Prediction of Successful Participation in a Lifestyle Activity Program using Data Mining Techniques

Marten Pijl [a] Joyca Lacroix [a] Steffen Pauws [a] Annelies Goris [b]

[a] *Philips Research, User Experiences Group, High Tech Campus 34, 5656AE Eindhoven*
[b] *Philips New Wellness Solutions, High Tech Campus 27, 5656AE Eindhoven*

Abstract

The growing number of people worldwide living a sedentary life has led to increased efforts into the development of effective physical activity intervention programs. However, many participants of such programs fail to complete the program, and as a result do not attain the desired increase in physical activity. By detecting participants who are at risk of dropping out in advance, it may be possible to intervene and prevent the participant from dropping out. The work in this paper discusses the classification of such participants, using an activity database containing participant characteristics and activity data gathered through an accelerometer worn by the participants. Using genetic programming techniques, database variables (called markers) are combined to form new sets of markers. The predictive power of these new markers are compared to the markers present in the database, based on classification using principal component analysis and a k-nearest neighbor classifier. Results show that without the use of genetic programming to combine markers, classification class accuracy is approximately 64%. By combining markers through genetic programming, classification class accuracy increases to 72%, which equates to a reduction in the number of errors of approximately 23%.

1 Introduction

Nowadays, numerous people worldwide fail to meet the recommended levels of physical activity to maintain good health. Therefore, many efforts are invested in the development of effective physical activity promotion programs. These programs increasingly rely on wearable technology to measure activity patterns (e.g., accelerometers, pedometers). Often, the wearable device is combined with a web-based program that allows users to monitor their activity patterns over time and that offers them feedback about progress towards predefined behavior goals [1, 4]. At the same time, the uploading of the measured activity patterns by the participants in the program via the web provides the program developers with a large database of activity patterns and program usage data (connection times and frequency) of the participants. Such a dataset contains a wealth of information about the participants that may be helpful to discover patterns in user variables and data patterns that distinguish successful from unsuccessful program completion. Prediction of unsuccessful program completion at an early point in time can be useful for the timely delivery of motivating triggers or the tailoring of program content.

In this paper, the prediction of successful program completion based on a large database of user data collected from participants in a twelve-week lifestyle activity program is explored. Participants are labeled as either dropouts or non-dropouts; the former refers to participants who abandon the program prior to the end of the twelve-week period, while the latter refers to participants who completed the twelve-week program, irrespective of whether an increase in physical activity has taken place. This work examines the classification of dropouts ahead of time, in particular through the use of genetic programming to find new combinations of database variables with more predictive power than the variables possess individually.

The outline of the remainder of the paper is as follows. Section 2 provides an overview of the methods used, explaining the genetic programming algorithm in detail, and discussing the remaining steps in the

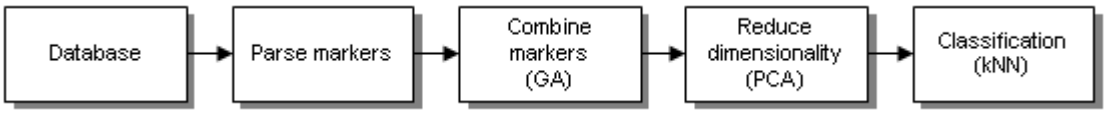

Figure 1: *Description of steps in the classification process, starting with the activity database, parsing markers from the database, combining markers through the genetic algorithm (GA), reducing the dimensionality of the marker set through principal component analysis (PCA), and classification of the data through the k-nearest neighbor algorithm (kNN).*

classification process. Subsequently, Section 3 lists the results obtained using the methods discussed in Section 2, in particular focusing on the performance of the genetic programming algorithm. In Section 4, the implications of the results are discussed. Finally, a short conclusion is provided in Section 5.

