FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

From Binary to Multi-class divisions: improvements on Hierarchical Divisive Human Activity Recognition

Tomás Vieira Caldas

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Pedro Mendes Moreira

Second Supervisor: João Manuel Paiva Cardoso

July 3, 2019

From Binary to Multi-class divisions: improvements on Hierarchical Divisive Human Activity Recognition

Tomás Vieira Caldas

Mestrado Integrado em Engenharia Informática e Computação

July 3, 2019

Abstract

In a world where there are more mobile devices than televisions, the amount and type of information a smartphone can provide is significantly increasing. Due to the use of sensors like the accelerometer and the gyroscope, a smartphone can retrieve specific data on-the-fly. Combining this with the factor that people take their device everywhere, and with the help of classification methods, the collected data can be converted into useful information about the user's daily activities and habits. People, often underestimate physically activity, and nowadays being physical inactive is almost considered a disease. With almost no progress made in order to reduce inactivity levels, friendly reminders can be introduced on your smartphone related to the lack of exercise.

In order to process the information needed to classify one's activity, we will resort to machine learning techniques, applying a divisive hierarchical approach. In this work, we will use a modified version of DIvise ANAlysis Clustering, a hierarchical clustering top-down technique. DIANA collects all the activities to be predicted in a big cluster and then starts to split them from a more generalized group to specific ones. Our version, will apply three classifiers (K-Nearest Neighbor, Decision Trees, Naive Bayes), instead of one, on each iteration and choose the one with higher accuracy. This way, instead of a unique way to classify an activity, we always have three (increased chance of correct classification). Once the technique is completed, a pruning phase will occur with the purpose of finding the optimum hierarchical structure to classify our data.

The goal of this project is to implement modifications to the already modified DIANA algorithm in order to have the best possible classification tree before moving to an online approach. The tree generated by the DIANA technique will have specific nodes replaced by their subtrees by turning binary into multi-class divisions. All these cuts will be analyzed and only the ones that provide a better classification accuracy will be accepted and executed. Additionally, since divisive methods have always been neglected by the scientific community, the results will be compared to a flat multi-class classification in order to see how well they behave.

With this work, we expect to provide the scientific community an improved version of the resulting DIANA hierarchical structure. That structure can later be exported to an Android mobile application to provide on-the-fly recognition on the physical activity the user is performing. From there all kinds of features can be implemented: target advertising, new eating habits or tracing of daily routines.

Keywords: Hierarchical Classification, Clustering, Machine Learning, Classifier, Pruning, Nodes

Resumo

Num Mundo onde há mais dispositivos móveis do que televisões, a quantidade e o tipo de informação que um *smartphone* pode fornecer vem aumentando significativamente. Devido ao uso de sensores como o *accelerometer* e o *giroscope*, um *smartphone* pode recolher dados específicos em tempo real. Combinando com o facto que as pessoas levam os seus dispositivos a todo o lado, e com a ajuda de métodos de classificação, os dados recolhidos podem ser convertidos em informação útil sobre as atividades diárias e os hábitos do utilizador. As pessoas, muitas vezes, subestimam a atividade física e, atualmente, a inatividade física é quase considerada uma doença. Com quase nenhum progresso feito de maneira a reduzir os níveis de inactividade, podemos ajudar a combater essa falta de consciência, introduzindo lembretes amigáveis no seu smartphone relacionados com a ausência de exercício físico.

De maneira a processar as informações necessárias para classificar uma atividade, recorreremos a técnicas de *machine learning*, aplicando uma divisão hierárquica. Neste trabalho, vamos usar uma versão modificada do *DIvise ANAlysis Clustering*, uma técnica hierárquica de *top-down clustering*. DIANA, junta todas as atividades a serem classificadas num grande *cluster* e, de seguida, começa a dividi-las de um grupo mais generalizado para grupos específicos. Na nossa versão, usaremos três classificadores (*K-Nearest Neighbour*, *Decision Tress* e *Naive Bayes*), em vez de um, e em cada iteração escolhemos aquele com maior precisão. Desta forma, em vez de uma única maneira de classificar uma atividade, há sempre três (maior chance de classificação correta). Assim que a esta técnica estiver concluída, vai occorer uma fase de *pruning* com o objectivo de encontrar a melhor estrutura hierárquica para classificar os dados.

Com este projeto pretendemos implementar modificações no já modificado algoritmo DIANA de maneira a obter a melhor árvore para classificação possível antes de seguir para uma abordagem *online*. Na árvore gerada pelo DIANA, específicos nós vão ser substituídos pelas suas sub-árvores ao transformar divisões binárias em divisões *multi-class*. Todos estes cortes vão ser analisados e só os que trouxerem vantagens em termos de classificação vão ser aceites e executados. Adicionalmente, como os métodos de divisão têm sido negligenciados pela comunidade científica, os resultados vão ser comparados com uma classificção *flat* de maneira a observar quão bem se comportam.

Com este trabalho, esperamos fornecer à comunidade científica uma versão melhorada da estrutura hierárquica resultante do algoritmo DIANA. Esta estrutura pode depois ser exportada para uma aplicação *Android* e reconhecer as ativadas praticadas pelo utilizador *on-the-fly*. A partir daí diversas *features* podem ser implementadas: *targer advertising*, novos hábitos alimentares ou simplesmente manter registo das rotinas diárias.

Palavras Chave: Classificação Hierárquica, Agrupamento, Aprendizagem Mecanizada, Classificador, Poda, Nós

"We all need to be mocked from time to time, lest we take ourselves too seriously."

George R.R. Martin

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations

ML	Machine Learning
AD	Average Dissimilarity
DIANA	DIvisive ANAlysis Clustering
FCT	Foundation of Science and Technology
INESC	Institute of Systems and Computer Engineering, Technology and Science
IMU	Inertial Measurement Units
PAMAP	Physical Activity Monitoring for Aging People
MST	Minimum Spanning Tree
DBSCAN	Desnity-Based Spatial Clustering of Applications with Noise

Chapter 1

Introduction

1.1 Context

The work proposed in this MIEIC dissertation intends to research the Human Activity Recognition providing conclusions based on improvements and experiments on the already existent modified DIANA algorithm. It falls in the context of the CONTEXTWA project, a research FCT-funded project involving INESC-TEC, INESC-ID and Altice Labs.

1.2 Motivation

Advances in sensing technology along with the increase of mobile devices (e.g., Apple Watch, Nike+), opened the opportunity to explore a relatively new, but wide, field of study. How can we take advantage of the fact that people take their smart devices everywhere? And how can the data from sensors of your smartphone be used to improve your daily lifestyle and those around you?

With the use of sensors, we can collect online data and classify the activity being performed by the user. As a result, we could provide information about his daily routine and what he could improve. This can prove to be really useful in the fight against the increase of sedentary lifestyle rates [How].

The World Health Organization stated that at least 1.4 billion adults do not have the necessary levels of physical activity, putting themselves at risk of chronic diseases [Lac]. Also, in [MHC+18] the researchers analyzed results from Exercise Treadmill Testing conducted between 1991 and 2014 and proved that cardiorespiratory fitness is associated with low-risk all-cause mortality.

Having this in mind, we could combine the improvements and excessive use of the mobile device with the data about a user daily routine, in order to provide information about the user lifestyle in many different ways. If a user is always watching TV we could warn him about

possible back problems, or if he has an inactive lifestyle give notifications about the risk of obesity or heart diseases [Hea].

Another possible application could be target advertising. If a user is an intense runner or cyclist, we could recommend TV programs, new accessories and content related.

Some studies have been made in this field, but more focused on the amount of activity a user does and at tracking progress with calories loss. Thus, this study will focus on improving a hierarchical classification model to improve the process of recognizing the activities themselves.

1.3 Objectives

With this dissertation we intend to create the best possible classification model, based on the DIANA method proposed in [KR90], that can later be used on a mobile device and provide a real-time activity recognition using data obtained from the smartphone sensors. For that, we will implement a Post-Pruning method on the hierarchical structure created by the original algorithm. The pruning method intends to find binary divisions that can be replaced by multi-class ones. This way, we will reduce the height of the tree, and consequently the chance of committing splitting errors. The idea behind this is that in the future works can use our idea to create a mobile application for HAR.

1.4 Document Structure

This paper is organized as follows: in Section 2 we will introduce some background information about the general Machine Learning topic, explain what is hierarchical clustering, the types of pruning methods and other relevant details necessary to fully understand our work.

In Section 3 we will present studies already made that in some way are related to ours. These will be divided into three categories: Human Activity Recognition, Clustering Techniques, and Human Activity Recognition through Hierarchical Clustering.

Section 4 presents the method that we are basing our work on, the Divisive Analysis Clustering, and how it decides how the splitting procedure goes. The modifications implemented on the original algorithm are explained as well.

Our proposed solution is demonstrated on Section 5 and the results obtained in Section 6.

Finally, in Section 7 we provide the conclusions we came to with our study and the future work that can be done.

Chapter 2

Background Information

2.1 Introduction

In this chapter we will introduce some fundamental concepts about Machine Learning regarding Hierarchical Classification, pruning methods, explain the differences between Agglomerative and Divisive Clustering and clarify which approach we are going to use. Also, we are going to state the dissimilarities between Multi-class and Binary divisions and different ways of collecting data and evaluating results.

2.2 Key Concepts

In order to fully understand how we are going to classify the user's activities, we need to comprehend basic notions about Machine Learning.

Despite the fact that ML is a vast and rapidly growing field of study, there are common concepts that apply to all its applications. In its core, it is an algorithm that predicts an outcome based on input variables.

As stated by Arthur Samuel, one of the pioneers on this area, way back in 1959 [Sam59].

> "[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed."

Among the different types of Learning practices, there is a divergence between Supervised and Unsupervised Learning that we need to address in order to proceed further in this study [Dif].

2.2.1 Supervised Learning

Supervised Learning is the most known and used Machine Learning practice. It is called Supervised because all the data is labeled [Sup], therefore the algorithm learns to predict the output from the input values. The goal is to learn a function so well, that given a sample of input data it effectively produces correct output data. It is mainly used in Classification and Regression problems.

2.2.2 Unsupervised Learning

The big difference to the method explained above is that in Unsupervised Learning the training dataset is not labeled [Sup], there is no corresponding output variable. In this case, the algorithm alone attempts to find patterns in the data. A good example of this technique is when the user wants to group data by their similarity, also known as Clustering. This application will be further explained in this paper.

There are other methodologies like Reinforcement Learning [Sze10]. One of the main applications is video games where agents learn through trial-and-error interactions with the environment and weight their results in a way to optimize performance on a long-term objective. This technique is interesting but is beyond the scope of this work since it does not fit our Human Activity Recognition paradigm.

2.3 Hierarchical Classification

Unlike most Machine Learning and Data Mining studies, where a flat classification is used, we are going to work on a more complex classification [SF11] problem using a hierarchical relationship between our classes.

Flat classifiers rely on a single decision from all the possible outputs without an inherent connection between them, whereas our approach uses a hierarchical structure (like a dendrogram or tree) exploiting the relationship between the existent classes.

Some example Hierarchies that follow this rule are the army ranks, where the member with most authority is at the top and the employees at the bottom, or the categorization of species, starting with the kingdom as the universal category followed up by phylum, class, order, etc...

Both these examples already have a predetermined hierarchy, whereas in our study we aim to find a way to generate a tree that displays the relationship between activities. For that we are going to use Hierarchical Clustering, a method that generates a hierarchy of clusters.

In order to generate the tree and create relations, the upper level nodes need to be split. This is what is going to establish parent-children connections between the classes. The splitting can be binary (two possible decisions) or multiple (2 or more decisions)

2.3.1 Binary Classification

Binary classification is the task of assigning an element to one of two categories (just like binary, 0 or 1), by measuring a series of attributes. Some of the most common examples are disease diagnosis and "fail or pass" tests.

The number of binary divisions is directly related to the lengthiness of the generated tree and to the number of possible divisions, which are two big factors in the classification accuracy (better explained in Section 5.1).

2.3.2 Multi-class Classification

In opposition to binary, multi-class classification splits one parent node into two or more children. However in a hierarchical classification approach, several multi-class divisions can generate a tree with a scarce number of parent-children relations.

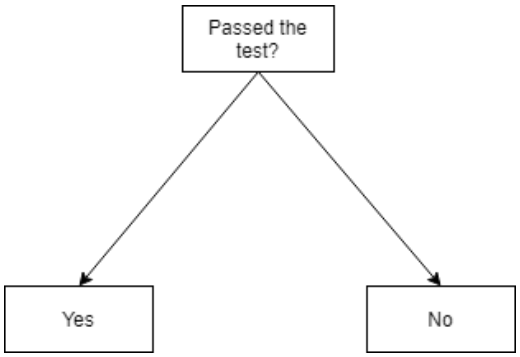

Figure 2.1: Binary Classification

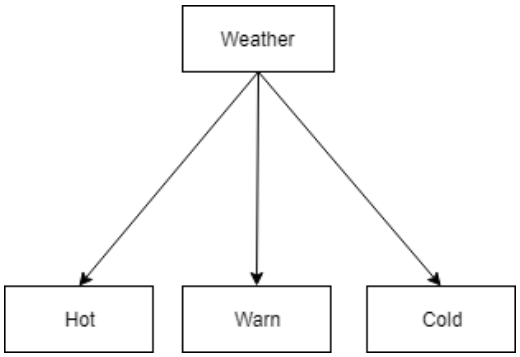

Figure 2.2: Multi-Class Classification

2.4 Hierarchical Clustering

Hierarchical Clustering builds a hierarchy of sets (called clusters), in such a way that all the objects in a set share a higher similarity with each other than with those in other clusters. Typically, the generation of these hierarchies falls into two types: Agglomerative or Divisive Clustering. The first starts "from the bottom" and moves up the hierarchy merging two clusters at a time, on the other hand, the divisive method starts "from the top" and goes down the hierarchy dividing larger clusters into smaller ones.

Like in all complex methods as this one, there are several paths we can take, as in, several ways a cluster can be divided (Divisive), or two can be agglomerated (Agglomerative). In order to make that decision, a measure of dissimilarity between clusters must be set. Typically, hierarchical clustering methods resort to an appropriate metric (formula to calculate the distance between two clusters) and a linkage criterion (from where the distance is calculated). These techniques will be further investigated in sections 2.4.3.1 and 2.4.3.2, but first we need to clarify some differences between the two types of Clustering.

2.4.1 Divisive Clustering

In our Human Activity Recognition work, we are going to separate all the activities to be predicted from a more generalized group to specific ones until each activity is individually separated. By the end of this process we will obtain the generated tree from the successive iterations. This tree will illustrate the activities hierarchy, therefore providing information about the relations and similarities between all the exercises.

An example of this is presented in Figure 2.3. Firstly all the activities are separated in Static or Moving sets, and then, in the activities themselves (being this the last iteration). Beware that this representation is just a vague example of an Human Activity Recognition tree.

