features, while the rest have feature importance less than 0.15. Moreover, the variable `last3m_move_insteadOV` had very low feature importance (lower than 0.05) for the decision tree, however in Extra-Trees this feature has feature importance 0.1. This feature indicates if the user used the bike instead of car or means of transport in order to move during the 3 months before the experiment. To understand why it is an important feature, we plot this variable for both dropout categories, in Figure 40. In this figure we observe that the users who answered that they *barely* used bike instead of car, have the largest percentage of dropout, compared to the other answers. This is probably due to the fact that users who are not willing to use bike instead of car, have less motivation to be physically active and hence are more prone to dropout. However, Fisher exact tests applied to the percentages of the possible answers, showed that the only statistically significant difference is between the *barely* and *very often* answers, as can be seen in Table 17.

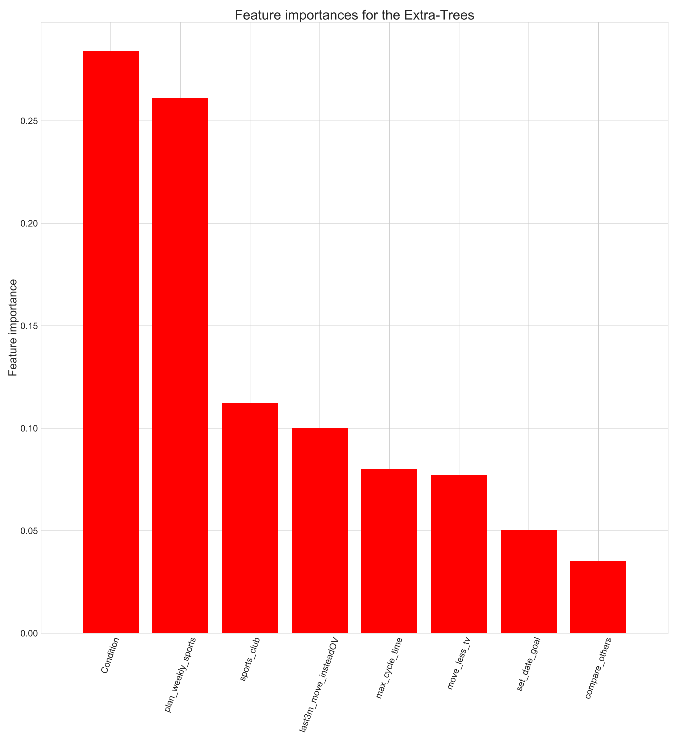


Figure 39: Feature importances of the Extra-Trees with the best parameters

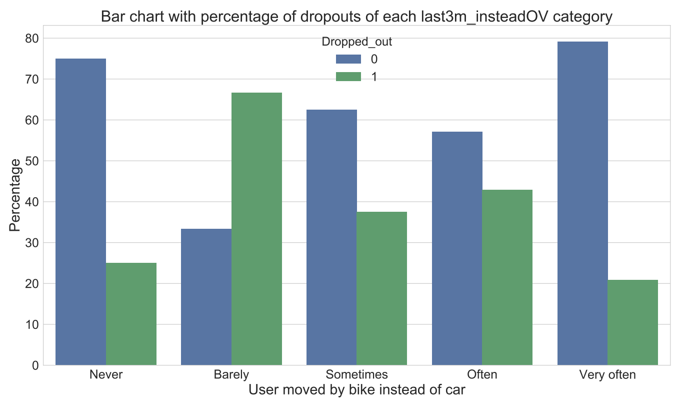


Figure 40: Bar chart showing the relation between dropout and the last3m_move_insteadOV

Table 17: P-values of the Fisher exact test for the significance in the differences of the dropout percentages between the answers in the last3m_insteadOV feature.

Answers	**Fisher P-value**
Barely & Never	0.26
Barely & Sometimes	0.15
Barely & Often	0.30
Barely & Very Often	0.01

In order to find which features are identified as most important by the XGBoost, we plot the feature importances in Figure 41. Note that except the `max_cycle_time` which is the only numerical feature, all the other variables in this figure are the dummified versions of the categorical features used for fitting. We observe that the XGBoost identifies as most important the `max_cycle_time` feature, which indicates the maximum cycle time that a user is willing to travel before taking means of transport.

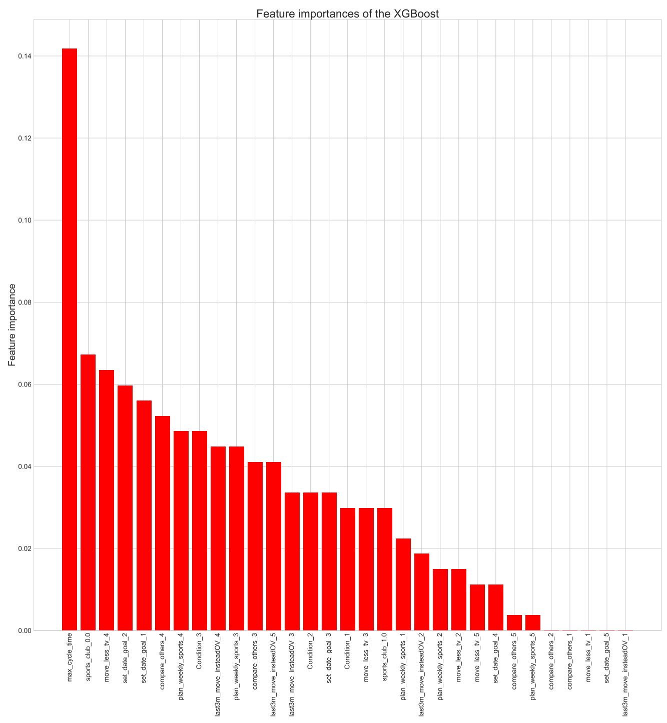


Figure 41: Feature importances of the XGBoost with the best parameters

To understand why this variable is important , we plot in Figure 42 the densities of the maximum cycling time for both dropped and non-dropped users. We observe that both densities are right skewed with a long right tail. However, the density for the dropped users indicates lower values than the density for the non-dropped users. This makes sense, as users who are willing to cycle more before taking the OV, are less likely to dropout, probably due to higher motivation.