2 Method

For the study described in this paper, the data of 950 participants in a lifestyle activity program with a duration of twelve weeks plus an assessment period of one week was employed. During program participation, participants wore a small device that contains an accelerometer unobtrusively on their body throughout the entire day. The device measures participants' daily activity. The activity data is uploaded to a central database via a web-service that also provides feedback about activity performance. In addition to the activity data, the central database stores various types of user data including body characteristics, activity goals, and usage patterns of the web-service. The work detailed in this paper aims to classify participants as either dropouts or non-dropouts. In this context, a dropout is defined as a participant who does not record any activity during the last two weeks of the program. This can be due to either the participant not wearing the device, or not uploading the activity data to the web-based service. In this section, information gain is discussed as a measure of the predictive power of a decision boundary. Next, the various components and implementation of a genetic programming algorithm to combine database markers are discussed. Finally, an overview is provided of the remaining techniques used for dropout classification, including the application of principal component analysis and a k-nearest neighbor classifier. An overview of the entire classification process is provided in Figure 1.

2.1 Markers

By parsing the data present in the database, a number of participant variables are retrieved, referred to as markers. Markers generally fall into one of three categories. First, static markers represent participant properties which remain more or less static over time, such as a participant's height or gender. Second, assessment markers relate to participant behavior recorded during the assessment period of the program, and include markers such as the average activity recorded during this time. Third, dynamic markers represent (aggregates of) participant behavior during a specific time frame within the twelve-week program. Dynamic markers typically change over time and therefore depend on the time frame under consideration, for example the average activity in the first week versus the average activity in the last week of the program.

2.2 Information gain

Information gain, a notion common in the field of decision trees, is an indication of the increase in purity obtained by splitting a set given some separator or decision boundary. Given a single marker or a set of markers, a hypersurface decision boundary can be defined, splitting the set of participants into two separate subsets. Ideally, for some subset of markers a decision boundary exists which intersects the set of participants such that one of the resulting subsets contains all dropouts, and the other subset contains all non-dropouts. The resulting subsets are then considered 'pure'. In practice, such a decision boundary may not exist. However, the purity of a decision boundary's resulting subsets serves as an indication of the predictive power of the corresponding subset of markers over which the decision boundary is defined.

Given a subset of participants, the impurity of the subset S is given by the entropy impurity measure i:

$$i(S) = -\sum_j P(c_j) \log_2 P(c_j) \tag{1}$$

where $P(c_j)$ is the fraction of participants in S belonging to class c_j. From decision trees, it is known the reduction of impurity for a given decision boundary over a subset S is given by:

$$\Delta i(S) = i(S) - P_1 i(S_1) - (1 - P_1) i(S_2) \tag{2}$$

where S_1 and S_2 are subsets of S generated by the given decision boundary, and P_1 the fraction of participants in S present in S_1. When entropy impurity is used, Δi is often referred to as the information gain. A more detailed description of impurity and information gain can be found for example in [3], within the context of decision trees.

The issue remains of how to select an optimal decision boundary over a subset M of n markers. As the number of markers in M increases, the number of potential decision boundaries quickly becomes unmanageable. In order to ensure the optimization problem remains tractable, only decision boundaries of the form $e_M < c$ are considered, where e_M is an expression of the markers in M, and c is a constant. Finding the optimal value for c given e_M becomes a one-dimensional optimization problem, which can be solved in $O(h)$ for h participants. Finding the optimal subset of markers M, and the corresponding expression e_M, is considerably more difficult.

2.3 Genetic programming

Using current techniques, finding a globally optimal decision boundary expression e_M over a subset M of markers requires an exhaustive search over all $E = \{e_M\}$ with $M \in \wp(\{m\})$, where $\wp(\{m\})$ denotes the powerset over all markers m. As there is no limit to how often a marker in M may appear in e_M, E is infinitely large. Even if the number of appearances of a single marker in e_M was restricted, the number of sets in $\wp(\{m\})$ increases exponentially with the number of markers in $\{m\}$, and an exhaustive search of E would soon become intractable.

However, the optimal decision boundary expression can be approximated by genetic programming techniques. Here, a set of candidate expressions is maintained and updated over a number of rounds in an iterative process, consisting of

- fitness: each of the candidate expressions is assigned a score, called the fitness of the expression.
- selection: based on the fitness of each expression, a number of expressions are eliminated from the pool of candidate expressions.
- reproduction: the candidate expressions are altered and re-arranged to form a new set of expressions.