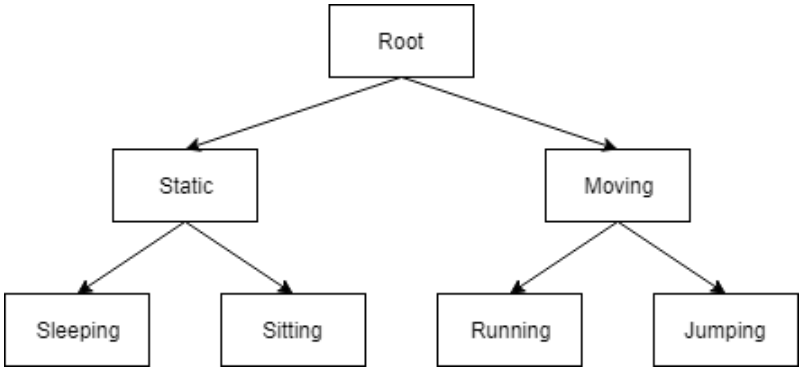

Figure 2.3: Example of a generated tree

This is called a top-down approach or simply Divisive Clustering. As we stated before and like the name suggests, we start with a big cluster containing all the activities (the root node) and level by level we continuously divide each cluster into two smaller ones. The algorithm stops once each cluster contains one and only one activity.

These clusters need to follow two rules [20009]:

1. **Mutually exclusive** — No activity in one cluster can be in more than one of his subclusters.

2. **Jointly exhaustive** — Every activity is in some subcluster.

2.4.2 Agglomerative Clustering

Another method is Agglomerative Clustering or bottom-up approach. In this case, we start with a set of singleton clusters, and successively merge pairs of clusters by similarity levels up til we only have one cluster that contains all the information.

Normally this technique is visualized as a dendrogram as shown in Figure 2.5. To understand a dendrogram, the key is to focus on the height that two clusters are merged. Being this a bottom-up approach we can see that A and B are the most similar, as the height of the link that joins them is the smallest, and that the biggest difference is between the <A B> cluster versus the <C,D,E,F> one.

These dendrograms are created, most of the times, from the analysis of a Scatter Plot. In the Figure 2.4 we present a Scatter Plot that could generate the dendrogram provided in Figure 2.5. Note that sometimes occurs information loss in a conversion like this. For example, the dendrogram suggests that C is closer to D than to B, because the clusters that contain them are merged first. However, if we analyze the original data from the Scatter Plot, we recognize that this is not true.

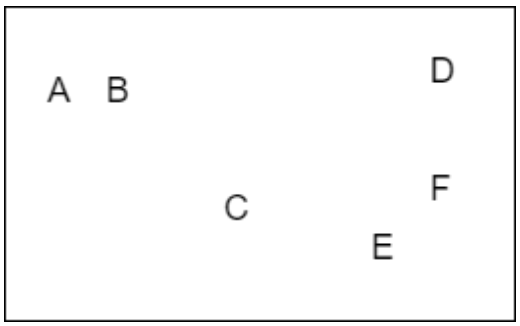

Figure 2.4: Example of Scatter Plot

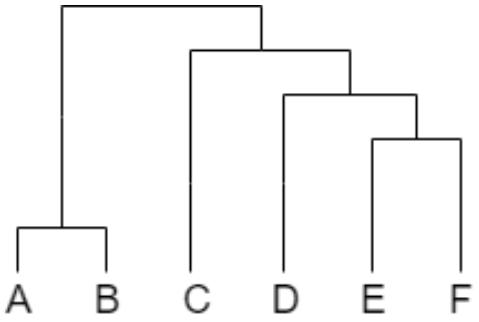

Figure 2.5: Correspondent dendrogram

From what we presented so far about these two methods it is clear that the main difference between them resides in the approach itself, either top-down or bottom-up. There is room for further investigation, comparing aspects such as effectiveness or efficiency, however that is not part of this study.

2.4.3 Cluster Dissimilarity

In each step of a Clustering method, considering binary classification (Section 2.3.1), a cluster is divided into two subpartitions. As proposed in [ECS65], there are exactly $2^{n-1}-1$ ways of splitting a set of n objects into two sets. For example, a group of 3 classes <A,B,C> can be divided in three different ways (Table 2.1).

In order to choose the way the clustering is done, it is required to work out the proximity matrix, a symmetric matrix containing the distance between each point. For that matter, we need a metric [SM12] to calculate the distance between the clusters, and a linkage criteria to define the inter-clusters proximity (from where to calculate the distance) [CNDN] [PG] [Hie].

2.4.3.1 Linkage Criteria

The Linkage Criterion determines from where the distance is calculated, which specifies the dissimilarity of pairs. Some commonly used methods to compute the distance are:

- **Single Linkage** — In single linkage, the distance between two clusters is defined as the shortest distance between two points, one point per cluster.
- **Complete Linkage** — Complete linkage is the opposite of the above. The distance is defined as the longest distance between two points in each cluster.
- **Average Linkage** — In average linkage the distance is the average distance between all pairs of objects, where each pair consists of one object from each cluster. Which means, the sum of all pairs distances, divided by the sum of the clusters size.
- **Ward's Method** — In Ward's Minimun Variance Criterion, the distance is defined as the error sum of squares (SSE) between the two clusters over all of the variables. In other words,

Table 2.1: Cluster <A,B,C> possible divisions

Sub Cluster 1	Sub Cluster 2
<A>	<B,C>
<B>	<A,C>
<C>	<A,B>

it pretends to merge the two clusters, estimates a centroid for the resulting cluster and then calculates the sum of the square error (SSE) of all the points from the new center.

- **Centroid Method** — Centroid Clustering is similar to the Average Linkage, but in this case, the centroid of each cluster is calculated and the distances is defined as the distance between the two centroids.

2.4.3.2 Metrics

After selecting a method from where to calculate the distance between two clusters, we need to choose a formula to compute that distance. Choosing a well-suited distance measure is crucial to the resulting hierarchy, as some elements may be closer given one measure, but farther away given another. To that matter, here are the most typically used formulas:

- **Euclidean Distance** — This used to calculate the distance between the points with a ruler. It is the straight line between them.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2} \tag{2.1}$$

- **Manhattan Distance** — The Manhattan distance calculates the distance that would be travelled from one point to another considering a grid path. In other words, the sum of the lengths between the points travelling parallel to the X and Y axis.

$$d(x,y) = \sum_{i=1}^{n}|xi - yi| \tag{2.2}$$

- **Jaccard Distance** — The Jaccard distance, is obtained by dividing the difference of the union sizes and the intersection of two sets by the union size.

$$d(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \tag{2.3}$$

- **Maximum Distance** — Considering two vectors x and y, this measure returns the maximum distance between two points (xi and yi) of those vectors.

$$d(x,y) = max_i(|xi - yi)| \tag{2.4}$$

2.5 Pruning

In almost all classification methods the misclassification error rate is never zero [Fra00]. High tree complexity, insufficient set of attributes or erroneous attribute values are just some examples that can cause poor classification performances.

In cases of high tree complexity, a large number of branches and consequently an excessively number of rules, many of which with low predictive value [Bra] due to noise, may lead to overfitting. To tackle this problem, and find the simplest tree that fits our data, pruning is necessary.

Pruning is an essential step to optimize the efficiency and increase the classification accuracy [PPP12], that consists of reducing the size of the tree by removing specific sections (normally the ones with little classification power).

There are two types of pruning, pre and post. Like the names suggest, pre-pruning happens during the process of generating the tree while post-pruning occurs after the tree is completely generated.

2.5.1 Pre-Pruning

As mentioned above, this method works as the tree is being built and the possible splitting steps are taken into consideration using some termination condition, normally a prespecified threshold. For results below that threshold, the development process of the branches that were taken into consideration is terminated, whereas for higher results the branches are expanded.

The problem here is defining a good termination condition. On one hand high threshold values generate oversimplified trees and on the other hand low values can result in highly complexional trees. Another example where the splitting procedure can be aborted is when the data is not enough to have a good prediction.

Minimum number of object pruning and Chi-square pruning are some of the most known pre-pruning methods. However, the algorithm that we are working on (Chapter 4) already constructs a hierarchical tree without their use. Considering that modifying the algorithm is not within the scope of this work, pre-Pruning methods were not considered (they can be further explored in [RA13]). Regardless, post-pruning methods can reveal themselves very promising since they work on top of the resulting tree.

2.5.2 Post-Pruning

Post-pruning techniques waits for the full grown tree and then follows a set of rules to remove certain nodes that may bring performance advantages. Following almost the same process as in Section 2.5.1, values are computed after and before the cut. If the new results are better the current subtree is replaced by the leaf nodes, if not the tree remains the same and other possible cuts are analyzed.

Although harder to implement than pre-Pruning, post-Pruning works better most of the times and it is the technique we are going to choose.

2.5.3 Post-Pruning Methods

Post-Pruning can be computationally expensive if all the pruning possibilities are analyzed, therefore, some methodologies have been presented in order to reduce the number of cuts to be considered. Before we explain how pruning was introduced in our work, first we have to explain some of the existent techniques.

2.5.3.1 Reduced Error Pruning

This method was proposed in [Qui87], and starting from the bottom (bottom-up type) prunes all nodes that when pruned do not decrease the tree's prediction accuracy (replacing them with its most popular class) [BA]. Although this is a very simple approach, according to the analysis in [EMS] this method produces the smallest pruned tree with the lowest error rate. Also, it performed just as good as most of the other methods in accuracy and better than most in relation to tree size.

2.5.3.2 Error Complexity Pruning

The goal of this approach is to find the error complexity of each node by calculating the error cost [PPP12].

$$R(t) = r(t) * p(t) \tag{2.5}$$

First, following the equation 2.5, the error cost for each node is calculated, where:

$$r(t) = \frac{n^o \textit{ of examples misclassified in the node}}{n^o \textit{ of examples}} \tag{2.6}$$

$$p(t) = \frac{n^o \textit{ of examples in the node}}{n^o \textit{ of examples}} \tag{2.7}$$

Then, for a not pruned non-leaf node T, the error cost of the subtree T is calculated using the Equation 2.8.

$$R(T) = \sum_{i=n^{o}leaves} R(i) \tag{2.8}$$

Finally, we can obtain the error complexity for each node using the Equation 2.9.

$$a(t) = \frac{R(t) - R(T)}{n^{o}\ of\ leaves - 1} \tag{2.9}$$

Once we have all the values, just choose the node with the minimum error complexity value, prune it and repeat the process.

2.5.3.3 Minimum Error Pruning

The method developed by [CB] is another bottom-up approach which aims to find a subtree that can be replaced by its leaf. For that, the estimated error rate (Equation 2.10) is computed with and without pruning. Having those two values, we make our decision based on the smaller value [PPP12].

$$E(t) = \frac{n_t - n_{t,c} + k - 1}{n_t + k} \tag{2.10}$$

Where:

- k = n^{o} of classes
- a_{n_i} = n^{o} of instances in node t
- $n_{t,c}$ = n^{o} of instances assigned to class c in node t

2.6 Retrieving data through sensors

Almost all Android devices have built-in sensors capable of measuring and retrieving raw data with high precision and accuracy. By measuring certain types of these sensors, we are able to monitor changes in the environment, infer user gestures or calculate and report the surrounding temperature. Only considering the Android platform, three broad categories of sensors are supported: [Sen]

2.6.1 Motion Sensors

These sensors let us monitor the motion of a device and are useful for measuring movements like tilt and shake, which tend to be a reflection of the actual user movement. The acceleration and

rotation are also considered motion sensors, whose job is to measure the forces along three axes. Some examples are the accelerometer, gyroscope and rotational vector sensors [Mot].

2.6.2 Environment Sensors

This type of sensors capture the state of the environment and can be used to monitor the humidity, light intensity, ambient pressure and temperature. A good example is the automatic screen brightness adjustment [Env].

2.6.3 Position Sensors

Position sensors, like the geomagnetic field sensor and the accelerometer, measure the position of the device relative to the World [Pos].

2.6.4 Vision-based Sensors

Vision-based sensors, like the name suggests, require some sort of camera to capture the series of images. Kinect for Xbox is a good example since it can sense human body motion in real-time with good performance. Despite that, cameras are not well accepted due to privacy, and environmental issues, such as illumination, noise and ambient occlusion. As a result, these sensors are only used for safety surveillance or gaming [WCW+].

2.6.5 Physiological sensors

Physiological sensors are not incorporated on Android devices and are sometimes referenced as Wearable Devices. They can be used to measure body temperate and heart rate (data that the *smartphone* can not retrieve) or simply get better acceleration or orientation values.

In this work, since we are not running our technique on an Android device, we will use a dataset with already processed data that was gathered using this type of sensors, which will be explained in detail in Section 4.5.

2.7 Evaluation of results

One of the most important parts of this kind of researches is the evaluation of results, since it is the ultimate proof that the developed approach provides a good solution or some kind of improvement.

Typically, in supervised learning, a confusion matrix is built (Table 4.8), which is a table that allows us to visualize the performance of our method. In a binary classifier (it can easily be extended to a multi-class classification) the table is divided into 4 quarters: [Sim]

- **True Positives (TP)** — The cases in which we predicted 'A' and the real label is 'A'. The top left corner.

Table 2.2: Example of final Confusion Matrix for binary classifier

	Predicted: A	**Predicted: Not A**
Actual: A	50	10
Actual: Not A	5	100

- **False Positives (FP)** — The cases in which we predicted 'A' and the real label is not 'A'. The top right corner.

- **False Negatives (FN)** — The cases in which we predicted not 'A' and the real label is 'A'. The bottom left corner.

- **True Negatives (TN)** — The cases in which we predicted not 'A' and the real label is not 'A'. The bottom right corner.

The total number of classifications is the sum of all quarters. In this case there are 50 + 10 + 5 + 100 = 165 instances to be classified. Using this matrix it is possible to compute more than simple proportions of correct classifications. We can estimate accuracy, precision, recall and f-measure, among others [Con].

2.7.1 Accuracy

Accuracy is the most intuitive evaluation measure, simply because it creates a proportion between the number of correct values and the total number of instances. However, such a simple solution can not tackle all the problems. Accuracy does not distinguish between the types of errors (False Positives and False Negatives), which in a cancer classification problem can cause serious problems between cancerous (positive classes) and non cancerous patients (negative class) [Jap06].

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{2.11}$$

2.7.2 Precision

This measure evaluates to what extent the classifier was correct at predicting examples as positives, so it does not fall into the problem specified in 2.7.1. In contrast, it does not evaluates how well a classifier works at deciding that a negative instances is actually negative. It can be seen as a measure of exactness [GM12].

$$PRecision = \frac{TP}{TP+FP} \tag{2.12}$$

2.7.3 Recall

Similar to Precision, Recall calculates how well the instances that had to be classified as positives actually were. So, also like Precision, the problem explained in 2.7.1 does not need to be taken in consideration. It can be seen as a measure of quantity [GM12].

$$Recall = \frac{TP}{TP + FN} \tag{2.13}$$

2.7.4 F-Measure

Not as easy to understand as the past three measures, F-Measure is the weighted average of Precision and Recall, taking into account False Positives as well as False Negatives [Acc].

$$F - Measure = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{2.14}$$

If False Positives and False Negatives have similar values it is better to go for accuracy, however if they have different numbers Recall and Precision, and therefore F-Measure, are the way to go.

2.8 Sliding Windows

When talking about Human Activity Recognition, we need to take into account crucial points such as the large volume of data collected per second and the characteristics of the activity signals. Points like this originate the challenging problem of accurately relating the sensor data with the specific activity.

Having this in mind, determining the appropriate window size is crucial to have promising results in activity recognition. On one hand small windows can slice the activity into multiple windows, whereas on the other hand longer windows can include more than one activity [NSW15].

As we will present, there are multiple approaches to define a suitable window size [Din18] [Car16].