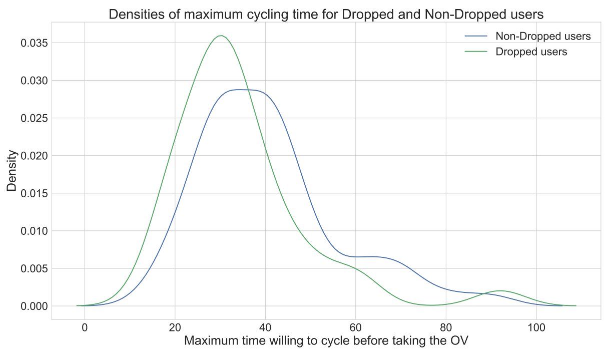


Figure 42: Densities of the maximum time willing to cycle before taking the OV, for dropped and non-dropped users.

The second most important feature is the `sports_club_0.0` which indicates the users that are not member of a sports club. This probably due to the large fluctuation of the dropout percentages for non members, as it was shown in Figure 36. The next important feature is the `move_less_tv_4` which indicates if the users replied *probably I can do it* in the question about watching less TV in order to move more. This is due to the large percentage of dropout for this specific answer, as it is shown in Figure 43. The rest of the features have importances lower than 0.06. In Figure 41 we can also observe that the feature `Condition_3` which indicates the users in the Fitbit group, is more important than the other two conditions, due to the larger percentage of dropouts for the Fitbit group (Figure 4).

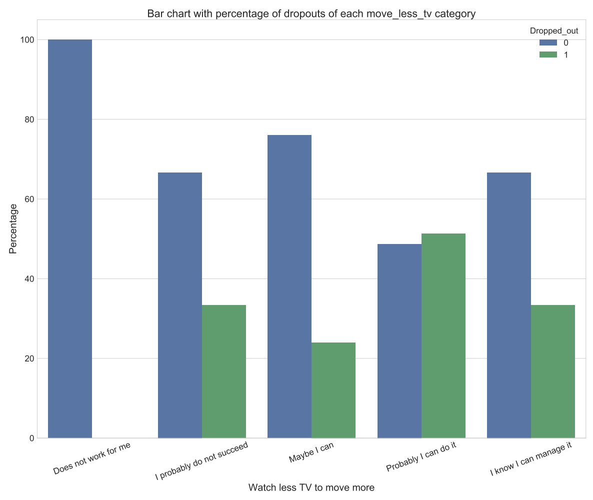
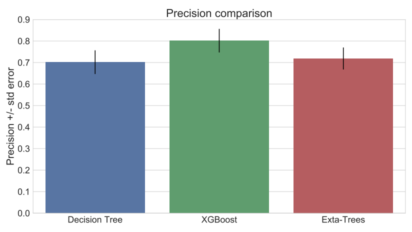


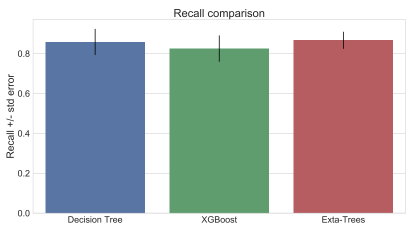
Figure 43: Bar chart showing the relation between dropout and the drop_move_less_tv variable.

4.2.5 Results & Evaluation

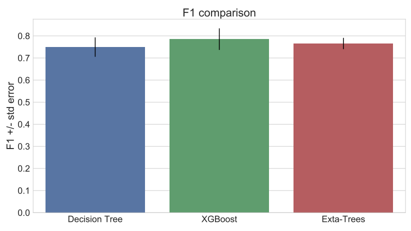
In order to evaluate the classifiers and compare their performance, we apply *K-Fold cross validation*. Opposed to the behavioral increase, in the case of dropout we have 35 positive instances and hence we can apply 10-fold cross validation. Note that since we fitted our algorithms to the same features, the comparison of the classifiers is more straightforward than in the case of activity increase. The cross validation results along with the standard error for 4 evaluation metrics, can be seen in Figure 44. In Figure 44a can be seen that the XGBoost achieves the highest average precision (0.801), while the Decision Tree and the Extra-Trees score lower (0.701 and 0.718 respectively). On the other hand, the XGBoost scores 0.825 as average recall, while the Decision Tree scores 0.858 and the Extra-Trees 0.866 (Figure 44b). When it comes to F_1-score in Figure 44c, the largest average value is achieved by the XGBoost (0.785), the second largest by the Extra-Trees (0.765) and the lowest by the Decision Tree (0.749). Finally, the Decision tree scores the largest ROC AUC (0.865) as can be seen in Figure 44d. Since the differences in the scores are small, Mann-Whitney U tests among all the possible combinations of classifiers and scores, showed that there are not statistically significant differences.