The genetic programming procedure aims to incrementally improve the candidate expressions, by means of small random changes and by combining good sub-expressions of several candidate expressions into new expressions. For a more detailed introduction to genetic programming, see for example [2].

2.3.1 Fitness

Given a set of participants, the fitness of a candidate expression can be determined by the information gain of that expression, as defined in Formula 2. In practice, the set of participants used for this purpose is a subset of all available participants, as it is advisable to maintain a separate train and test set to avoid contamination of the results due to possible overfitting. As decision boundaries take the form of $e_M < c$, the optimal value for c must be found to determine the fitness for each candidate expression e_M. Given a set of h participants, computing the results for an expression e_M for each participant will yield at most h unique results for that expression. If there are p unique results V (with $p \leq h$), there are at most $p - 1$ values for c to consider, as there are at most $p - 1$ permutations for subsets S_1 and S_2 (an empty S_1 or S_2 results in an information gain of 0). If V is sorted such that $V = v_1, v_2, \ldots, v_p$ with $\forall i, j (i < j \rightarrow v_i < v_j)$, the examined values for c can then be any which satisfy $c_1, c_2, \ldots, c_{p-1} | \forall i (v_i < c_i \leq v_{i+1})$.

In general, shorter expressions are preferred over longer expressions. Reasons for this include that longer expressions are not necessarily more informative than shorter expressions, and may be needlessly

complicated (this can be seen as a version of Occam's razor). More importantly, longer expressions are more likely to be the result of overfitting, and as such may lack generalization ability. For these reasons, a penalty to the fitness score is introduced, which is deducted from an expression's fitness for each operator in the expression beyond a fixed number. As a result, introducing additional operators to an expression will only result in a higher fitness if the increase in information gain due to these operators exceeds the penalties incurred.

2.3.2 Selection

To allow for the creation of new candidate expressions, a number of the current candidate expressions are removed. Generally, it is preferable to retain the expressions with the highest fitness values. However, it is often beneficial to also retain a number of candidates with lower fitness scores, in the hopes that these candidates will move away from the local optimum and discover other optimums in the search space. This feature, in part, differentiates genetic algorithms from local search algorithms such as gradient descent. To accomplish this, candidates to be retained are generally selected by chance, where candidates with higher fitness are more likely to be retained than candidates with lower fitness.

Numerous selection algorithms exist based around this principle, of which tournament selection is one of the more popular. In tournament selection, two candidates are selected at random, of which the candidate with the highest fitness is retained, and the other candidate is discarded. This process is repeated by randomly selecting two more of the candidates that have not been selected previously. Once there are no candidates left for selection, the selection process starts anew for those candidates that have been retained, until the desired number of discarded candidates has been reached (this may happen before a tournament round is complete, in which case any remaining candidates are automatically retained). The process of tournament selection is such that candidates with higher fitness are preferred. Due to randomly selecting sets of candidates however, candidates with lower fitness also have a chance to be retained.

2.3.3 Reproduction

After selection, a new set of candidate expressions is created using the selected candidates as a basis. In genetic algorithm, two main types of mechanisms govern this reproduction process. First, mutation of an existing candidate involves small random changes to the characteristics of the original candidate. Second, crossover involves combining attributes of two existing candidates to create a new candidate. To define such operators specifically for this application, the format of the candidate expressions is first defined. A given candidate expression e_M can be defined by the grammar

$$e_M \rightarrow (e_M \; operator \; marker) \mid marker \tag{3}$$

$$operator \rightarrow + \mid - \mid * \mid / \tag{4}$$

$$marker \rightarrow m \in M \tag{5}$$

In other words, valid expressions are any left-associative, infix expressions containing only addition, subtraction, multiplication and division as operators, and only marker variables as operands. Although the above grammar is fairly basic, it still allows for a large amount of variation in potential candidate equations.