2.8.1 Non-Overlapping Sliding Windows

Non-Overlapping Sliding Windows, like the name suggests, never share information with each other, according to a pre-specified static window size. For example, if the number of instances received is 6 and the window size is 3, each window will have three consecutive instances - one window with the instance 1,2,3 and the other with 4,5,6.

2.8.2 Overlapping Sliding Windows

In contrast to the method explained above, Overlapping Sliding Windows make it possible for two windows to have the same instance. Using the same example, one window could have the instances 1,2,3 another with 2,3,4 and a third one with 4,5,6.

2.8.3 Dynamic Sliding Windows

Another type of approach is to dynamically change the window size in order to fit for lengthier signals. The objective is to choose a window size long enough to be able to have one instance per window for the largest number of instances possible. Then, in the case we receive an activity with a signal longer than the window, the last one would expand to accommodate the entire activity [NSW15].

Chapter 3

Related Work

3.1 Human Activity Recognition

For the past ten years, Human Activity Recognition works have been an active field of study due to the rise of microelectronics, easy to carry sensors and available data. Several studies have been carried out in order to improve the accuracy in the most realistic conditions possible.

In 2013, Óscar et al. [SLL] conducted a survey on Human Activity Recognition using Wearable Sensors that analyzed and classified twenty eight systems. First, these systems were divided into two types according to the learning process, unsupervised and supervised. And then, for the last case in online or offline approaches.

Bearing in mind that our goal is to find the simplest tree that can fit our data while achieving better results than the modified DIANA algorithm, our work falls into the supervised offline approach. While analyzing the most relevant techniques, the authors summarize the approaches in a table similar to Table 3.1.

While some studies may present really good accuracy levels, they either do not follow our hierarchical clustering technique, use a different set of sensors or recognize a narrow range of activities. Studies [CAL10], [JGK+08] and [CYL+08] conducted their experiments inside a laboratory under controlled conditions, for instance. Once applied to real-life conditions, the model may perform worst under new and diverse circumstances. On the other hand [MM11], [PA08] and [VLL+11] do not give any information related to the data collection procedure. [HJ08] and [HJ] achieved an accuracy of 97 %, however, only four activities are considered: walking, running, being still and jumping. Regarding physical aspects, these activities are quite different, which makes it easier to classify. In [MWA+07] and [ZS09] the data was collected only from one person and in [HNK09] from four. Considering that each person is different and moves in its own way, this number of subjects is insufficient in order to make global predictions.

[PEK+06] used numerous sensors on the subject body, a 5 kg rucksack with a laptop as the integration device and a speech recognizer recording the user's voice, making this system highly

Table 3.1: State-of-the-art for offline supervised systems. Adapted from [SLL].

References	Activities	Sensors	Obtrusive	ID	Experiment	Flexibility	Features	Learning	Accuracy
[BI]	AMB, DA	ACC	High	None	NAT	MNL	TD, FD	KNN, C4.5, NB	84 %
[HNK09]	AMB	ACC	Low	Laptop	N/S	MNL	HAAR filters	C4.5	99,91 %
[PEK+06]	AMB, DA	ACC, ENV, VS	High	PC	NAT	MNL	TD, FD	DR, KNN	86 %
[HJ08]	AMB	ACC	Low	PC	N/S	MNL	AR	KNN, SVM	92,5 %
[HJ]	AMB	ACC	Low	PC	N/S	MNL	DCT, PCA	SVM	97,51 %
[ZS09]	AMB, TR	ACC, GYR	High	PC	N/S	SPC	AV, 3DD	HMM	90 %
[AB10]	AMB	Electroded	High	None	NAT	MNL	PCA, SFFS	BN, LS, KNN, DTW, ANN	87 - 99 %
[CAL10]	UB	ACC	High	PC	LAB	MNL	TD	LDA	77 %
[MM11]	DA	ACC	Low	None	N/S	SPC	DTW	DTW ensemble	84,3 %
[PA08]	AMB, DA	ACC	Medium	N/S	N/S	Both	Relative Energy	NB, HMM	95-97 %
[VLL+11]	AMB, DA	ACC	Medium	N/s	N/S	N/S	TD	SMCRF	88,38 %
[LPLP12]	AMB	ACC, VS	Medium	Cellphone	NAT	MNL	TD, FD, PR, TF	ALR, BAgging, C4.5, NB, BN	95,7 %
[KYLT10]	AMB, TR	ACC	Medium	Computer	NAT	MNL	AR, SMA, TA, LDA	ANN	97,9 %
[JGK+08]	AMB	ACC, SPI	High	Tablet	LAB	Both	TD, FD	CART, KNN	86 - 95 %
[CYL+08]	AMB, DA, HW	ACC	Medium	N/S	LAB	MNL	TD, FD	FBF	93 %
[MWA+07]	AMB, MIL	ACC	High	Laptop	Both	SPC	TD, FD	Boosting	90 %

obtrusive. The authors in [LPLP12] presented a system with 100 % accuracy for some activities that only required one sensor and a mobile phone, nevertheless, only five ambulation activities were considered and the mobile application was developed with Java Micro Edition, which has fallen into disuse.

There are also some relevant and interesting studies such as [BI] and [KYLT10]. The first recognizes over twenty activities, including watching TV and working at the computer, in a naturalistic environment. It uses five bi-axial accelerometers initially, yet they proved that with only two similar results could be achieved. The latter work is very interesting and deserves further reading on possible future work as it classifies not only ambulatory activities but also transitions among them, such as sitting to walking or sitting to lying.

3.2 Clustering techniques

As seen, there is a large number of metrics and criteria (Section 2.4.3), that together with different approaches and data types can originate a wide range of techniques for creating clusters.

In [ZGH+] the authors propose two Euclidean MST based algorithms. One k-constrained, where edges of a generated MST with representative points - cluster point closest to the geometric centroid of that cluster - are removed (following a predefined criterion) until k clusters can be created from sets of representative points. The second is unconstrained and removes the edges that affect the most the tree's overall standard deviation of edge weights.

The popular k-means algorithm defines k as the number of centroids you will need in your dataset [Und], and then, tries to allocate all the data points to the nearest cluster, while taking into account keeping the centroids as small as possible. In [LVJV03] the global k-means algorithm is presented, which is used as a local search procedure and works in an incremental way to add one new cluster iteratively, discarding the need to randomly select initial values for the centroids. Also, some modifications are explored in order to reduce the computational performance.

Fuzzy c-means, by Dunn in [Dun73], is very similar to k-means as it also aims to minimize the objective function. However, since it is a fuzzy algorithm, one instance can be placed in more than one cluster, so, values like the cluster fuzziness must be included in the objective function formula.

Proposed by Fukunaga and Hostetler, [FH75], Mean Shift Clustering in contrast to k-means makes no assumptions related to the number of clusters or the shape of the distribution. On the other hand, it uses sliding-windows in an attempt to find the densest areas until there is no direction at which a shift can increase the number of points included in the window [Der05].

Similar to Mean Shift Clustering, DBSCAN a density-based algorithm [EKSX96], does not use a pre-set number of clusters. On the contrary, it groups together points by their closeness, considering parameters like the minimum number of points and the maximum distance. However, unlike Mean Shift Clustering, it treats outliers as noise, thus they are not assigned to clusters whose points are very distinct.

3.3 HAR through Hierarchical Clustering

Most of the studies already presented used flat classifiers to predict the activity executed. However, researches have also been made using hierarchical approaches.

[KVKEK11] presented a two-layer hierarchical model for predicting activities from a sequence of actions. All the activities were performed inside a house equipped with a sensor network. For the structure, the first level represents the activities and the bottom layer the actions clusters.

A model to predict complex exercises such as pole vaulting was proposed in [GHS12]. The considered activities consist of a variable number of spatiotemporal parts, so, fragments of moving objects (tracklets) are clustered in order to generate a tree that can create a "video" correspondent of the activity.

In [KYLT10] the authors propose a hierarchical scheme with relations between the activity and its state (transition, dynamic or static). At a lower level, the activity state is recognized, and from there, with the use of signals and tilt angle, the activity itself is predicted.

Another study, [WCW+], uses a top-down approach on a decision tree to classify the human activities. Starting at the root node, the activities are split into clusters according to their movement state (static or dynamic). From there each cluster is split into a set of n leaf nodes, where n is the number of activities in that cluster. Despite being similar to ours, this study creates a decision tree

from activities (standing, sitting, lying, walking, going upstairs, going downstairs) that are easily distinguishable. Thus it does not provide any guarantees in cases of several similar activities. [2]

Table 3.2: Abreviations and acronyms. Adapted from [SLL]

Abreviation	**Meaning**
ACC	Accelerometers
AMB	Ambulation Activites
ANN	Artificial Neural Network
ALR	Additive Logistic Regression classifier
AR	Auto-Regressive model coefficients
AV	Angular velocity
BN	Bayesian Network classifier
CART	Classification and Regression Tree
DA	Daily Activities
DCT	Discriptive Cosine Transform
DT	Decision Tree-based classifier
DTW	Daily Dynamic Time Warping
ENV	Daily Environmental Sensors
FBF	Fuzzy Basis Function
FD	Frequency-domain featurest
GYR	Gyroscope
HMM	Hidden Markov Models
HW	Housework Activites
ID	Integration Device
KNN	k-Nearest Neighbors classifier
LAB	Laboratory controlled experiment
LDA	Linear Discriminant Analysis
LS	Least Squares algorithm
MIL	Military Activities
MNL	Monolithic classifier
NAT	Naturalistic experiment
NB	Naive Bayes classifierr
N/S	Not Specified
PCA	Principal Component Analysis
PR	Polynomial Regression
SFFS	Sequential Forward Feature Selection
SMA	Signal Magnitude Area
SMCRF	Semi-Markovian Conditional Random Field
SPC	User-specific classifier
SPI	Spiroergometry
TA	Tilt Angle
TD	Time-domain features
TF	Transient Features
TR	Transition betweem activities
UB	Upper body activities
VS	Vital sign sensors

Chapter 4

Divisive Analysis Clustering

Ever since the beginning of Clustering techniques, Divisive methods were put to a side, or merely mentioned. As a matter of fact most of the works about these techniques refer to Agglomerative Clustering when they talk about Hierarchical Clustering. This undervaluation from the scientific community towards Divisive Clustering appears to derive from the high computational [SB13] aspects of the algorithm. As we mentioned before, there are $2^{n-1}-1$ possible divisions in each step, which can be computationally demanding [CNDN] if they are all considered.

4.1 DIANA

In [MSWDM64] the authors assume that is possible the creation of a method that does not take into account all the potential divisions of a cluster. The original DIANA algorithm [KR90] is based on that proposal. To proceed with this study we need to understand how the cluster process goes.

The goal is to split a cluster C into C' and C" such that the two resulting clusters have the highest dissimilarity value between them, among all the potential values of cluster pairs.

In order to do that, first we need to find the object in C that is the most dissimilar to all the other objects in C. For that, the algorithm calculates the average dissimilarity (Equation 4.1) value of all objects and chooses the one with the highest value. The identified object is removed and starts a new cluster C'. Now, we can test which objects agree more with C' than with C. For each element remaining in the initial cluster two values are calculated: the Average Dissimilarity with the elements in C and the Average Dissimilarity with the elements in C'. Comparing the results, we can decide if an object changes cluster. Once we can not make anymore changes the iteration stops. The new subclusters are C' and C"(the elements that remained in C) [Din18] [CNDN].

$$AverageDissimilarity(A) = \frac{\sum_{i=B}^{Z} d(A,B)}{N-1} \tag{4.1}$$

B - first element to calculate the dissimilarity to.

Z - last element to calculate the dissimilarity to.

N - Number of elements in the cluster.

To illustrate this algorithm we are going to cluster the data presented in the matrix of dissimilarities 4.1.

Table 4.1: Matrix of dissimilarities

	A	**B**	**C**	**D**	**E**	**F**
A	0	16	47	72	77	79
B	16	0	37	57	65	66
C	47	37	0	40	30	35
D	72	57	40	0	31	23
E	77	65	30	31	0	10
F	79	66	35	23	10	0

First, we need to calculate the Average Dissimilarity for all elements of the cluster using the equation 4.1.

Table 4.2: First DIANA iteration using the values from Table 4.1

Element	**Average Dissimilarity (AD)**
A	(16+47+72+77+79) / 5 = 58.2
B	(16+37+57+65+66) / 5 = 48.2
C	(47+37+40+30+35) / 5 = 37.8
D	(72+57+40+31+23) / 5 = 44.6
E	(77+65+30+31+10) / 5 = 42.6
F	(79+66+35+23+10) / 5 = 42.6

Basically it is the sum of all the values in a line divided by the number of elements minus one. Using the values in the first matrix of dissimilarity we obtain the Table 4.2.

As we can observe, the element that disagrees more (with the higher Average Dissimilarity) is the element A, so it is chosen to start the splinter group. At the moment we have cluster C with elements <B,C,D,E,F> and subcluster C' with <A>. For the second iteration we compute the average dissimilarity of each element in cluster C with the remaining objects and compare it to the average dissimilarity with the objects in C'(both using the values of Table 4.1). The results are shown in Table 4.3.

Table 4.3: Second DIANA iteration using the values from Table 4.1

Element	**AD with elements in C**	**AD with elements in C'**	**Difference**
B	(37+57+65+66) / 4 = 56.25	16	40.25
C	(37+40+30+35) / 4 = 35.5	47	-11.5
D	(57+40+31+23) / 4 = 37.75	72	-34.25
E	(65+30+31+10) / 4 = 34	77	-43
F	(66+35+23+10) / 4 = 33.5	79	-45.5

The element in which the difference is larger will move from C to C'. By analyzing the results, it is clear that the value of B lies much further from the remaining elements than the splinter group. Thus, element B moves to the splinter group, so the new cluster is <A,B> and the initial one <C,D,E,F>. Now we repeat this step until no element can be moved.

After the third iteration (Table 4.4) we can see that all the differences have become negative, which means the remaining elements have a higher similarity within each other than with the subcluster created. Therefore, no further iterations are made, we have completed our first divisive step which resulted into the clusters <A,B> and <C,D,E,F>.

Table 4.4: Third DIANA iteration using values from 4.1

Element	**AD with elements in C**	**AD with elements in C'**	**Difference**
C	(40+30+35) / 3 = 35	(47+37) / 2 = 42	-7
D	(40+31+23) / 3 = 31.3	(72+57) / 2 = 64.5	-33.2
E	(30+31+10) / 3 = 23.7	(77+65) / 2 = 71	-47.3
F	(35+23+10) / 3 = 22.7	(79+66) / 2 = 72.5	-49.8

Now that the initial cluster was divided we need to apply the same rules to each subcluster created. To decide the one to split we must compare their diameter (largest dissimilarity value between two of its objects [MSWDM64]). The diameter of <A,B> is 16, and for <C,D,E,F> is 40. Therefore, we will continue with cluster <C,D,E,F>, and the respective dissimilarity matrix (Table 4.5).

Table 4.5: Matrix of dissimilarities 2

	C	**D**	**E**	**F**
C	0	40	30	35
D	40	0	31	23
E	30	31	0	10
F	35	23	10	0

Once again, we need to find the element to start the splinter group, the one with the highest average dissimilarity. By analyzing the values in Table 4.6 we choose element C as the new cluster starter.

Table 4.6: Second cluster first iteration

Element	**Average Dissimilarity**
C	(40+30+35) / 3 = 35
D	(23+31+23) / 3 = 25.7
E	(30+31+10) / 3 = 23.7
F	(35+23+10) / 3 = 22.7

Afterwards,we calculate the two AD values. Analyzing Table 4.7 we verify that all the differences are negative, consequently the splitting stops. We end up with subclusters <C> and <D, E, F>.