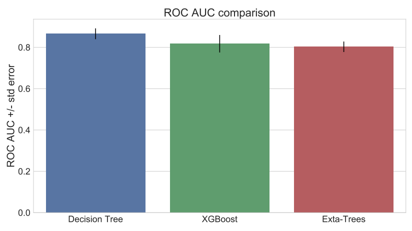
(a) *Precision*



(b) *Recall*



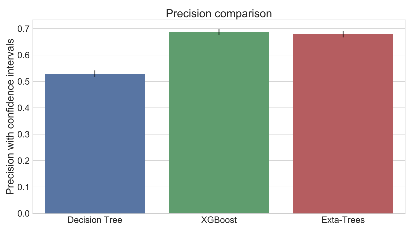
(c) *F_1-score*



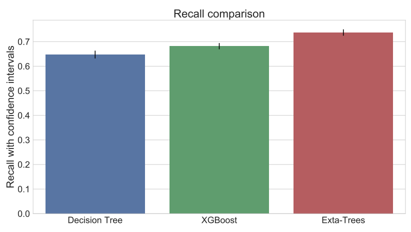
(d) *Area Under the ROC Curve*

Figure 44: 10 - Fold cross validation results for 4 metrics

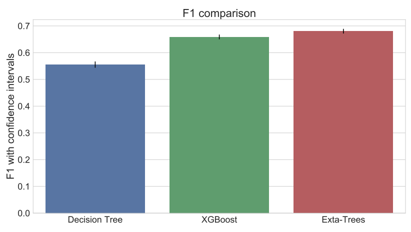
As with the case of predicting behavioral increase, we also want to check the stability of the classifiers by varying the random seed. With this procedure we check how the results of the classifiers vary if we apply slight changes in the training of the models. We apply repeated 10-fold cross validation by using 100 different values for `random_state` in the classifiers. This yields 1000 scores for each model and we use these scores to obtain bootstrap confidence intervals. The results of the repeated cross validation along with the confidence intervals can be seen in Figure 45.



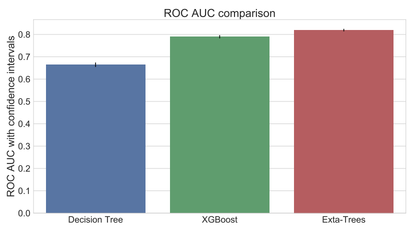
(a) *Precision*



(b) *Recall*



(c) *F_1-score*



(d) *Area Under the ROC Curve*

Figure 45: Modified repeated 10 - Fold cross validation results for 4 metrics

From this figure can be seen that the scores for all the models are lower than in the case of the 10-fold cross validation. This indicates instability for the models, probably due to the small sample size, as the models did not have enough training data. The largest reduction in the scores is observed for the Decision Tree for which the F_1-score has dropped from 0.74 to 0.55, indicating a prediction slightly better than random. The XGBoost, still achieves the highest precision (0.68) compared to the Extra-Trees (0.67) and the Decision Tree (0.52). Although, a Mann-Whitney U test between the precision of the XGBoost and the Extra-Trees produces a p-value 0.06, indicating that the difference is not statistically significant. As in the case of predicting the behavioral increase, the Extra-Trees classifier scores the highest recall (0.73), which is significantly larger than the recall of Decision Trees and the XGBoost, as the confidence intervals do not coincide. When it comes to the F_1, the Extra-Trees achieves the largest score (0.68), while the XGBoost scores 0.658. A Mann-Whitney U test between the F_1-scores for these models produced a p-value 0.0005, meaning that the difference is statistically significant. Concerning the ROC AUC score, we see that in the case of dropout where the data are not as skewed as in behavioral increase, its values are closer to those of the F_1-score. This means that when the data are not so skewed, the ROC AUC metric provides a better indication of the model's performance than in the case of highly skewed data.

Through the procedure of the repeated cross validation we see that the Decision Tree is the most unstable among the models. Generally, the decision trees are known to be unstable algorithms, meaning that they produce drastically different results from training sets that differ just slightly [13]. We saw that the Decision Tree was quite stable in the case of behavioral increase where the positive instances were only 5, however in the case of dropout the problem is more complicated and hence more complex algorithms than the XGBoost or the Extra-Trees perform better. The Extra-Trees classifier achieved the largest F_1-score and thus, this is the model that should be utilized in a practical application.

5 Discussion

In this study, tree-based machine learning algorithms were used in order to predict behavioral increase and dropout in a physical activity experiment. Through the feature selection process, the most important features that affect both attributes were identified. Further investigation on the identified features, showed several patterns that explain how each variable affects behavioral increase or dropout.

For the behavioral increase, it was found that all the users with significantly increased physical activity levels, totally agree with fact that physical activity helps to look better. This is an indication that physical appearance is a vital motivation for increasing the activity levels. This feature might be generalizable in other datasets, however only in the physical activity domain. This means that the same pattern might be observed again in other datasets concerning physical activity, as this features is directly connected to sports.

Moreover, all the users with increased behavior stated that they have the necessary resources to use a moving app. This feature may also be generalizable in other datasets, but only inside the domain of physical activity intervention apps, as this feature directly refers to moving apps.

Another finding is that most users with increased behavior rarely have a challenging feeling when people perform better than they do. This indicates an insensitivity to social comparison for the users with increased behavior, and this pattern may generalize also in other domains. This means that the pattern of insensitivity to social comparison can be also observed in datasets outside the physical activity domain, as this feature is not directly connected to physical activity.

On the other hand, patterns such as the one indicating that the users with increased behavior compare themselves with healthier people, may only hold for this specific dataset. This pattern is in line with the statement in [32] that *upward* comparison (comparison with people that perform better) can be beneficial to the users who use the target as a motivation to self-improve.