Given the above format for candidate equations, the mutation mechanism can be implemented through random changes to either a marker variable in the equation or to an operator in the equation. For example, by changing a '$+$' into a '$-$'. The probability of such a change occurring for a specific marker variable or operator is given by the mutation rate parameter. Let $e_M = m_1 o_1 m_2 o_2 \cdots m_{n-1} o_{n-1} m_n$ be a candidate expression of length n, where the m_i represent marker variables, and the o_i represent operators. Then, mutation of a marker variable in e_M can be defined as a transformation $e_M \rightarrow e_M^*$ from a candidate expression e_M to an expression e_M^* where $e_M \rightarrow e_M^*$ is given by

$$m_1 o_1 m_2 o_2 \cdots m_k \cdots m_{n-1} o_{n-1} m_n \rightarrow m_1 o_1 m_2 o_2 \cdots m_k^* \cdots m_{n-1} o_{n-1} m_n \tag{6}$$

with $m_k \neq m_k^*$, for any k with $1 \leq k \leq n$. Similarly, mutation of an operator can be defined as a transformation $e_M \rightarrow e_M^*$ given by

$$m_1 o_1 m_2 o_2 \cdots o_k \cdots m_{n-1} o_{n-1} m_n \rightarrow m_1 o_1 m_2 o_2 \cdots o_k^* \cdots m_{n-1} o_{n-1} m_n \tag{7}$$

with $o_k \neq o_k^*$, for any k with $1 \leq k < n$.

Alternatively, a new candidate expression can be created through crossover. Here, two candidate expressions are selected at random, and each expression is split at a random point along its length. A new candidate expression is created by either attaching the front of the first expression to the tail of the second expression, or vice versa. Naturally, the resulting candidate expression must be a valid expression itself. This can be achieved by always splitting expressions after (or before) an operator. The impact of the crossover mechanism on reproduction is governed by the crossover rate parameter, which determines the fraction of new candidates created through the crossover mechanism. Let e_M^a and e_M^b be two candidate expressions of lengths n and l respectively. Specifically, let e_M^a and e_M^b be given by

$$e_M^a = m_1^a o_1^a m_2^a o_2^a \cdots m_{n-1}^a o_{n-1}^a m_n^a \tag{8}$$

$$e_M^b = m_1^b o_1^b m_2^b o_2^b \cdots m_{l-1}^b o_{l-1}^b m_l^b \tag{9}$$

Crossover can then be defined as a transformation $e_M^a, e_M^b \rightarrow e_M^*$ from expressions e_M^a and e_M^b to an expression e_M^*, where $e_M^a, e_M^b \rightarrow e_M^*$ is given by

$$e_M^a, e_M^b \rightarrow m_1^a o_1^a m_2^a o_2^a \cdots m_k^a o_k^a m_h^b \cdots m_{l-1}^b o_{l-1}^b m_l^b \tag{10}$$

for any k, h with $1 \leq k < n$ and $1 < h \leq l$.

Generally, genetic algorithms operate on candidates with some fixed length representation. However, expressions can be altered by removing or introducing new variables into the expression. This leads to the introduction of two additional reproduction mechanisms, not necessarily standard in genetic programming, insertion and deletion. The insertion mechanism inserts a new operator and marker pair into an expression at a random position. Similarly, the deletion mechanism removes a random operator and marker pair from an expression. The probability of insertion or deletion occurring in a new candidate is governed by the insertion rate and deletion rate, respectively. Insertion can be defined as a transformation $e_M \rightarrow e_M^*$ from e_M to an expression e_M^*, where $e_M \rightarrow e_M^*$ is given by

$$m_1 o_1 m_2 o_2 \cdots m_k o_k m_{k+1} \cdots m_{n-1} o_{n-1} m_n \rightarrow m_1 o_1 m_2 o_2 \cdots m_k o_k m_i o_i m_{k+1} \cdots m_{n-1} o_{n-1} m_n \tag{11}$$

for any k with $1 \leq k < n$, and with $m_i \in M$ and $o_i \in \{+, -, *, /\}$. Similarly, deletion can be defined as a transformation $e_M \rightarrow e_M^*$ from e_M to an expression e_M^*, where $e_M \rightarrow e_M^*$ is given by