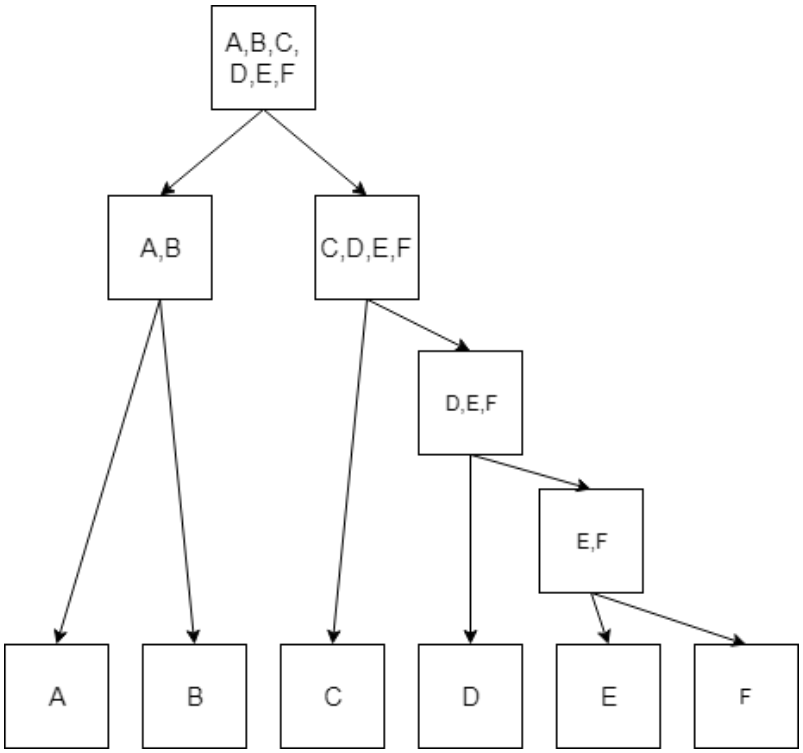

Figure 4.1: Final Tree from the DIANA algorithm

By this time, we have the clusters <A,B>, <C> and <D,E,F>. <C> contains only one element, so it can not be divided anymore, we must choose between <A, B> and <D, E, F>. Once more, we pick the group with the highest diameter and repeat the process all over again until all our clusters are all singletons.

Table 4.7: Second cluster second iteration

Element	AD with elements in C	AD with elements in C'	Difference
D	(31+23) / 2 = 27	40	-13
E	(31+10) / 2 = 20.5	30	-9.5
F	(23+10) / 2 = 16.5	35	-18.5

We are not going to illustrate the rest of the method since the calculus and the procedures are the same as above. The resulting tree is shown in Figure 4.1.

4.2 Modified DIANA

This Modified DIANA technique was proposed by INESC-TEC. In their work, among other features, they implemented a multi-classifier evaluation for each level and semi-supervised learning. Since our work is a continuation from theirs, we can abstract ourselves from the implementation

of those features. For now we will just focus on explaining how the algorithm differs from the basic one.

4.3 Modifications

The main modifications that we are going address for now are the multi-classifier level evaluation and the semi-supervised learning incorporation in the classifiers. We are also going to make a brief description on how each classifier works.

4.3.1 Dissimilarity Matrices

The basic DIANA algorithm dissimilarity matrix is computed using a specific Metric like the ones explained in Section 2.4.3.2. Applying that metric to the confusion matrices, three dissimilarity matrices are calculated instead of one. Each confusion matrix comes from the application of one of the three algorithms: Naive Bayes, Decision Trees and K-Nearest Neighbor [Din18].

4.3.2 Semi-supervised Learning

In the Human Activity Recognition study, the classified data is highly dynamic. In terms of value analysis, this means that a step for a tall person can be completely different than one from a smaller person. This is were semi-supervised learning appears.

This method takes in consideration previously correctly classified data from the user (labeled data), to improve the classification accuracy for that user next predictions (unlabeled data). This helps in the classification of activities from specific users and may even help in future ones that did not existed at the start [Whab].

4.3.3 Classifiers

As we stated above, we can abstract ourselves from the implementation of the algorithms, as they are already implemented in the technique provided by INESC-TEC. However, they still need to be analyzed in order to have some background information of what is happening.

4.3.3.1 Naive Bayes

Naive Bayes is a probabilistic classifier between two features that tries to classify a new class, based on the currently existing ones ([Nai]). Basically, it calculates the probability of one class given a prediction ([6Py]).

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)} \tag{4.2}$$

Table 4.8: Confusion Matrix

	A	**B**	**C**
A	94	16	10
B	0	113	16
C	4	4	92

4.3.3.2 K-Nearest Neighbor

In this algorithm, an object is classified considering the classes of the K nearest neighbors we wish to take vote from [Int]. The new element is classified as the class with more number of elements in the selected neighbors.

4.3.3.3 Decision Trees

This algorithm is a decision support tool in shape of a tree, where each internal node represents a "question" and each branch one "answer". The available decisions generate a map of possible outcomes considering a set of choices, providing a way to weigh the consequences of actions [Whaa].

4.4 Technique

Although this is a modified technique, the process of the algorithm does not really change much from the one explained in Section 4.1. Once we have computed the dissimilarity matrices, the splitting procedure is the same, but instead of choosing the highest Average Dissimilarity value from one matrix, we choose the highest one among the tree.

However, the process to originate those matrices is different. Using each classifier's confusion matrix we need to calculate the dissimilarity between each element to create the dissimilarity matrix. For that we are going to use a custom measure [Din18].

By analyzing the confusion matrix in Table 4.8, where in each cell the column represents the predicted class and the row the actual class (or vice versa) [Ste97], we can calculate the dissimilarity between pairs of elements comparing the number of correct instances and the total instances [Din18].

$$d(A,B) = \frac{94+113}{94+113+16} = 0.93 \tag{4.3}$$

In the Equation 4.3 we calculate the dissimilarity between A and B by dividing the number of correct guess in the algorithm by all the guesses. The result can be any number between 0 and 1, being 1 the perfect value, two perfectly distinguished activities, and 0, two activities that can not be distinguished at all.

After applying this measure to all the possible pairs of elements, we originate the dissimilarity matrix. Once all the three matrices are set, the DIANA algorithm resumes as explained before.

4.5 Dataset

As previously mentioned, we are not exporting our method to an Android device, neither gathering data from the *smartphone* sensors. So, we are going to use the PAMAP2 dataset.

PAMAP2 was created to tackle the lack of a standard and commonly used dataset for physical activity monitoring, as a result, it is freely available for research and academic purposes [RS12b]. All the benchmarks and discussion of results were also already analyzed and presented in [RS12a] (with good results), so we can abstract ourselves from this.

This dataset contains data from 18 different physical activities, conducted by 9 different users, using 3 IMUs (over the wrist on the dominant arm, on the chest and on the dominant side's ankle) and a heart rate monitor. The files contain 54 columns, one for the timestamp, 52 of raw sensory data and one with the activity label (2.2.1).

Each user had to follow a protocol to practice 12 of those 18 activities resulting in a different file for each person. As a result of this selection, not all of the activities have correct outputs in each user's file. So, our program discards those that have zero correct outputs since the number of instances is to low to rely on the results.

The 18 activities are: lying, sitting, standing, walking, running, cycling, nordic walking, watching TV, computer work, car driving, ascending stairs, descending stairs, vaccuum cleaning, ironing, folding laundry, house cleaning, playing soccer and rope jumping.

Chapter 5

Proposed Solution

Now that we already understood the basis of our work we can demonstrate the implementations that were made in order to improve the method explained above.

5.1 The problem

Since the entire DIANA algorithm explained above works with binary divisions, in the event of a high possible starting activities number, the resulting tree can be very lengthy. Also, our top-down classification approach does not allow for an instance to return to a node's parent once it has been assigned to that node. This, together with the activities complexity, user subjectivity and sensor data reliability can cause a high number of incorrect classifications.

5.1.1 Propagation of Errors

As we just explained, the classification process is irreversible, therefore, once an instance is misclassified to a non-leaf node, all the decisions made from there will be wrong.

Although the most divergent clusters and easy to classify are situated on the top level of the tree, the low-level decisions are highly dependent on the upper ones [BDP+]. An instance that needs to go through seven classifications before reaching a leaf will have a higher error probability than one that needs to be classified fewer times.

5.2 The Approach

At the end of the execution of the DIANA algorithm, we have two trees. A flat multi-classification one, and the one generated by the algorithm. Considering that for each tree we have the confusion matrices of the leaf nodes, we can easily prune a tree, compare the values (using the measures presented in 2.7) and decide if it is worth to accept that cut or not.

So, our job was to continuously prune the DIANA algorithm generated tree until we reach the root node or a cut produces worst values.

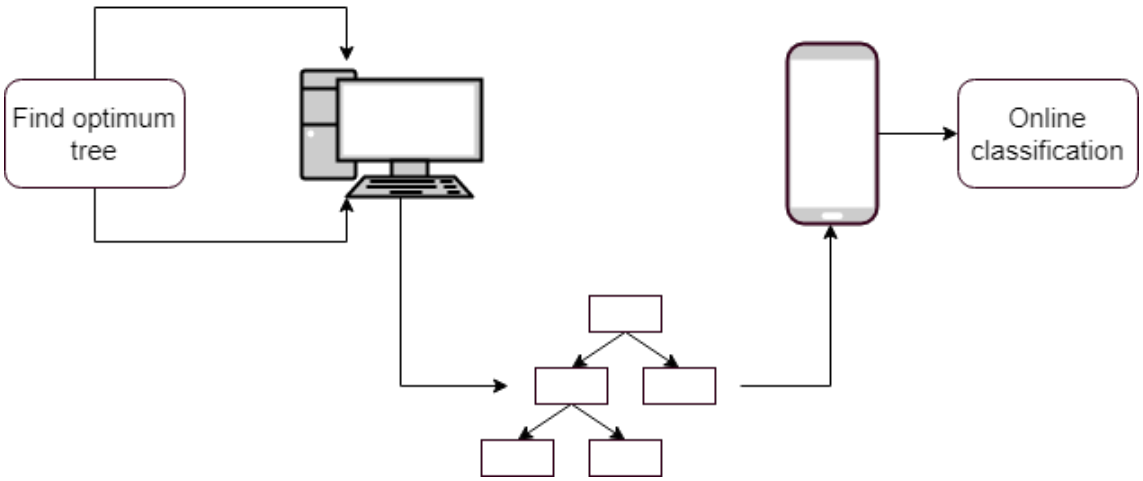

Figure 5.1: Workflow

Our goal is that in the future, the optimum tree can be used in an online classifier, on other words, provide a real-time classification on the activity the user is performing by the means of a mobile phone (Figure 5.1).

5.2.1 Pruning Methods

For the pruning process, we did not choose any of the methods explained in Section 2.5.2, however, the decision to prune or not was adopted from the one explained in Section 2.5.3.3. We opted to go this way because we wanted to test all the possible cuts and be one hundred percent sure a better tree was being skipped. Even if this approach has a higher computational cost, it can prove to be advantageous during the on-the-fly classification.

In the figures, we demonstrate the two implemented alternatives to prune the tree.

The first case, Figure 5.2, verifies if there is any subtree where one of the root children is a leaf and the other is a non-leaf. For the non-leaf node, all its children must be leaves.

On the other hand, Figure 5.4, verifies if there is any subtree where all the root children are non-leaf nodes but all their children are leaves.

As we can see from analyzing Figure 5.3 and Figure 5.5, the result is always the original root node divided into multiple children (the existent leaves in the original subtree). The in-between node is removed.

We need to keep in mind that over the course of our algorithm, subtrees with more than two children can be created. So, our implementations need to be scalable in order to detect all the possible pruning situations.

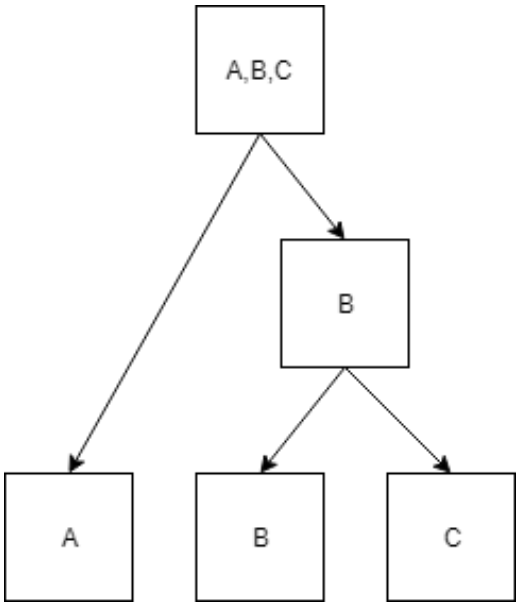

Figure 5.2: Example of subtree

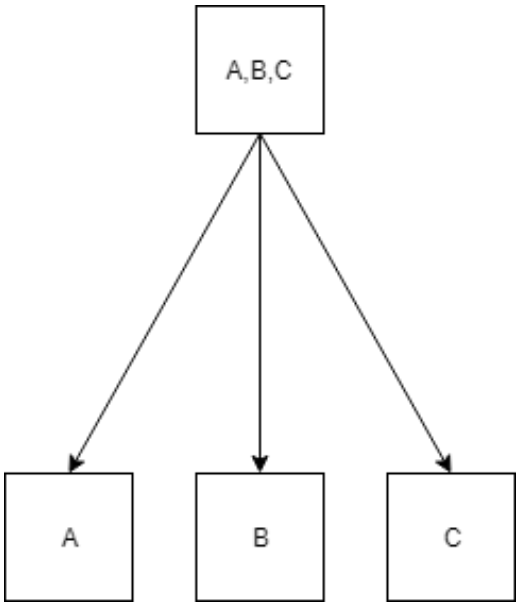

Figure 5.3: Correspondent pruned subtree

For example, in situations similar to the one in Figure 5.2, there can be two, or more, non-leaf root children (with all their children as leaves) and two, or more, leaf root children. Our method must recognize that this is a valid pruning situation.

5.3 Architecture

To better understand the entire workflow of our project we can analyze Figure 5.6.

Firstly, we must input our starting variables (window size, sensors, classifiers, among others), which will greatly impact our final results and the DIANA generated tree. Then, we must train our model with a selection of our labeled instances using the input variables. Thirdly, it is necessary to test our model in order to create the confusion matrices, and consequently the tree, for the execution of our algorithm.

Once the algorithm has finished running for the first time and we have the confusion matrices of the leaf nodes, we can start our pruning process.

If it is possible to prune (following the rules described in Section 5.2.1), the cut is simulated, the model is tested again but with the new tree and the algorithm is carried out again. In case of better results than the previous run, the cut is accepted and the results are saved. If the results are worst, the cut is discarded and the pruning process is repeated (if possible).

As soon as it is not possible to make another pruning iteration, the process is terminated and all the results are compared. The results analysis and comparison is a very important part, as it is what will validate or not our studies. It will be more carefully explained in Chapter 5.