The models used for predicting behavioral increase were tuned through 5-fold cross validation, providing relatively high results for one specific seed, however the results are lower when we take an average of 100 seeds. This modified version of repeated cross validation with different seeds is important because the models might perform well in this dataset with one specific seed, however this same seed might give much worse results in a completely new dataset. We saw that the evaluation results changed after the repeated cross validation, especially for the Random Forest. To elaborate, if we would deploy the Random Forest in completely new data, and we had only evaluated with K-Fold cross validation, we would expect an F_1-score around 0.8 based on the results of Figure 27. However, it could be the case for the classifier seed to be suboptimal for this new dataset, meaning that the real F_1-score could be under 0.5 as it is shown in Figure 28. Thus, it is important to evaluate the robustness of the models in order have a better indication of their real performance, as we did in this study with the modified repeated cross validation. The most important reason for the large variation in the prediction results after the modified repeated cross validation, is the small size of the dataset as there are not enough training data to create robust models. For the behavioral increase prediction, the classifier that worked best was the simplest and most interpretable one, namely the Decision Tree. Hence, in a practical application this is the model that should be deployed in order to predict behavioral increase in brand new data. An example of such a practical application is to identify the users who are most likely to show an increase in their physical activity levels, before the start of a new experiment, in order to use these users as a positive influence to less motivated users (probably through social media platforms). As in [44] where the Decision Tree achieved relatively high accuracy scores (over 70%), we also see in our study that the Decision Tree can perform well in the physical activity domain. On the other hand, the Random Forest which achieved high performance in [14], could not perform adequately in the case of behavioral increase, probably due to lack of data and positive instances. Moreover, from the evaluation of the models using both F_1 and ROC AUC metrics, we verified the statement in [25] that ROC AUC may mask poor performance when the data are highly skewed, as in the case of predicting behavioral increase.

Concerning the dropout of the experiment, this study extends the analysis started in [36]. Important patterns were observed among the three conditions for users that are members or non-members of a sports club. This pattern could be generalizable in other datasets in the physical activity domain. One reason for this is because people that are not members of a club might be less interested in physical activity and hence more prone to dropout. However, being part of a sports club could indicate an activity pattern that people are content with, causing them to forego the additional activity tracking from the intervention app.

Another pattern that could be generalized in other datasets in the physical activity domain is the one indicating that users who *barely* used their bike instead of car in the last 3 months before the experiment are more prone to dropout than the rest of the users. Also, users that used their bike instead of car *very often* have the lowest dropout percentage. An intuitive explanation behind this pattern is that using the bike instead of car on a frequent basis, indicates a larger motivation to be physically active, something that negatively affects the dropout rate.

On the other hand, the pattern indicating that users who *sometimes* or *often* plan their sports on a weekly basis, are more prone to dropout, might hold only for the specific dataset. This means that this pattern may have occurred in this dataset by chance and might not be observed in new data.

Moreover, the (weak) pattern showing that social comparison might have a negative effect on the dropout percentage could be generalizable in other datasets and probably in other domains. This is the case due to the fact that people who constantly compare themselves with others, might be more motivated and hence less prone to dropout, than people who are not competitive at all.

As with predicting behavioral increase case, we also applied for dropout a modified repeated cross validation procedure for the created models, by cross-validating using 100 different seeds. Again, there was a drop in the scoring results for all the models which is due to the small sample size. However, in the case of dropout, more complicated models outperformed the Decision Tree. More specifically, the Extra-Trees produced significantly larger F_1 than the XGBoost and hence this is the model that should be used for predicting dropout in new data. By comparing our findings to those in [41], we see that the Extra-Trees classifier can indeed perform quite well in the physical activity domain, even when the data are limited. By comparing the Extra-Trees with the XGBoost, we see that the latter achieves slightly worse results but is more computationally efficient, since it is faster algorithm than the Extra-Trees.

We need to mention that due to the fact that we applied the feature selection on the whole dataset, the cross validation results might be slightly optimistic. This is because the selected features hold some information which is included in the test folds of the cross-validation, introducing a bias in the results. This debate regarding the feature subset selection bias, has been discussed in articles such as [20], [43] and [37]. In order to produce completely unbiased cross-validation results, the feature selection should ideally be applied within the cross-validation loop. This means that for each CV fold the models should select the most important features based on the training set of this specific fold. This would result in K sets of important features, where K is the number of the CV folds. Although, it needs to be clear that this procedure only guarantees that the CV results are completely unbiased and does not imply an optimal feature selection. In order to find the best features based on the dataset at hand, the feature selection should be done in the whole dataset, as we did in this study.

6 Suggestions for Future Work

This study about predicting behavioral increase and dropout in the physical activity domain, has uncovered certain trends and possible relationships between the features and the attributes, that open up new avenues for future research. The most essential aspect that could improve the results significantly is acquiring new data. This could be achieved through another randomized controlled trial, probably with more participants (ideally more than 200 for a margin of error less than 7.1% [3]) and for a longer experimental period (6 months) in order for the results to be more valid. By getting new data it could be checked if the patterns identified in this study, also exist in the new dataset. Moreover, with these data it is possible to get a better indication for the determinants of the behavioral increase, as more positive instances could be created and the time interval of the first and final measurement could be larger than 2 months.

Before the start of a new randomized control trial, the Extra-Trees classifier could also be used to predict the users that are most prone to dropout. Depending on the size of the participant pool, the users with the highest probability of dropout can be excluded from the experiment if there is a maximum number of participants that must not be exceeded. However, this procedure has a pitfall, meaning that it can create a bias in the participant selection as we may only include users with specific characteristics. One possible way to deal with this trade-off, is to only ignore users that exceed a certain threshold for the probability of dropout. For example, we can only exclude users that have a probability to dropout over 90%. A safer way to deal with this pitfall, is to use the new experiment only as an evaluation of the created models, by checking if the models predict the dropouts accurately. Then the models can be re-trained and re-tuned on both the old and new data and they can be finally deployed in the next experiment.

By getting new data, the models could be further improved and evaluated, as there would be more training data. A richer dataset can provide a more reliable evaluation of the models. Except the K-Fold and the repeated cross validation, the data can also be split in training, validation and test sets. The validation set can be used for the hyper-parameter tuning and then the performance of the classifiers can be tested on the test set, which consists of unseen data. Furthermore, the high dimensionality of the data turns out to complicate the analysis in a great extent and hence, a technique to reduce the dimensions could be proven useful. Dimensionality reduction can be achieved by applying Principal Component Analysis (PCA). This technique is an orthogonal linear transformation which is applied to transform the data to a new coordinate system. In the first coordinate (the principal component) of this new system, lies a projection of the data with the greatest variance, and each succeeding coordinate has the largest possible variance under the constraint that it is orthogonal to the preceding coordinate [26]. Using only the principal components to explain most of the variation, results to dimensionality reduction, as many of the dimensions could be redundant due to high correlation. This reduction in the dimensions of a high dimensional dataset, such as the one used in this study, can simplify the analysis and improve the models' performance.