$$m_1 o_1 m_2 o_2 \cdots m_k o_k m_{k+1} \cdots m_{n-1} o_{n-1} m_n \rightarrow m_1 o_1 m_2 o_2 \cdots m_{k-1} o_{k-1} m_{k+1} \cdots m_{n-1} o_{n-1} m_n \tag{12}$$

for any k with $1 < k < n$, or

$$m_1 o_1 m_2 o_2 \cdots m_{n-1} o_{n-1} m_n \rightarrow m_2 o_2 \cdots m_{n-1} o_{n-1} m_n \tag{13}$$

for $k = 1$.

2.4 Classification

Upon completion, the genetic algorithm yields a population of expressions which form the left hand side of potential decision boundaries. Conceivably, a classifier could be constructed by determining the optimal decision boundary from the expressions in the population, and classify participants depending on their location relative to this decision boundary. However, as a genetic algorithm is not restricted to local search only, it is possible that a number of expressions in the population score well by successfully partitioning different subsets of participants. Therefore, it may be advantageous to construct a classifier based on all or multiple expressions in the final population, rather than only on the fittest expression. The value of each expression for each participant can be calculated, resulting in a representation of each participant as a d-dimensional vector, given a population of d expressions. Participants can then be viewed as spatial objects in a d-dimensional Euclidian space.

As there may exist a large amount of correlation between dimensions, principal component analysis is used to obtain a more compact representation of the d-dimensional participant space. By analyzing the resulting principal components, the number of dimensions can be reduced, whilst retaining sufficient complexity to account for the majority of the variance in the participant data. For this work, sufficient dimensions were retained to explain at least 95% of the variance in the data. A classifier can then be constructed using any metric based classification algorithm. In order to ensure all dimensions are scaled comparatively, all

dimensions are normalized to have zero mean and unit variance prior to principal component analysis. In this paper, a k-nearest neighbor classifier was used, as it is a simple algorithm which is known to perform well on data with complex decision surfaces. Other classification algorithms (learning vector quantization and naive Bayes classifier) have also been examined, but offered no improved performance compared to k-nearest neighbor.

3 Results

Ideally, future dropouts are identified as such before actually dropping out of the program, as this would allow for timely interventions to prevent this. Therefore, the aim is to predict dropouts one week prior to dropping out, using data available at that time. First, for all known dropouts amongst the participants, the actual dropout day is determined. The dropout day is defined as the first day on which no activity is recorded for the remainder of the program. Similarly, the dropout week is the week of the program corresponding to the dropout day (the first seven days of the program form the first week of the program, and so on). Dropouts are then labeled according to their dropout week. As there are 12 weeks in the program, and dropouts per definition have no recorded activity in the last two weeks, a separate set of dropouts is created for 11 of the program weeks (dropouts in week 11 have the first day of week 11 as their actual dropout day). As more data on each participant becomes available as the program progresses, the aim becomes to classify each dropout in the week prior to their dropout week, using only data available up to (and not including) the dropout week.

With this in mind, dropouts were classified using the following scheme: when classifying dropouts in week n, only data available in week $n-1$ and earlier is considered. For example, when classifying dropouts in week 10, data recorded in weeks 10, 11 and 12 is not considered. In effect, this creates an expanding window of data, growing in size with each week classified. For each week, a separate set of genetic markers, principal component analysis and k-nearest neighbor classifier is constructed. 10-fold cross validation was used each week to obtain separate train and test sets. In 10-fold cross validation, data is separated into ten subsets, and iteratively one is selected as test set while the remaining subsets form the training set. Classifiers are trained and evaluated for each test and train set individually, and results are averaged to obtain a classification score for each week.

As the set of dropouts is now divided over 11 weeks (and the non-dropouts remain present throughout the program), the number of non-dropouts vastly exceeds the number of dropouts for any given week, creating a class imbalance. Using the standard measure of accuracy for classifier results (number of correct classifications divided by the total number of classifications) in such a situation leads to misrepresentation of classifier performance, as the accuracy on the non-dropout class dominates the classifier accuracy. Therefore, the class accuracy measure is used instead. Here, the accuracies over the individual classes (dropouts and non-dropouts) are computed first, and the class accuracy is defined as the mean over the accuracies of the individual classes. For instance, using a majority class baseline (all participants are classified as the most common class) yields a class accuracy of $\frac{100\%+0\%}{2} = 50\%$.