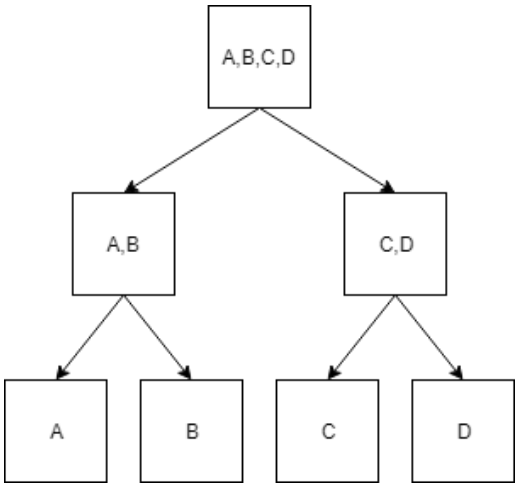

Figure 5.4: Example of subtree

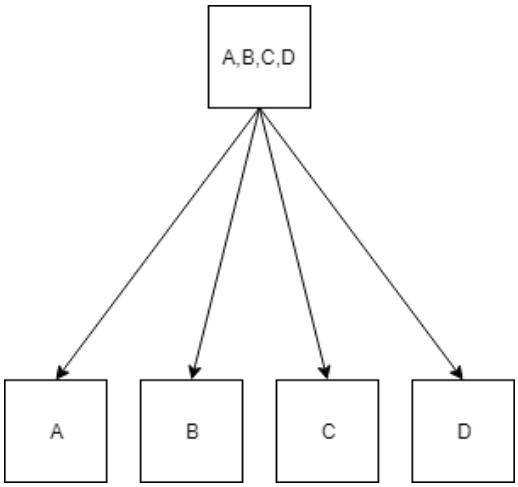

Figure 5.5: Correspondent pruned subtree

5.4 Run Configurations

Based on what was explained above, some variables need to be set before running the project.

- **Window Size** — As explained in Section 2.8 the window size is what determines how the sensors data is segmented.
- **Overlap Factor** — Since we are using the Overlapping Sliding Windows method (Section 2.8.2) we need to define the overlap percentage that will be carried to the next window.
- **Classifiers** — The modified DIANA technique used three classifiers instead of one, like in the basic algorithm. Since the classifiers to be used are passed in the input variables and we are always working on top of the modified algorithm, all the classifiers are chosen.

  To choose the classifiers we must pass a 0|1 string of three elements: kNN, Naive Bayes, Hoeffding Tree.
- **Sensors** — As mentioned, our dataset has data retrieved by three different sensors. For testing purposes the user can choose the sensors he wants to be enabled.

  Like the classifiers it must be a 0|1 string of three elements: hand, chest, ankle.
- **Testing User** — In addition, our dataset has information of nine different users. In order to have results within the same testing environment, only one user was selected in each experiment.

5.5 Tools

During the development process of our work, these were the used tools:

5.5.1 Eclipse

Since both the MOA API and the previously developed DIANA Algorithm were prepared for a JAVA environment, Eclipse was the choosen IDE as it support JAVA development as well as the required Ivy dependency management.

5.5.2 Massive Online Analysis

Massive Online Analysis is a free open-source software directed to the handling of data streams. It allows experiments on online learning, supports several classifiers like Naive Bayes and Decisions Trees, and is written in Java. Some of these features are exactly problems we need to address in our work.

5.5.3 JGrapht

Jgrapht is a JAVA open source graph drawing software, which facilitates graphs visualization by generating PNG images from data. Since our work requires an extensive graph analysis in order to visualize and confirm the pruning iterations this framework was introduced.

Previously the algorithm had to be run in Debug mode and the values had to be written down manually every time there was a change. Currently, whenever a change is made, a new image is generated with the new graph.

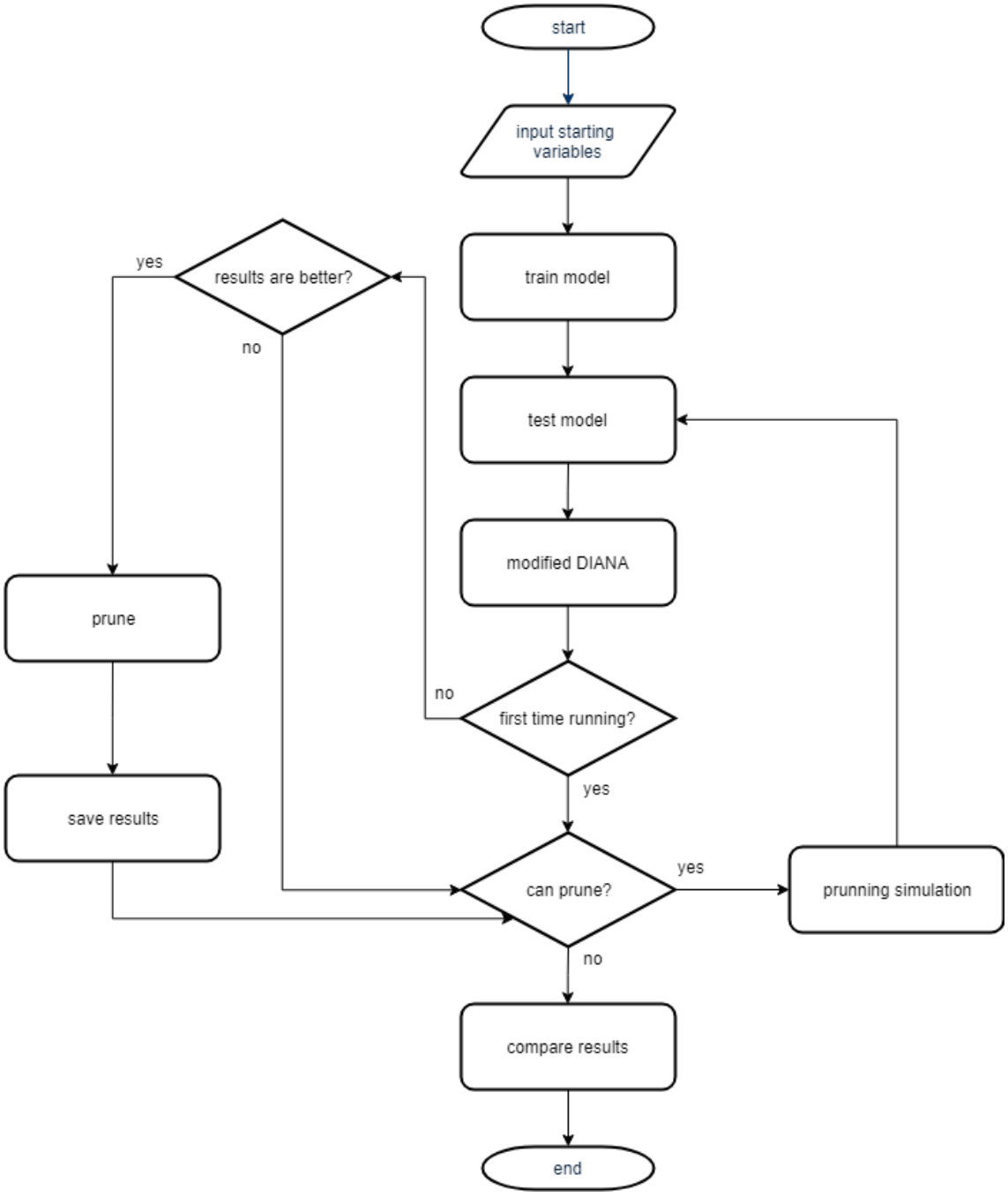

Figure 5.6: Project Architecture

Chapter 6

Analysis of Results

6.1 Setup

As seen in Section 5.4 we need to define some variables before running our work. The window size and the overlap factor are the most complex ones, since small changes in these values can produce different or unreliable results. However, since our study is based on the algorithm described in [Din18], which uses the same input variables, we are going to take advantage of the information already retrieved by them. First, 200 HZ and 70 % overlap were chosen on their experiments, nevertheless, after some testing 400 HZ turned out to be the most suited value for the window size. As a result, 400 HZ and 70 % overlap were the values we used.

Regarding the enabled classifiers and sensors we wanted to test our modifications against the best possible scenario. On one hand, situations like having only the chest sensors activated would generate erroneous values for activities like walking, standing and sitting, since these rely more on the ankle movement. On the other hand, with only one classifier there was no guarantee if it was the best possible accuracy. With this in mind, all sensors and classifiers were active in our experiments.

6.2 Experiments

Using the setup explained above, our pruning process was applied on seven of the nine users. User six was not taken into consideration because it was the one that we trained our model with, since its data contained almost all possible activities. Given that user nine only performed rope jumping it was also discarded as it was not going to provide relevant conclusions to our work.

We are not going to compare all the results for the seven experiments since some of them generated similar outputs and the procedures were not different. However, there were some interesting ones that we need to address. All the presented trees from now on are the PNG images created by the JgraphT framework mentioned before.

6.2.1 User 2

First, we are going to experiment the work we developed with the data from User2. At the end of the modified DIANA, the generated structure is the one on Figure 6.1, with an accuracy of 46,8 %. Before having results to compare with this we need to look for pruning opportunities in the tree.

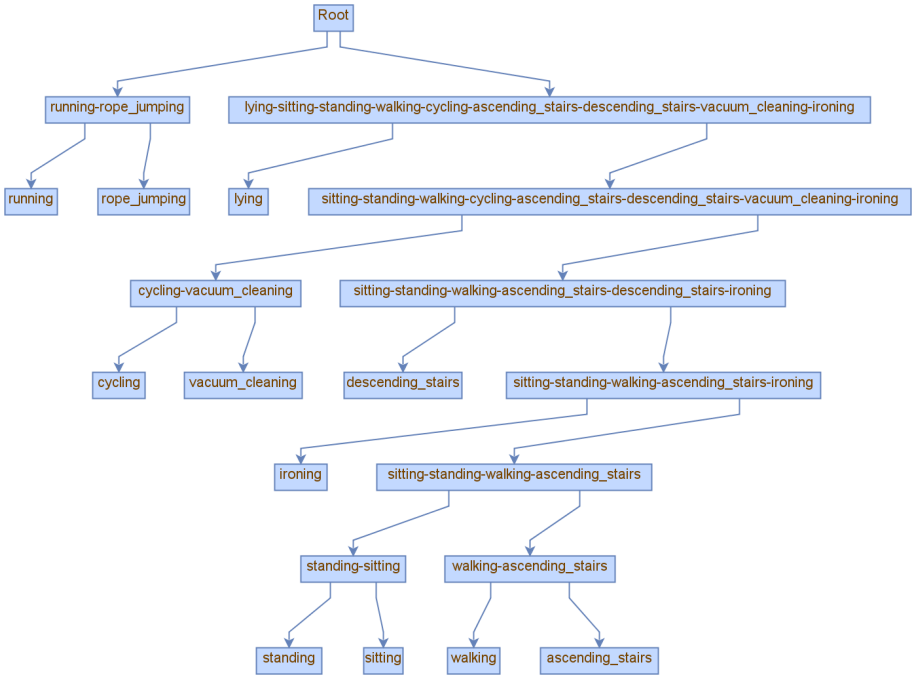

Figure 6.1: User 2 DIANA generated tree

Analyzing the nodes relations using the implemented methods, we can see that the <sitting-standing-walking-descending_stairs> binary divisive subtree can be replaced by a multi-class one. Both <standing-sitting> and <walking-ascending_stars> only have leaf children, which means we can remove them and create direct relations between the root node of this subtree and the four children: <standing>, <sitting>, <walking> and <ascending_stairs>. Figure 6.2 illustrates this cut, and the model is tested with the new tree.

Table 6.1: User 2 results comparison

	DIANA	**First Pruning Iteration**	**Flat classification**
Accuracy	46,8%	52%	58,9%

Looking at both results, the new structure improves the global accuracy of the original algorithm, which means that our pruning method worked. This being said, we can search for cuts in the new tree.

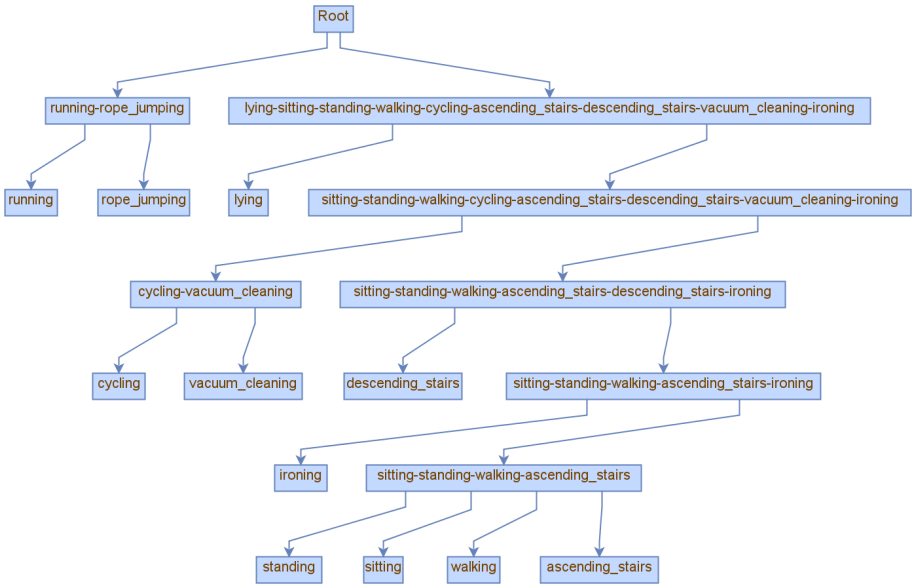

Figure 6.2: User 2 tree after first pruning iteration

Following the same methods, the node <sitting-standing-walking-ascending_stairs-ironing> is our only possible cut, since <ironing> is already a leaf node and <sitting-standing-walking-ascending_stairs> only has leaf children. The former is removed and it's children go up one level, resulting in the tree on Figure 6.3.

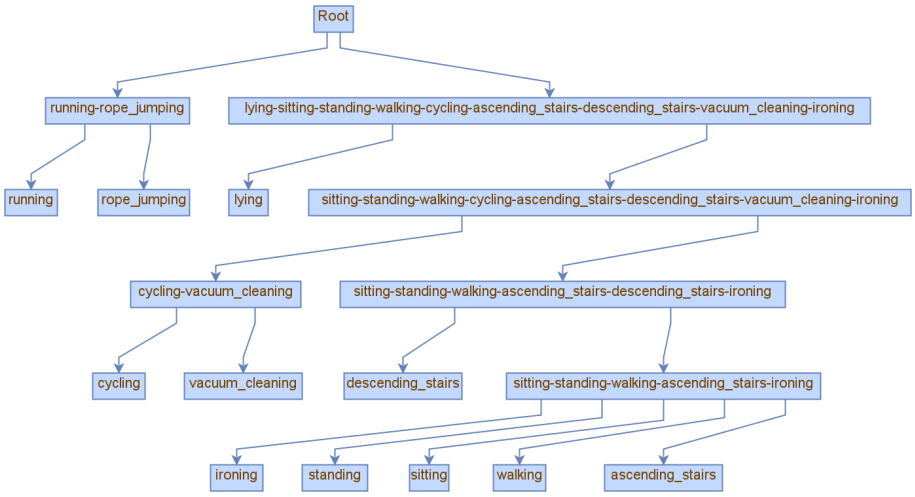

Figure 6.3: User 2 tree after second pruning iteration

Testing the model on the new tree we get a resulting accuracy of 51,5 %. This value is less than the previously obtained accuracy (Table 6.2), so, the cut is denied, and the subtree <sitting-standing-walking-ascending_stairs-ironing> is rolled back to the former state. As it is, there are no more pruning possibilities, our Post-Pruning method is therefore terminated and the optimum tree is the one on Figure 6.2 with an accuracy of 52 %.