The created models could also become more robust by applying the feature selection (or possibly the PCA) and the hyper-parameter tuning using K-fold cross validation in an inner loop. Then, each hyper-parameter setting is again cross-validated in an outer loop for a large number of possible seeds. The setting that achieved the best results after this procedure can be used to create the classifiers.

When new data is collected, there are several hypotheses that could be tested in order to check if the patterns observed in this dataset can also be generalized in other datasets. These hypotheses are the following:

1. The A2G Full group has the largest percentage of behavioral increase compared to the other two groups.
2. Users that totally agree with the fact that physical activity helps them look better, have a larger probability to show significant increase in their behavior than users that agree less with the statement.
3. Having the necessary resources to use a moving app has a positive effect on increasing physical activity levels.
4. Users that compare themselves with healthier people, have a larger probability to increase their activity levels than users that compare themselves with less healthy people.
5. Users who show an increase in their physical activity levels are not sensitive to social comparison.
6. Users in the Fitbit group are more prone to dropout than users in the A2G groups due to no interaction with an intervention app.
7. Being a member of a sports club has a negative effect on the dropout rate on the A2G Full users and a positive effect on the dropout rate of the Fitbit users.
8. Users that *often* or *sometimes* plan their sports on a weekly basis are more prone to dropout than the rest of the participants.
9. Social comparison (comparison with others) has a negative effect on the dropout rate.
10. Users who *barely* used their bike instead of car in the last 3 months before the experiment, are more prone to dropout than the rest of the participants. Also, users who used their bike instead of car *very often* have the smallest probability to dropout.

If hypothesis 1 is confirmed on the new dataset, then we will have a stronger indication that the tailored coaching messages indeed help people to increase their activity levels. Also, in the case that hypothesis 6 is confirmed, then it will be an indication that using an intervention app motivates people to continue being physically active and not drop out from the experiment. The rest of the hypotheses from the above list that will be confirmed after the new randomized control trial, will enhance the claim that the corresponding patterns do not only hold for the specific dataset but are also generalizable in other datasets.

A Appendix

In this appendix we provide a brief explanation of all the 90 features extracted from the intake questionnaire. These features are the following:

1. **Condition**: 1 = A2G Full, 2 = A2G Light, 3 = Fitbit
2. **gender**: 0 = Man, 1 = Woman
3. **age**: the age of the user (integer)
4. **BMI**: the Body Mass Index of the user (float)
5. **Study**: 1 = full time education, 2 = part time education, 3 = no study
6. **Job**: 1 = no job, 2 = job, 3 = multiple jobs
7. **sports**: 1 = no sports, 2 = one sport, 3 = several sports
8. **sports_club**: 1 if user does sports at a club/association, 0 otherwise
9. **sports_home**: 1 if user does sports at home or outside, 0 otherwise
10. **home_away**: 1 = user lives with parents, 2 = user lives alone
11. **active**: How active user thinks he is. 1 = Very insufficient, 2 = insufficient, 3 = Adequate, 4 = Very adequate
12. **move_motivation**: How much motivated a user is, in a scale from 0 to 10.
13. **activity_satisfaction**: How much satisfied a user is with his activity levels, in a scale from 0 to 10.
14. **compare_others**: How often a user compares himself to other when he estimates how much he moves. 1 = Never, 2 = Rarely, 3 = Sometimes, 4 = Many times, 5 = Always.
15. **comp_worse_better**: User compares himself with individuals that perform worse or better. 1 = better, 2 = worse
16. **feel_perform_better**: How often a user feels good if he performs better than people at work/study. 1 = Never, 2 = Rarely, 3 = Sometimes, 4 = Many times, 5 = Always.
17. **challenge_feeling**: How often a user has a challenge feeling when people at work/study perform better than him/her. 1 = Never, 2 = Rarely, 3 = Sometimes, 4 = Many times, 5 = Always.
18. **days_intensive_exercise**: Days in the week before the intake, that a user had intensive exercise for at least 30 minutes per day. 1 = no day, 2 = 1 day, . . . , 8 = 7 days.
19. **times_intensive_onepiece**: Times in the week before the intake, that a user had very intense exercise for at least 20 minutes in one piece. 1 = 0 times, 2 = 1 time, . . . , 8 = 7 times.
20. **max_walk_time**: Maximum time (in minutes) for which the user is prepared to walk instead of taking the car or the OV.
21. **max_cycle_time**: Maximum time (in minutes) for which the user is prepared to cycle instead of taking the car or the OV.
22. **max_stairs**: Maximum number of floors (integer) for which the user is prepared to use the stairs instead of the elevator.
23. **intension_move_next6months**: Intension of the user to move more or do more sports in the next six months. 1 = Certainly not, 2 = Probably not, 3 = Maybe not / maybe, 4 = Probably, 5 = Certainly
24. **intension_move_nextmonth**: Intension of the user to move more or do more sports in the next month. 1 = Certainly not, 2 = Probably not, 3 = Maybe not / maybe, 4 = Probably, 5 = Certainly
25. **intension_move_nextweek**: Intension of the user to move more or do more sports in the next week. 1 = Certainly not, 2 = Probably not, 3 = Maybe not / maybe, 4 = Probably, 5 = Certainly
26. **sports_look_better**: To what extent the user agrees or disagrees with the fact that having enough sports/moving, then he/she looks better. 1 = Do not agree at all, 2 = Do not agree with, 3 = Agree, 4 = Totally agree
27. **sports_pos_impact**: To what extent the user agrees or disagrees with the fact that having enough sports/moving, has a positive impact on his/her health. 1 = Do not agree at all, 2 = Do not agree with, 3 = Agree, 4 = Totally agree
28. **sports_lose_weight**: To what extent the user agrees or disagrees with the fact that having enough sports/moving, then he/she loses weight. 1 = Do not agree at all, 2 = Do not agree with, 3 = Agree, 4 = Totally agree
29. **sports_fit_feeling**: To what extent the user agrees or disagrees with the fact that having enough sports/moving, then he/she gets a fit feeling. 1 = Do not agree at all, 2 = Do not agree with, 3 = Agree, 4 = Totally agree
30. **sports_feel_relax**: To what extent the user agrees or disagrees with the fact that having enough sports/moving, then he/she gets feels relaxed. 1 = Do not agree at all, 2 = Do not agree with, 3 = Agree, 4 = Totally agree