To determine the improvement gained from the use of genetic programming, classification results from markers obtained from genetic programming are compared to classification results from markers obtained directly from the participant database. As described above, dropouts are determined for each week separately, using data available at that time. In both conditions, principal component analysis is used to reduce the dimensionality of the participant data, after which the participants are classified using a k-nearest neighbor classifier. The optimal value for the parameter k is established experimentally for each week separately. Due to the sensitivity of the k-nearest neighbor algorithm to class imbalance, discussed in the previous paragraph, before classification dropout samples in the training set are randomly duplicated until the training set becomes balanced.

The results of dropout classification using basic markers obtained from the available participant data are shown on the left hand side of Table 1. The table shows class accuracy results for every week in which classification is performed. Over the eleven weeks, the recorded class accuracy scores remain fairly consistent, with slightly higher scores towards the start of the program. The final week of the program, however, scores considerably lower in class accuracy. As can be seen in the table, the average class accuracy over all eleven weeks for this method is 0.6371, i.e. approximately 64%.

Likewise, the results of dropout classification using markers obtained through genetic programming are shown on the right hand side of Table 1. Like before, the class accuracy results for each week are listed separately. Over the eleven weeks, results again appear fairly consistent, with the exceptions in week four,

Week	*Without GP*	*With GP*
Week 1	0.6873	0.7524
Week 2	0.6629	0.7396
Week 3	0.6667	0.7310
Week 4	0.6043	0.6484
Week 5	0.6443	0.7201
Week 6	0.6364	0.7528
Week 7	0.6389	0.6711
Week 8	0.6586	0.7838
Week 9	0.6165	0.8107
Week 10	0.6247	0.7334
Week 11	0.5679	0.5773
Average	**0.6371**	**0.7201**

Table 1: *Class accuracy per week of dropout classification using markers obtained from participant data, both with and without the use of genetic programming. The class accuracies obtained without the use of genetic programming are listed under 'Without GP', while the class accuracies obtained using genetic programming are listed under 'With GP'. Class accuracies were obtained using 10-fold cross validation.*

seven and most noticeably week eleven. When compared to the results obtained using only basic markers, the use of genetic programming results in an increase of class accuracy for every week of the program. As can be seen in Table 1, the average class accuracy over all weeks of the program when using genetic programming is 0.7201, or 72%.

4 Discussion

In the previous section, it was shown that classification using basic markers resulted in an average class accuracy of 0.6371, while classification using genetic programming resulted in an average class accuracy of 0.7201. When compared to a majority class baseline, which in a two class problem yields a class accuracy of 0.5, both of the classification methods discussed here clearly outperform this baseline. When comparing both methods, classification after genetic programming clearly outperforms classification based on basic markers only, as expected. When examining the error rates of both methods ($error\ rate = 1 - accuracy$), the use of genetic programming results in a reduction in the number of classification errors of approximately 23% ($\frac{(1-0.6371)+(1-0.7201)}{1-0.6371} * 100\%$).

When comparing the classification results on a weekly basis, the low class accuracy in week eleven forms an obvious outlier for both classification methods. A possible explanation may be in the definition of a dropout; those who drop out in week eleven are close to the decision boundary which separates dropouts from non-dropouts. For example, a participant who stops registering activity data at this late stage of the program may or may not be considered a dropout depending on whether they stopped recording activity a day sooner or later. As this decision boundary is somewhat arbitrary, it may algorithmically be difficult to distinguish between participants close to the decision boundary, yielding poor classification accuracy scores as a result. Note that for week four, a similar observation can be made with regard to class accuracy, although proximity to the decision boundary seems an unlikely cause in this case. Here, a possible explanation may be that the classification problem for this week is more difficult or complex compared to other weeks. Further investigation is required to determine if this is the case, and if so, why.