Table 6.2: User 2 results comparison

	DIANA	First Pruning Iteration	Second pruning Iteration	Flat classification
Accuracy	46,8%	52%	51,5%	58,9%

6.2.2 User 3

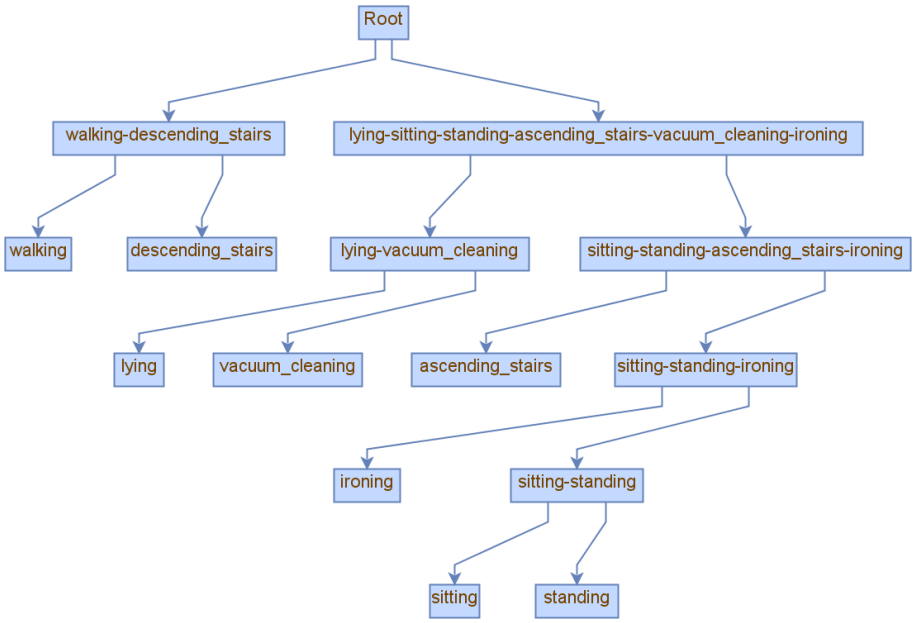

Figure 6.4: User 3 DIANA generated tree

For the second experiment, we used User 3, which at the end of DIANA has an accuracy of 72,8 % (represented on Figure 6.4). Examining the relations we can see that the <sitting-standing> subtree can be replaced by it's children, <sitting> and <standing>. Doing this cut, the node <sitting-standing-ironing> becomes multi-class divided with <ironing>, <sitting> and <standing> as children. Our first pruning iteration is terminated. Using the new tree (Figure 6.5), it is time to test the data and compare the accuracy.

Comparing the results in Table 6.3 it is clear that the pruning process did not increased the final accuracy. Therefore, the cut is denied, the <sitting-standing-ironing> subtree is rolled back to the initial state and we come back to the initial tree.

Again, we need to find a node that can be pruned following the implemented methods. However, the one used above, since it was already tested, can not be considered. As so, by analyzing the tree, there are no more possible cuts in our structure. The Post-Pruning process is concluded, with no advantages compared to the implemented DIANA.

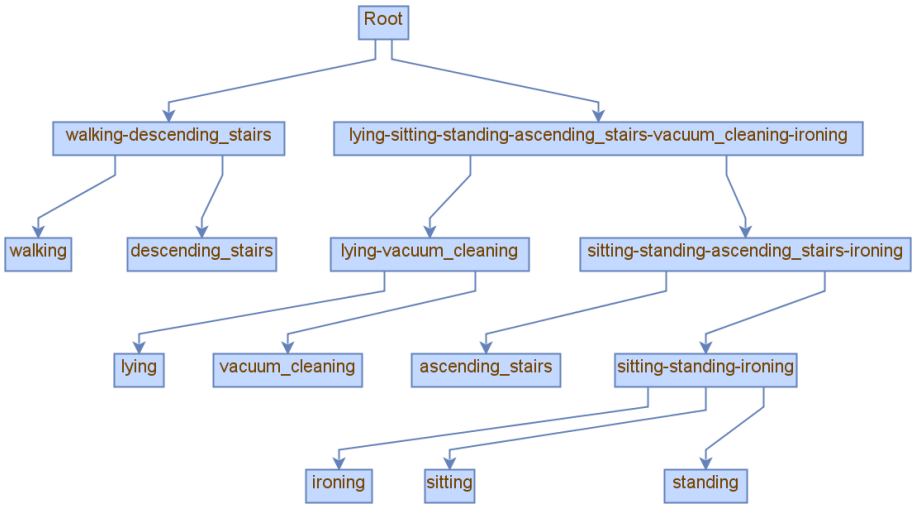

Figure 6.5: User 3 tree after one pruning iteration

6.2.3 Other experiments

For all the remaining users the same test was done, comparing results before and after pruning, just like those explained above. Users 2 and 3 worked as an example for the possible outcomes. On one hand the results are better after the first and second pruning iteration, and on the other hand the accuracy never improved. In the Table 6.4 is presented a brief summary of all the results:

6.3 Analysis

Looking at Table 6.4 there is a clear relation between the number of levels and the Post-Pruning result. While Users 3 and 4 hierarchical structures have 6 levels, the rest of the users have 7 or more. The difference from 6 to 7 may look minimal, however for activities like standing and sitting (whose number of instances classified as such is significant) one more classification procedure can be the difference from a correct recognition or not. Also, for instances that are badly classified at the top levels, higher the tree, higher the propagation of errors. That is why Users 3 and 4 have the best accuracy after the DIANA execution, and therefore, the most difficult to improve.

Nodes dissimilarity is another key factor that can influence the result. The DIANA algorithm is based on the dissimilarities between activities, and therefore, so is the the first tree that we are working on. In some cases, the divisions made by DIANA may not be the most appropriate ones due to badly selected features. Thus, making it extremely hard to provide relevant improvements with our implementations. Also, for some users, the data collected may not be good enough to have a clear distinction between activities. The most similar activities are only split on the final nodes (deeper in the tree), which can reduce the global accuracy, like in User 2. This is directly related to the problem stated above.

There are some other factors that can impact the obtained results. These are related to the dataset and are out our reach, like the number of users used and the sensors. On one hand, the

Table 6.3: User 3 results comparison

	DIANA	**Pruned DIANA**	**Flat classification**
Accuracy	72,8 %	68,5 %	88,3%

sensors need to be perfectly calibrated in order to acquire the most accurate data. On the other hand, nine users (reduced to seven in our case) is a very low number for testing purposes. For HAR works the number of testing users needs to be higher, as each person has a different physiology and different ways of practicing exercise.

Table 6.4: Summary of results

	User 1	**User 2**	**User 3**	**User 4**	**User 5**	**User 7**	**User 8**
Number of Levels	9	8	6	6	8	7	7
Accepted Pruning Iterations	0	1	0	0	2	3	1
DIANA	76,3%	46,8%	72,8%	75,1%	71,7%	59,7%	68,3%
First Pruning Iteration	71,4%	52%	68,5%	73,5%	73,6%	63,3%	72%
Second Pruning Iteration	-	51,5%	-	-	76,4%	67,6%	69,9%
Third Pruning Iteration	-	-	-	-	73,4%	71,3%	-
Fourth Pruning Iteration	-	-	-	-	-	71%	-
Flat Classification	70,1%	58,9%	88,3%	84,6%	84,4%	90,1%	61,3%

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The objective of the work was to study how well turning binary-divisions into multi-class ones could improve the classification of a hierarchical divisive method. In order to do that, Post-Pruning was the approach we chose. There is a huge amount of pruning techniques, however we presented a simple solution implemented by us that considered all the pruning possibilities.

Despite not having consistent results, our approach presented some interesting ones. The trees with the most number of leaves are the likeliest to have the global accuracy increased as we keep pruning and therefore reduce the height. Also, the flat classification structure that we compared to our values, still proves to be superior related to the hierarchical options.

7.2 Future Work

From the study already made, several ideas come to mind when we think about future work. First, comparisons can be made using the Post-Pruning methods presented in Section 2.5.2 regarding computational costs, algorithm run time and accuracy. With these values, conclusions can be deduced about the effectiveness of the different approaches.

Another possible implementation is related to the dataset. At the moment, our work can only be applied to PAMAP2. It would be interesting to modify it and see how well it can run with other different data and activities.

Following this idea, the next step could be recognizing activity transitions like sitting to walking or walking to running, and vice versa. The advantage is that once these transitions are identified, the related window could be removed from the training phase. During a transition, the data is inconsistent and could lead to erroneous decisions.

As an alternative to our bottom-up pruning approach, a top-down pruning strategy could be applied. This could provide results not only on the hierarchical structure generated, but also on the comparison of different pruning methods.

Finally, the most relevant improvement is to export the resulting tree into an Android App and recognize physical activities promptly. Data will have to be retrieved from the Android Sensors, handled and then passed onto the classification procedure. Once this is done, there is plenty of information that can be analyzed, from medical advice to target advertising.

Human Activity Recognition is a vast and subjective field, and obviously, there is huge room for improvement. From better sensors, different algorithms and distinct data to advances in technology, this is clearly an area that will always be of great enhancement opportunity.

References

[20009] Distances between Clustering, Hierarchical Clustering. Technical report, 2009.

[6Py] 6 Easy Steps to Learn Naive Bayes Algorithm (with code in Python).

[AB10] Kerem Altun and Billur Barshan. Human Activity Recognition Using Inertial/-Magnetic Sensor Units. pages 38–51. Springer, Berlin, Heidelberg, 2010.

[Acc] Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures - Exsilio Blog.

[BA] Leonard A Breslow and David W Aha. Simplifying Decision Trees: A Survey 1. Technical report.

[BDP+] Oresti Banos, Miguel Damas, Hector Pomares, Fernando Rojas, Blanca Delgado-Marquez, and Olga Valenzuela. Human activity recognition based on a sensor weighting hierarchical classifier.

[BI] Ling Bao and Stephen S Intille. Activity Recognition from User-Annotated Acceleration Data. Technical report.

[Bra] Max Bramer. Pre-pruning Classification Trees to Reduce Overfitting in Noisy Domains. Technical report.

[CAL10] Jingyuan Cheng, Oliver Amft, and Paul Lukowicz. Active Capacitive Sensing: Exploring a New Wearable Sensing Modality for Activity Recognition. pages 319–336. Springer, Berlin, Heidelberg, 2010.

[Car16] Hugo Louro Cardoso. Predicting Activities from Smartphones. Technical report, 2016.

[CB] Bojan Cestnik and Ivan Bratko. On estimating probabilities in tree pruning. In *Machine Learning — EWSL-91*, pages 138–150. Springer-Verlag, Berlin/Heidelberg.

[CNDN] Rodrigo C Camargos, Paulo R Nietto, Maria Do, and Carmo Nicoletti. Agglomerative and Divisive Approaches to Unsupervised Learning in Gestalt Clusters. Technical report.

[Con] Confusion Matrix in Machine Learning - GeeksforGeeks.

[CYL+08] Yen-Ping Chen, Jhun-Ying Yang, Shun-Nan Liou, Gwo-Yun Lee, and Jeen-Shing Wang. Online classifier construction algorithm for human activity detection using a tri-axial accelerometer. *Applied Mathematics and Computation*, 205(2):849–860, 11 2008.

REFERENCES

[Der05] Konstantinos G Derpanis. Mean Shift Clustering. Technical report, 2005.

[Dif] Difference Between Supervised, Unsupervised, & Reinforcement Learning | NVIDIA Blog.

[Din18] Joel Alexandre Ezequiel Dinis. Hierarchical Classification using hierarchical clustering: an application to Human Activity Recognition. 8 2018.

[Dun73] J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics*, 3(3):32–57, 1 1973.

[ECS65] A. W. F. Edwards and L. L. Cavalli-Sforza. A Method for Cluster Analysis. *Biometrics*, 21(2):362, 6 1965.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Technical report, 1996.

[EMS] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. Decision Tree Pruning as a Search in the State Space. Technical report.

[Env] Environment sensors | Android Developers.

[FH75] Keinosuke Fukunaga and Larry D Hostetler. The Estimation of the Gradient of a Density Function, with Applications in Pattern-Recognition. Technical Report 1, 1975.

[Fra00] Eibe Frank. Pruning Decision Trees and Lists. Technical report, 2000.

[GHS12] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Recognizing activities with cluster-trees of tracklets. 13:10, 2012.

[GM12] Anshul Goyal and Rajni Mehta. Performance Comparison of Naïve Bayes and J48 Classification Algorithms. Technical Report 11, 2012.

[Hea] Health Risks of an Inactive Lifestyle.

[Hie] Hierarchical Clustering | solver.

[HJ] Zhenyu He and Lianwen Jin. *Activity Recognition from acceleration data Based on Discrete Consine Transform and SVM*.

[HJ08] Zhen-Yu He and Lian-Wen Jin. *Activity Recognition From Acceleration Data Using AR Model Representation and SVM*. 2008.

[HNK09] Yuya Hanai, Jun Nishimura, and Tadahiro Kuroda. Haar-Like Filtering for Human Activity Recognition Using 3D Accelerometer. In *2009 IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop*, pages 675–678. IEEE, 1 2009.

[How] How technology and inactive lifestyles are changing our children | Guardian Sustainable Business | The Guardian.

[Int] Introduction to KNN, K-Nearest Neighbors : Simplified.

REFERENCES

[Jap06] Nathalie Japkowicz. Why Question Machine Learning Evaluation Methods? (An illustrative review of the shortcomings of current methods). Technical report, 2006.

[JGK+08] Luciana C. Jatoba, Ulrich Grossmann, Chistophe Kunze, Jorg Ottenbacher, and Wilhelm Stork. Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2008, pages 5250–5253. IEEE, 8 2008.

[KR90] Leonard Kaufman and Peter J. Rousseeuw, editors. *Finding Groups in Data*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA, 3 1990.

[KVKEK11] B J A Krose, T L M Van Kasteren, G Englebienne, and B J A Kröse. Lecture Notes in Computer Science Hierarchical Activity Recognition Using Automatically Clustered Actions. 2011.

[KYLT10] A M Khan, Young-Koo Lee, S Y Lee, and Tae-Seong Kim. A Triaxial Accelerometer-Based Physical-Activity Recognition via Augmented-Signal Features and a Hierarchical Recognizer. *IEEE Transactions on Information Technology in Biomedicine*, 14(5):1166–1172, 9 2010.

[Lac] Lack of exercise puts 1.4 billion people at risk of disease, WHO warns - Business Insider.

[LPLP12] Óscar D. Lara, Alfredo J. Pérez, Miguel A. Labrador, and José D. Posada. Centinela: A human activity recognition system based on acceleration and vital sign data. *Pervasive and Mobile Computing*, 8(5):717–729, 10 2012.

[LVJV03] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2 2003.

[MHC+18] Kyle Mandsager, Serge Harb, Paul Cremer, Dermot Phelan, Steven E. Nissen, and Wael Jaber. Association of Cardiorespiratory Fitness With Long-term Mortality Among Adults Undergoing Exercise Treadmill Testing. *JAMA Network Open*, 1(6):e183605, 10 2018.

[MM11] David McGlynn and Michael G. Madden. An Ensemble Dynamic Time Warping Classifier with Application to Activity Recognition. In *Research and Development in Intelligent Systems XXVII*, pages 339–352. Springer London, London, 12 2011.