31. **sports_less_stressed**: To what extent the user agrees or disagrees with the fact that having enough sports/moving, then he/she is less stressed. 1 = Do not agree at all, 2 = Do not agree with, 3 = Agree, 4 = Totally agree
32. **get_up_early_move**: Which statement best describes if a user can get up early to move/workout even on weekends, for at least half a year. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
33. **move_after_longday**: Which statement best describes if a user can move or exercise after a long labor day, for at least half a year. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
34. **move_even_terrified**: Which statement best describes if a user can go moving or exercising even if he/she feels terrified, for at least half a year. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
35. **free_time_toexercise**: Which statement best describes if a user can free up time to move or exercise, for at least half a year. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
36. **move_with_others**: Which statement best describes if a user can keep moving with others for at least half a year, even if they seem too fast or slow to him/her. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
37. **move_stressful_changes**: Which statement best describes if a user can keep moving or exercising for at least half a year, even during stressful changes in his/her life. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
38. **active_first_thenfun**: Which statement best describes if a user can be active first and then do something fun, for at least half a year. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
39. **move_even_family_moretime**: Which statement best describes if a user can keep moving or exercising, for at least half a year, even if family, friends and/or partner claim more time. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
40. **move_even_chores**: Which statement best describes if a user can keep moving or exercising, for at least half a year, even if he/she has household chores. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
41. **move_even_busy_work**: Which statement best describes if a user can keep moving or exercising, for at least half a year, even if it is very busy at work. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
42. **move_even_social_obl**: Which statement best describes if a user can keep moving or exercising, for at least half a year, even if his/her social obligations require a lot of time. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
43. **move_less_tv**: Which statement best describes if a user can read/study less or watch less TV for at least half a year, in order to move more. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
44. **move_even_badweather**: Which statement best describes if a user can keep moving or exercising, for at least half a year, even if the weather is bad. 1 = I know that does not work for me, 2 = I probably do not succeed, 3 = Maybe I can, 4 = Probably I can do that, 5 = I know I can manage it.
45. **often_set_movegoal**: The user often sets a moving/sports goal. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
46. **more_than_1goal**: The user normally has more than one big move/sports goal. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
47. **set_date_goal**: The user often sets a date for when he/she has to achieve his/her move/sport goal. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
48. **goals_motivate**: The user's goals motivate him/her to do sports or exercise. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
49. **divide_goals**: The user tends to divide difficult goals into smaller ones. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
50. **goals_keep_prog**: The user normally keeps a progress towards his/her goals. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).

51. **goals_roadmap**: The user has developed a roadmap to achieve his/her goals. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
52. **achieve_goals**: The user normally achieves the goals that he/she sets to himself/herself. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
53. **analyze_goals**: If the user does not achieve a goal, he/she analyzes what went wrong. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
54. **disclose_goals**: The user discloses his/her goals by telling them to other people. In a scale from 1 (the statement does not describe me at all) to 5 (the statement totally describes me).
55. **friends_think_exercise**: The user's friends think that he/she has to move or exercise sufficiently. 1 = Totally disagree, 2 = Disagree, 3 = Disagree/agree, 4 = Agree, 5 = Totally agree, 6 = Not Applicable.
56. **students_think_exercise**: The user's fellow students find that he/she has to move or exercise sufficiently. 1 = Totally disagree, 2 = Disagree, 3 = Disagree/agree, 4 = Agree, 5 = Totally agree, 6 = Not Applicable.
57. **brother_think_exercise**: The user's fellow brother(s) and/or sister(s) find that he/she has to move or exercise sufficiently. 1 = Totally disagree, 2 = Disagree, 3 = Disagree/agree, 4 = Agree, 5 = Totally agree, 6 = Not Applicable.
58. **theone_compare_moves**: The one with whom the user usually compares himself/herself moves: 1 = Much less than him/her, 2 = Less than him/her, 3 = Same as him/her, 4 = More than him/her, 5 = Much more than him/her, 6 = Not Applicable
59. **theone_compare_is**: The one with whom the user usually compares himself/herself is: 1 = Much less healthy than him/her, 2 = Less healthy than him/her, 3 = As healthy as him/her, 4 = Healthier than him/her, 5 = Much healthier than him/her, 6 = Not Applicable
60. **last3m_set_free_time**: In the last 3 months, the user set free time for his/her daily move. 1 = Never, 2 = Barely, 3 = Sometimes, 4 = Often, 5 = Very often.
61. **last3m_move_insteadOV**: In the last 3 months, the user moved by bike or by feet, instead of car or the OV. 1 = Never, 2 = Barely, 3 = Sometimes, 4 = Often, 5 = Very often.
62. **last3m_move_withsomeone**: In the last 3 months, the user went to play or move with someone else. 1 = Never, 2 = Barely, 3 = Sometimes, 4 = Often, 5 = Very often.
63. **last3m_write_agenda**: In the last 3 months, the user wrote in his/her agenda to move or exercise. 1 = Never, 2 = Barely, 3 = Sometimes, 4 = Often, 5 = Very often.
64. **last3m_move_badweather**: In the last 3 months, the user made plans to move in bad weather. 1 = Never, 2 = Barely, 3 = Sometimes, 4 = Often, 5 = Very often.
65. **never_time_sports**: If the user never seems to have enough time to do sports. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
66. **sports_never_high_prior**: If sports never have a high priority in the users agenda/plans. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
67. **hard_find_time_sports**: If the user finds it hard to find time to do sports. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
68. **plan_sports_appointments**: If the user plans all his/her appointments about sports agreements. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
69. **plan_sports_fixed**: If the user plans all his/her sports activities at a fixed time. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
70. **plan_weekly_sports**: If the user plans his/her weekly sports activities. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
71. **not_sport_busy**: If the user does not sport as much as he/she is very busy. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
72. **everything_plan_sports**: Everything is planned around the user's sports activities, including work and college. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
73. **sports_routine**: The user always tries to do sports on the same day and at the same time, in order to become a routine. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
74. **write_sports_calendar**: The user writes his/her sports in a calendar. In a scale from 1 (the statement does not describe me at all), to 5 (the statement totally describes me).
75. **sports_suff_live**: Sufficient sports/ moving is something that fits the way the user wants to live. From a scale of 1 (totally disagree) to 7 (totally agree).
76. **sports_suff_suits**: Sufficient sports/ moving is something that suits the user. From a scale of 1 (totally disagree) to 7 (totally agree).