In contrast, other weeks show considerable deviation from the average classification accuracy for the genetic programming method, but no coinciding deviation in accuracy for the method using basic markers only. Examples of this include week seven and nine. Possible explanations are that either for these weeks the combination of markers adds an amount of information that is lesser or greater than average (for weeks seven and nine, respectively), or that the cause lies in inconsistent performance often inherent in genetic algorithms due to their stochastic nature. Again, this requires further investigation.

From a practical perspective, the above results show improved classification accuracy can be obtained through the use of genetic programming. However, there are also a number of disadvantages to this method. First, the large number of parameters can present difficulties for those unfamiliar with the algorithm. Fortu-

nately, experiments have shown classification results to be fairly invariant to small parameter changes, and there is no need to construct a different set of parameters for each week of the program. Second, genetic programming algorithms can become computationally expensive, especially as the number of participants and markers in the database increases. Fortunately, classification only requires a single run of the algorithm per week regardless of the number of classifications made, which should ensure the computational costs remain manageable. Finally, the use of possibly complex combinations of markers, rather than single markers, complicates human interpretation of classifier decisions. Indeed, examination of the combined markers generated by genetic programming rarely yields intuitive combinations. This may have consequences with regard to the acceptance and practical usefulness of these classifiers to prevent dropout through interventions in a lifestyle activity program.

For future implementation, a number of alterations to the described algorithms can be made. First, instead of principal component analysis, a supervised method for dimension reduction can be used (such as supervised PCA). As such methods reorganize the data based on class labels rather than variance, the resulting reorganized data might be better suited to metric classifiers such as k-nearest neighbor. Secondly, instead of considering all data up to the week to be classified, a window of the most recent data only could be considered for classification. Data from the first few weeks of the program may no longer be relevant in or representative of the final weeks of the program, and as such their inclusion may act as noise in the classifier. Finally, the current methods may be extended with a cost sensitive function for certain types of errors. For example, it may be considered more important to correctly classify dropouts compared to the correct classification of non-dropouts. Deciding on appropriate costs, however, remains a difficult issue.

5 Conclusion

The work here shows that the addition of genetic programming to dropout classification yields a considerable increase in classifier accuracy. It also shows that combining markers, even using only simple mathematical operators, can yield measures which are more informative with regard to a participant's future behavior than measures using these markers individually. The use of genetic programming is particularly useful when the number of basic markers is very large, making exhaustive approaches to combining markers infeasible, and when there are no clear intuitions regarding which or how markers should be combined. It was shown that when applied to prediction of participants' continued participation in a lifestyle activity program, the use of genetic programming to create combined markers resulted in a reduction in the number of classifier errors of 23%. As a more accurate prediction opens up the possibility of intervention for participants who are at risk of dropping out of the activity program, the number of participants completing the program is likely to increase as a result. The results presented in this paper emphasize that through combining markers considerable improvements in the prediction of dropouts can be attained, a task that can be considered at least challenging for humans. Therefore, the use of data mining techniques to predict participants dropping out of an activity program ahead of time seems a promising initial step toward the effective prevention of such dropout by timely interventions.

References

[1] A. Goris and R. Holmes. The effect of a lifestyle activity intervention program on improving physical activity behavior of employees. In *PERSUASIVE '08: Proceedings of the 3rd international conference on Persuasive Technology*, pages 23–34. Springer-Verlag, 2008.

[2] J. Koza and R. Poli. *Genetic Programming*. Springer US, 1992.

[3] P. Hart R. Duda and D. Stork. *Pattern classification, 2nd edition*. Wiley, 2000.

[4] L. J. Ware, R. Hurling, O. Bataveljic W. B. Fairley, L. T. Hurst, P. Murray, L. K. Rennie, E. C. Tomkins, A. Finn, R. M. Cobain, A. D. Pearson, and P. J. Foreyt. Rates and determinants of uptake and use of an internet physical activity and weight management program in office and manufacturing work sites in england: Cohort study. *Journal of Medical Internet Research*, 10(4):e56, Dec 2008.