[Mot] Motion sensors | Android Developers.

[MSWDM64] P. MACNAUGHTON-SMITH, W. T. WILLIAMS, M. B. DALE, and L. G. MOCKETT. Dissimilarity Analysis: a new Technique of Hierarchical Subdivision. *Nature*, 202(4936):1034–1035, 6 1964.

[MWA+07] David Minnen, Tracy Westeyn, Daniel Ashbrook, Peter Presti, and Thad Starner. Recognizing Soldier Activities in the Field. pages 236–241. Springer, Berlin, Heidelberg, 2007.

[Nai] Naive Bayes Classifier.

REFERENCES

[NSW15] M. H. M. Noor, Z. Salcic, and K. I-K. Wang. Dynamic sliding window method for physical activity recognition using a single tri-axial accelerometer. In *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, pages 102–107. IEEE, 6 2015.

[PA08] Nam Pham and Tarek Abdelzaher. Robust Dynamic Human Activity Recognition Based on Relative Energy Allocation. In *Distributed Computing in Sensor Systems*, pages 525–530. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[PEK+06] Juha Pärkkä, Miikka Ermes, Panu Korpipää, Jani Mäntyjärvi, Johannes Peltola, and Ilkka Korhonen. Activity Classification Using Realistic Data From Wearable Sensors. *IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE*, 10(1), 2006.

[PG] Shraddha Pandit and Suchita Gupta. A Comparative Study on Distance Measuring Approaches for Clustering.

[Pos] Position sensors | Android Developers.

[PPP12] Nikita Patel, S P B Patel, and Associate Prof. Study of Various Decision Tree Pruning Methods with their Empirical Comparison in WEKA Saurabh Upadhyay. Technical Report 12, 2012.

[Qui87] J.R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 9 1987.

[RA13] Shweta Rajput and Amit Arora. Designing Spam Model-Classification Analysis using Decision Trees. Technical Report 10, 2013.

[RS12a] Attila Reiss and Didier Stricker. Creating and benchmarking a new dataset for physical activity monitoring. In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '12*, page 1, New York, New York, USA, 2012. ACM Press.

[RS12b] Attila Reiss and Didier Stricker. Introducing a New Benchmarked Dataset for Activity Monitoring. In *2012 16th International Symposium on Wearable Computers*, pages 108–109. IEEE, 6 2012.

[Sam59] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, 7 1959.

[SB13] K Sasirekha and P Baby. Agglomerative Hierarchical Clustering Algorithm-A Review. *International Journal of Scientific and Research Publications*, 3(3), 2013.

[Sen] Sensors Overview | Android Developers.

[SF11] Carlos N. Silla and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 1 2011.

[Sim] Simple guide to confusion matrix terminology.

[SLL] Sensors Sensorsoscar, D Lara, and Miguel A Labrador. A Survey on Human Activity Recognition using Wearable SensorsOscar. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 15(3).

REFERENCES

[SM12] T Soni Madhulatha. An Overview O Clustering Methods. 2(4):719–725, 2012.

[Ste97] Stephen V Stehman. & Forestry, 320 Bray Hall, Syrxusr, NY 13210. Kmhwl 1 S L>˜ccwrhrr 1996: rvcised. *OElsrmer Science Inc*, 62:77–89, 1997.

[Sup] Supervised and Unsupervised Machine Learning Algorithms.

[Sze10] Csaba Szepesvári. Algorithms for Reinforcement Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 1 2010.

[Und] Understanding K-means Clustering in Machine Learning. Technical report.

[VLL+11] La The Vinh, Sungyoung Lee, Hung Xuan Le, Hung Quoc Ngo, Hyoung Il Kim, Manhyung Han, and Young-Koo Lee. Semi-Markov conditional random fields for accelerometer-based activity recognition. *Applied Intelligence*, 35(2):226–241, 10 2011.

[WCW+] Aiguo Wang, Guilin Chen, Xi Wu, Li Liu, Ning An, and Chih-Yung Chang. Towards Human Activity Recognition: A Hierarchical Feature Selection Framework.

[Whaa] What is a Decision Tree Diagram | Lucidchart.

[Whab] What is Semi-Supervised Learning?

[ZGH+] Yan Zhou, Oleksandr Grygorash, Thomas F Hain, Meador Inge, and Zach Jorgensen. Clustering with Minimum Spanning Trees. Technical report.

[ZS09] Chun Zhu and Weihua Sheng. Human daily activity recognition in robot-assisted living using multi-sensor fusion. In *2009 IEEE International Conference on Robotics and Automation*, pages 2154–2159. IEEE, 5 2009.