77. **myself_sport_enough**: The user sees himself/herself as someone who sports/moves enough. From a scale of 1 (totally disagree) to 7 (totally agree).
78. **app_useful**: The user finds a sports/moving app useful for increasing/maintaining his/her physical activity. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
79. **easy_skilled_app**: It is easy for the user to become skilled in using a sports/moving app. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
80. **like_sportsapp**: The user likes moving with a sports/moving app. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
81. **app_relatives**: If most friends/relatives of the user use a sports/moving app, then he/she will use it too. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
82. **app_resources**: The user has the necessary resources to use a sports/moving app. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
83. **app_knowledge**: The user has the necessary knowledge to use a sports/moving app. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
84. **app_intention**: The user has the intention to use a sports/moving app within the next 5 months. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
85. **app_fun**: A sports/moving app makes moving fun. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
86. **app_intimidating**: A sports/moving app is somewhat intimidating to the user. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
87. **app_able**: The user is able to use a sports/moving app when performing sports/moving activities. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
88. **app_privacy**: The privacy settings of a sports/moving app make it easy for the user to know what pesonal information the app receives from him/her. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
89. **app_safe_publish**: The user feels safe to publish information about his/her sports/moving activities via a sports/moving app. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.
90. **app_obligations**: The user thinks that there are too many obligations to use a sports/moving app. 1 = Totally disagree, 2 = Somewhat disagree, 3 = Neutral, 4 = Somewhat agree, 5 = Totally agree.

References

[1] NCD Alliance.(2016, May 9).The 4th leading risk factor for death worldwide: physical inactivity is an urgent public health priority. Retrieved from https://ncdalliance.org/news-events/blog/the-4th-leading-cause-of-death-worldwide-physical-inactivity-is-an-urgent-public-health-priority

[2] Opalytics. (2017, May 31). Combining Optimization with Machine Learning for Better Decisions [Webinar], *Gurobi Webinars*. Retrieved from http://www.gurobi.com/resources/seminars-and-videos/machine-learning-webinar-I

[3] Science Buddies. Sample Size: How Many Survey Participants Do I Need? Retrieved from https://www.sciencebuddies.org/science-fair-projects/references/sample-size-surveys

[4] World Health Organization.(2017, April 15). Physical activity. Retrieved from http://www.who.int/topics/physical_activity/en/

[5] W. H. O. (2010). Global Recommendations on Physical Activity for Health. Retrieved from http://www.webcitation.org/6ITBUAKj4

[6] Ahmed, M. U., & Loutfi, A. (2013). Physical activity identification using supervised machine learning and based on pulse rate. *International Journal of Advanced Computer Sciences and Applications*, 4(7), 210-217.

[7] Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, 25(2), 197-227.

[8] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees.

[9] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321-357.

[10] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. *In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. pp. 785-794. ACM.

[11] Conn, V. S., Hafdahl, A. R., & Mehr, D. R. (2011). Interventions to Increase Physical Activity among Healthy Adults: Meta-Analysis of Outcomes. *American Journal of Public Health*, 101(4), 751758. http://doi.org/10.2105/AJPH.2010.194381

[12] Direito, A., Jiang, Y., Whittaker, R., & Maddison, R. (2015). Smartphone apps to improve fitness and increase physical activity among young people: protocol of the Apps for IMproving FITness (AIMFIT) randomized controlled trial. *BMC public health*, 15(1), 635.

[13] Dwyer, K., & Holte, R. (2007). Decision tree instability and active learning. *Machine Learning: ECML 2007*, 128-139.

[14] Ellis, K., Kerr, J., Godbole, S., Lanckriet, G., Wing, D., & Marshall, S. (2014). A random forest classifier for the prediction of energy expenditure and type of physical activity from wrist and hip accelerometers. *Physiological measurement*, 35(11), 2191.

[15] Emery M., (2017, January 21). Why Kagglers Love XGBoost. Retrieved from http://matthewemery.ca/Why-Kagglers-Love-XGBoost/

[16] Fergus, P., Hussain, A., Hearty, J., Fairclough, S., Boddy, L., Mackintosh, K. A., ... & Radi, N. (2015, August). A Machine Learning Approach to Measure and Monitor Physical Activity in Children to Help Fight Overweight and Obesity. *In International Conference on Intelligent Computing*. pp. 676-688. Springer, Cham.

[17] Fisher, R. A. (1922). On the interpretation of χ^2 from contingency tables, and the calculation of P". *Journal of the Royal Statistical Society.* 85 (1): 8794. doi:10.2307/2340521.

[18] Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63(1), 3-42.

[19] Gregorutti, B., Michel, B., & Saint-Pierre, P. (2017). Correlation and variable importance in random forests. *Statistics and Computing*, 27(3), 659-678.

[20] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 1157-1182.

[21] Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1), 389-422.

[22] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *"The Elements of Statistical Learning (2nd ed.)"*. Springer. ISBN 0-387-95284-5.

[23] Ho, Tin Kam (1995). Random Decision Forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC.* 1416, pp. 278282.

[24] Ho, T. K. (2002). A data complexity analysis of comparative advantages of decision forest constructors. *Pattern Analysis & Applications*, 5(2), 102-112.

[25] Jeni, L. A., Cohn, J. F., & De La Torre, F. (2013). Facing imbalanced data–Recommendations for the use of performance metrics. In Affective Computing and Intelligent Interaction (ACII), *Humaine Association Conference* (pp. 245-251). IEEE.

[26] Jolliffe I.T. (2002). *Principal Component Analysis*, Series: Springer Series in Statistics, 2nd ed., Springer, NY, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4

[27] King, A. C., Hekler, E. B., Grieco, L. A., Winter, S. J., Sheats, J. L., Buman, M. P., ... & Cirimele, J. (2013). Harnessing different motivational frames via mobile phones to promote daily physical activity and reduce sedentary behavior in aging adults. *PloS one*, 8(4), e62613.

[28] Klein, M. C., Manzoor, A., Middelweerd, A., Mollee, J. S., & te Velde, S. J. (2015). Encouraging physical activity via a personalized mobile system. *IEEE Internet Computing*, 19(4), 20-27.

[29] Klein, Michel C. A., Adnan Manzoor, Julia S. Mollee. (2017). Active2Gether: A Personalized M-Health Intervention to Encourage Physical Activity. *Sensors (Basel, Switzerland)*, 17.6: 1436.

[30] Kruskal; Wallis (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association.* 47 (260): 583621.

[31] Lee, I. M., Shiroma, E. J., Lobelo, F., Puska, P., Blair, S. N., Katzmarzyk, P. T., & Lancet Physical Activity Series Working Group. (2012). Effect of physical inactivity on major non-communicable diseases worldwide: an analysis of burden of disease and life expectancy. *The lancet*, 380(9838), 219-229.

[32] Lockwood, P., & Kunda, Z. (1997). Superstars and me: Predicting the impact of role models on the self. *Journal of personality and social psychology*, 73(1), 91.

[33] McLachlan, Geoffrey J.; Do, Kim-Anh; Ambroise, Christophe (2004). *"Analyzing microarray gene expression data"*. Wiley.

[34] Michie, S., Abraham, C., Whittington, C., McAteer, J., & Gupta, S. (2009). Effective Techniques in Healthy Eating and Physical Activity Interventions: A Meta-Regression. *Health Psychology*, 28(6), 690-701. DOI: 10.1037/a0016136

[35] Mollee, J.S.; Middelweerd, A.; Kurvers, R.L. (2017). What technological features are used in smartphone apps that promote physical activity? A review and content

analysis. *Personal and Ubiquitous Computing: special issue on supporting a healthier lifestyle with e-coaching systems*, 21(4), 663-643. (in press)

[36] Mollee, J.S.; Middelweerd, A.; te Velde, S.J.; Klein, M.C.A. (2017) Evaluation of a personalized coaching system for physical activity: User appreciation and adherence. *In Health-i-Coach - Intelligent Technologies for Coaching in Health ; ACM 11th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth 2017)* (in press)

[37] Motta A. (2015, July 29). Cross Validation done wrong. Retrieved from http://www.alfredo.motta.name/cross-validation-done-wrong/

[38] Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7.

[39] Rice, John (1995). *"Mathematical Statistics and Data Analysis" (2nd ed.)*. Duxbury Press

[40] Rosenblatt, M. (1956). *"Remarks on Some Nonparametric Estimates of a Density Function". The Annals of Mathematical Statistics.* 27 (3): 832

[41] Saez, Y., Baldominos, A., & Isasi, P. (2017). A Comparison Study of Classifier Algorithms for Cross-Person Physical Activity Recognition. *Sensors (Basel, Switzerland)*, 17(1), 66. http://doi.org/10.3390/s17010066

[42] Shapiro, S. S.; Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika.* 52 (34): 591611. JSTOR 2333709. MR 205384. doi:10.1093/biomet/52.3-4.591. p. 593

[43] Singhi, S. K., & Liu, H. (2006, June). Feature subset selection bias for classification learning. *In Proceedings of the 23rd international conference on Machine learning* (pp. 849-856). ACM.

[44] Trost, S. G., Fragala-Pinkham, M., Lennon, N., & O'neil, M. E. (2016). Decision Trees for Detection of Activity Intensity in Youth with Cerebral Palsy. *Medicine and science in sports and exercise*, 48(5), 958-966.

[45] Tilo Wendler and Sren Grttrup. (2016). *Data Mining with SPSS Modeler: Theory, Exercises and Solutions* (1st ed.). Springer Publishing Company, Incorporated.

[46] Vanwinckelen, G., & Blockeel, H. (2012, January). On estimating model accuracy with repeated cross-validation. *In BeneLearn 2012: Proceedings of the 21st Belgian-Dutch Conference on Machine Learning* (pp. 39-44).

[47] Wilcoxon, Frank (1945). Individual comparisons by ranking methods. *Biometrics Bulletin.* 1 (6): 8083.