REFERENCES

Appendix A

Results

A.1 User 1 Results

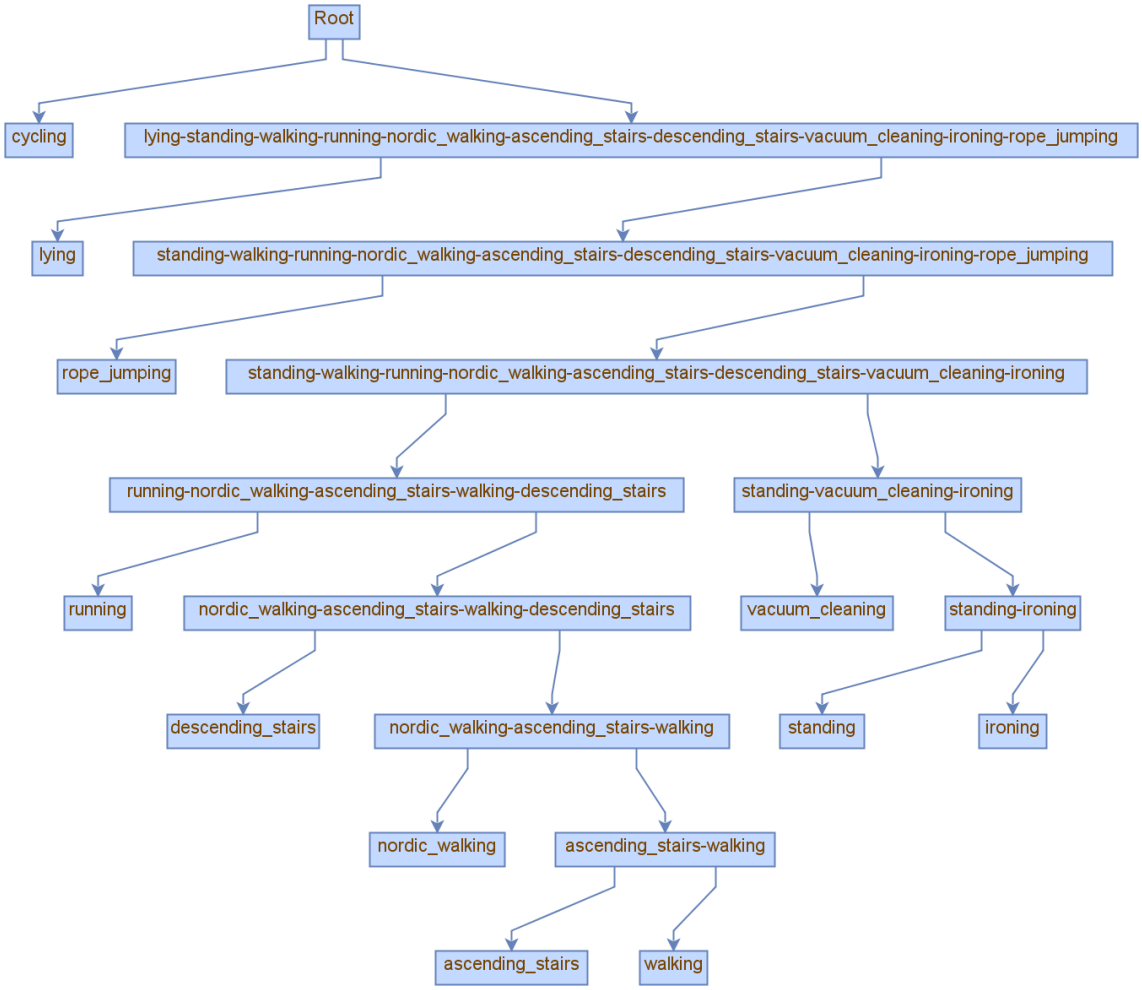

Figure A.1: User 1 initial tree

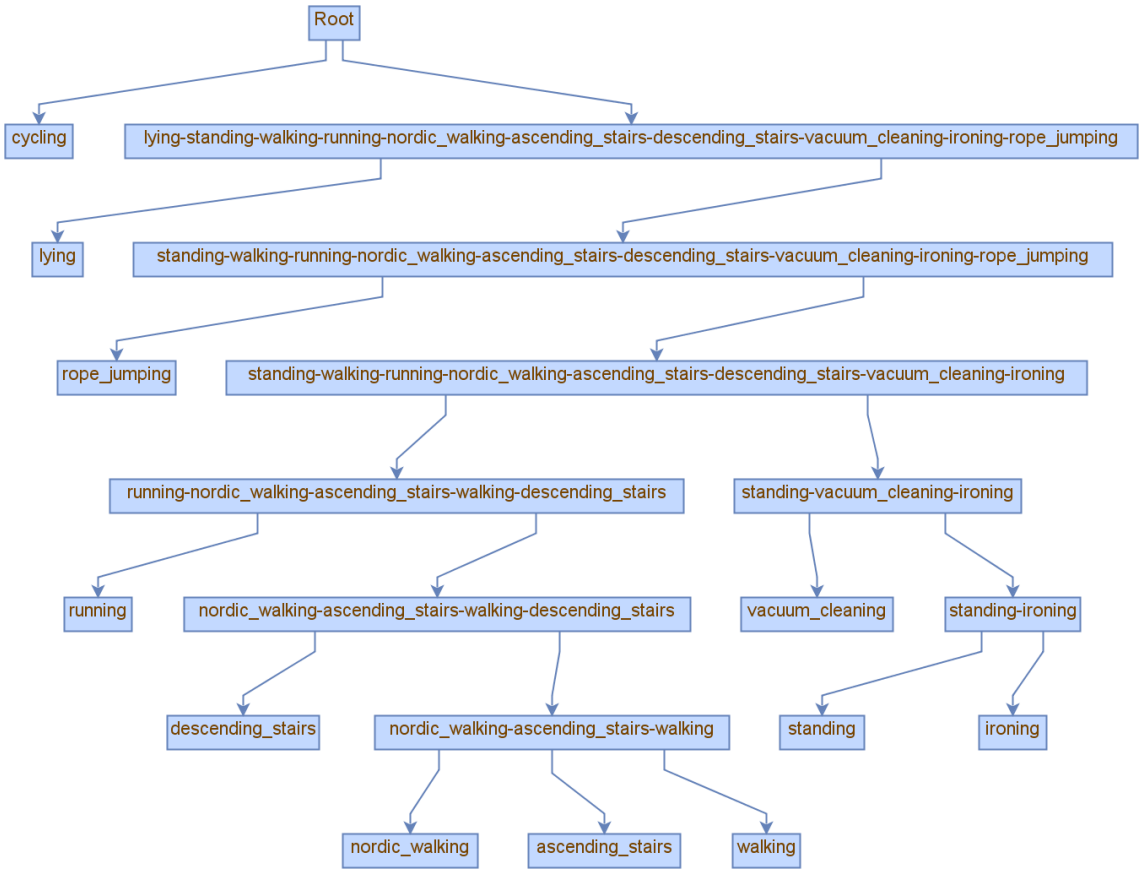

Figure A.2: User 1 tree after first pruning iteration. Pruning was not accepted

A.2 User 4 Results

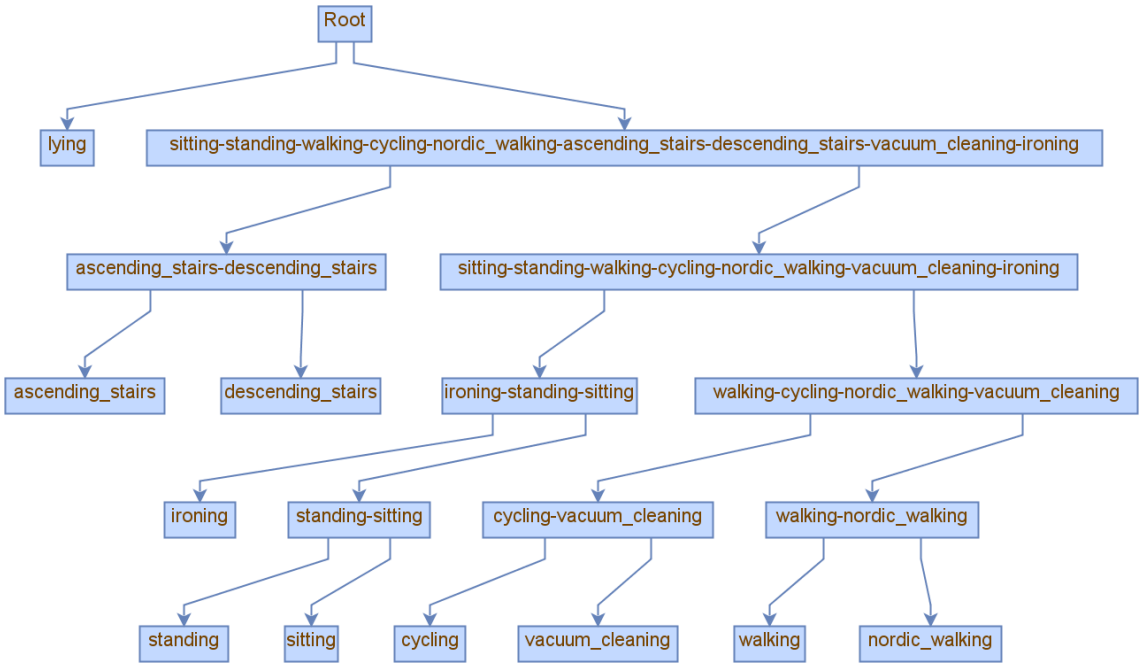

Figure A.3: User 4 initial tree

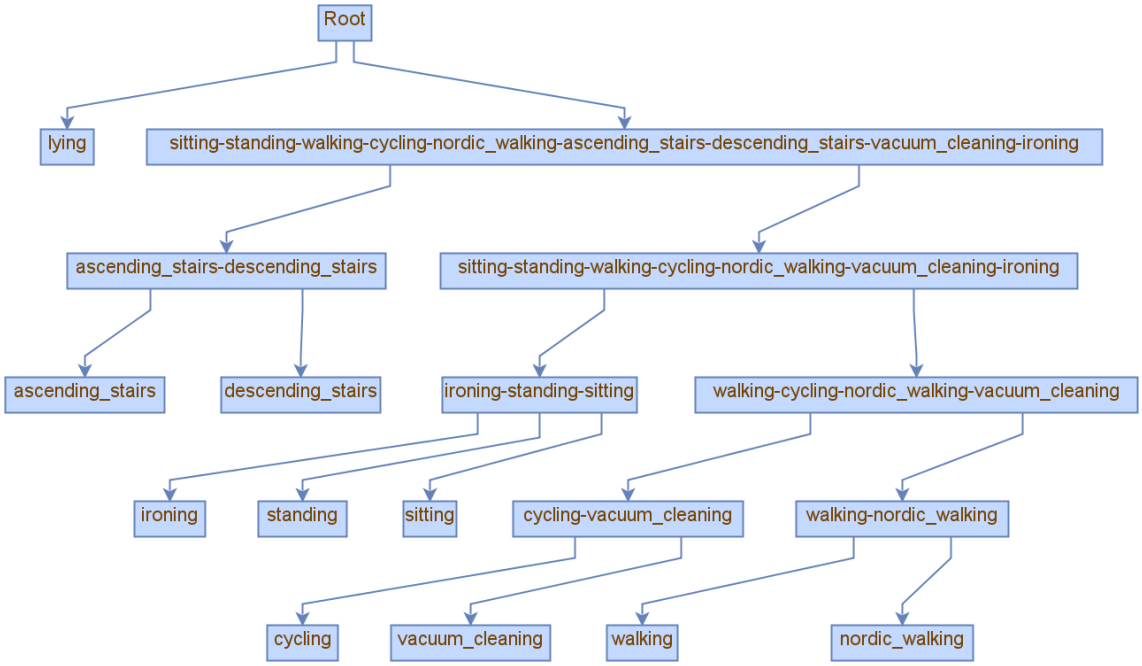

Figure A.4: User 4 tree after first pruning iteration. Pruning was not accepted

A.3 User 5 Results

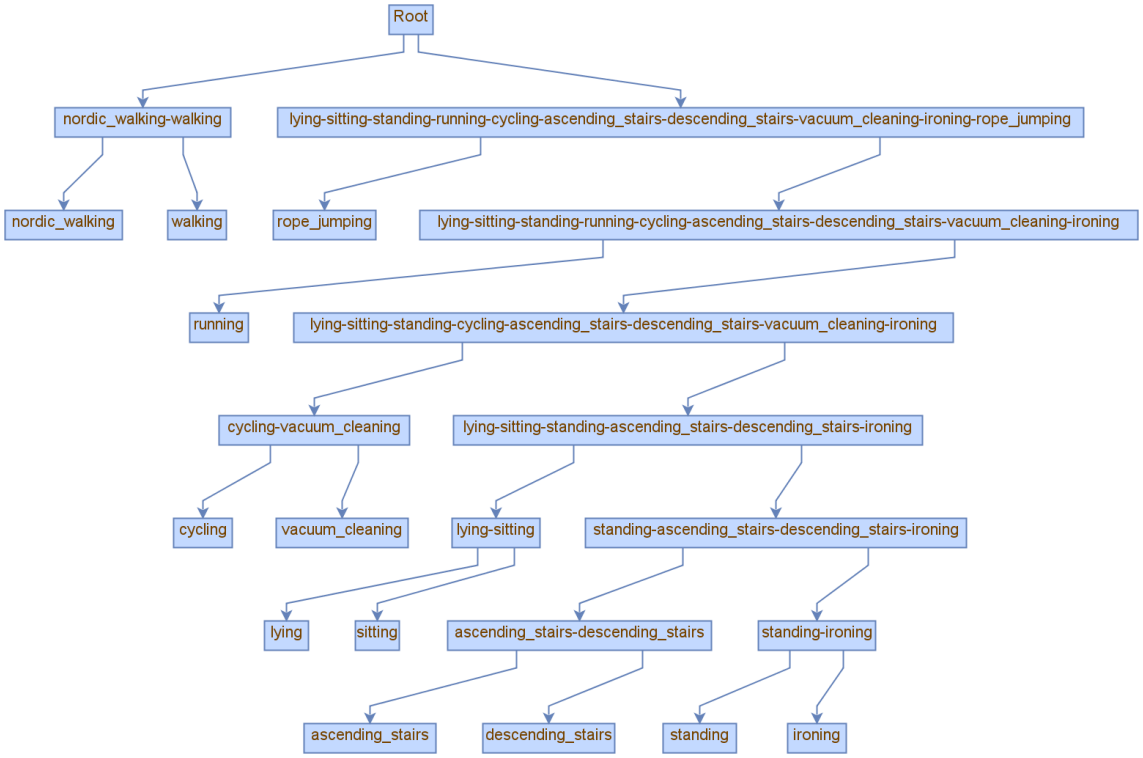

Figure A.5: User 5 initial tree

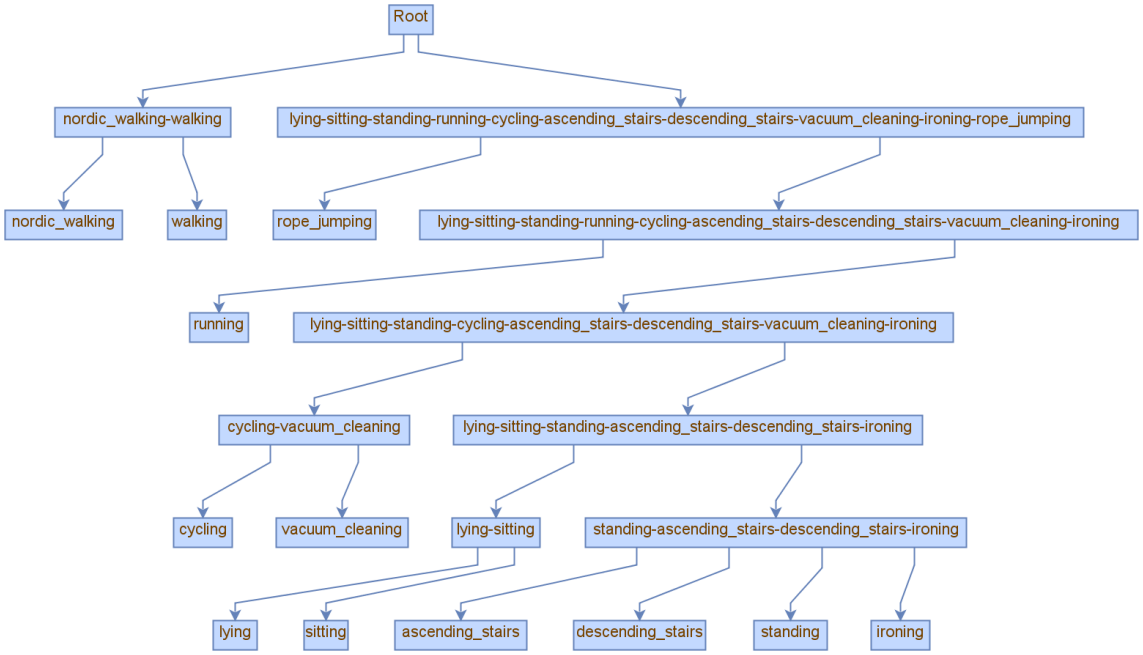

Figure A.6: User 5 tree after first pruning iteration. Pruning was accepted

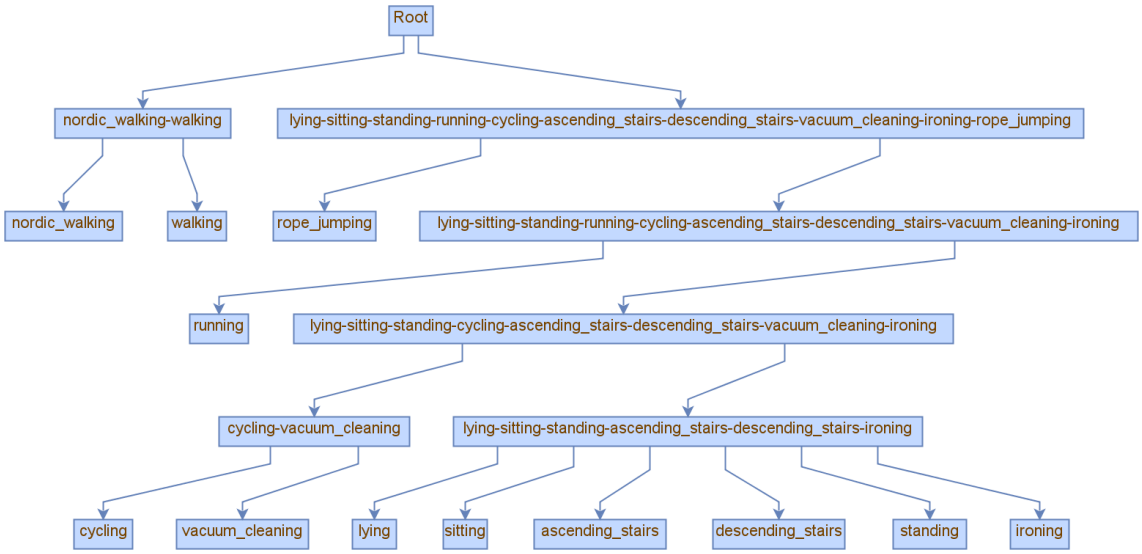

Figure A.7: User 5 tree after second pruning iteration. Pruning was accepted

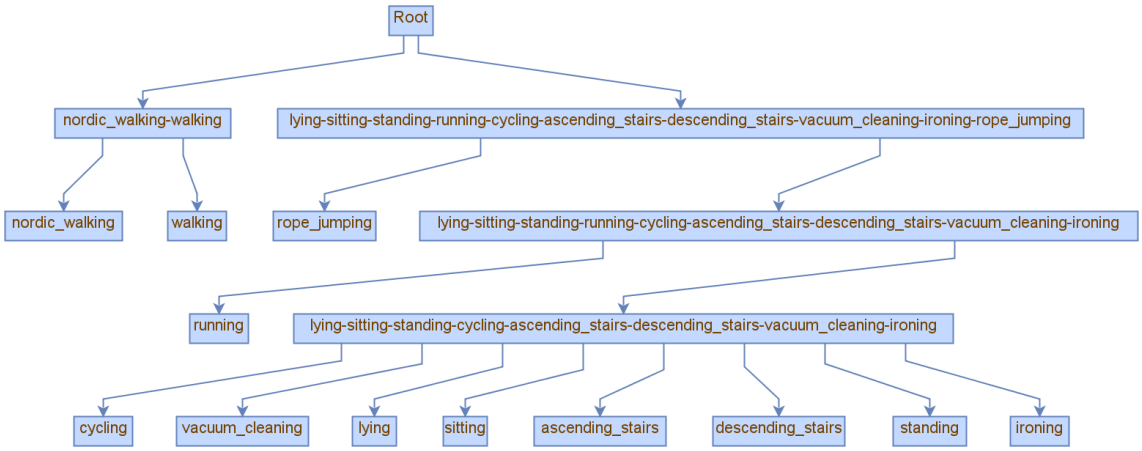

Figure A.8: User 5 tree after third pruning iteration. Pruning was not accepted

A.4 User 7 Results

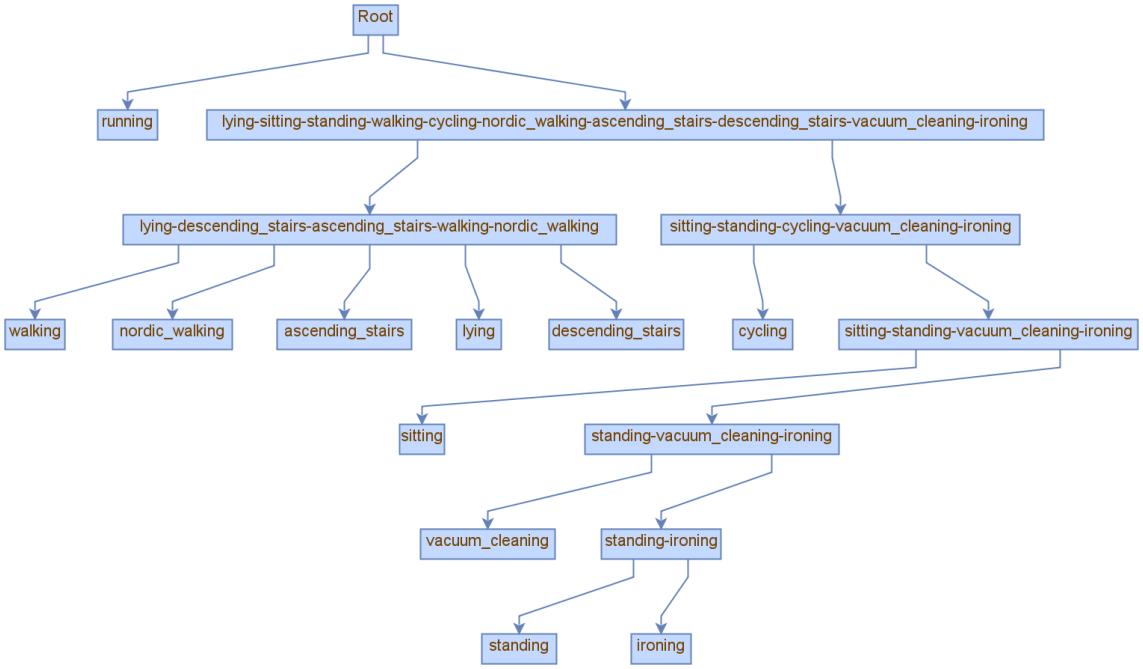

Figure A.9: User 7 initial tree

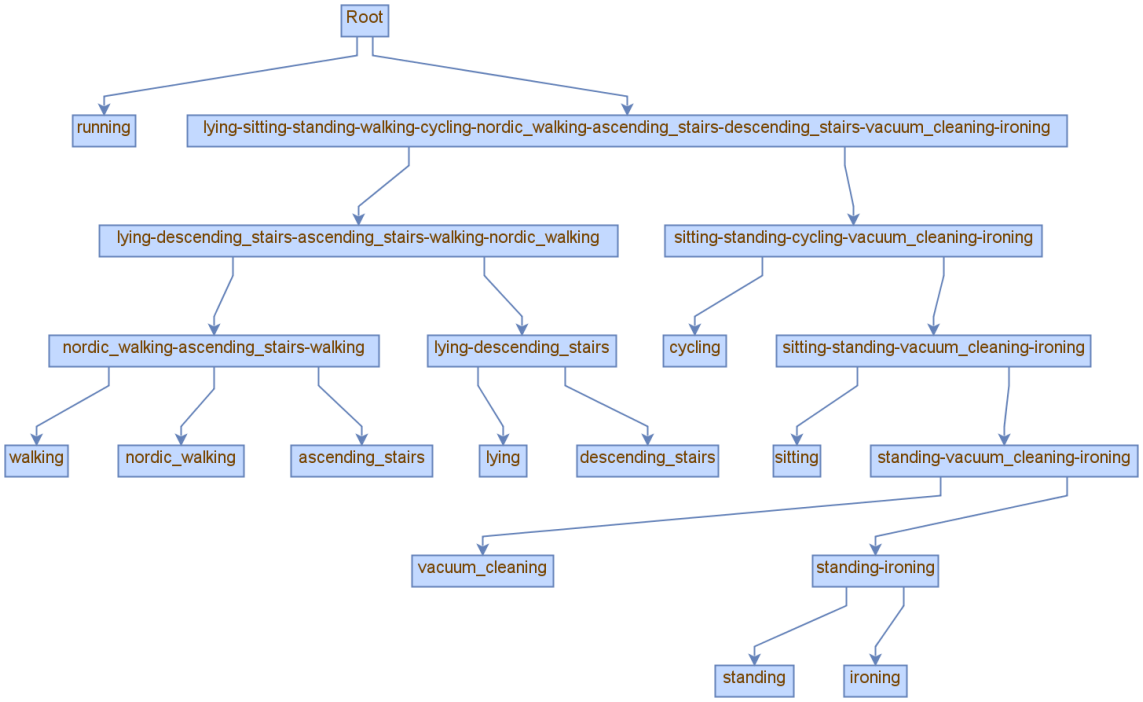

Figure A.10: User 7 tree after first pruning iteration. Pruning was accepted

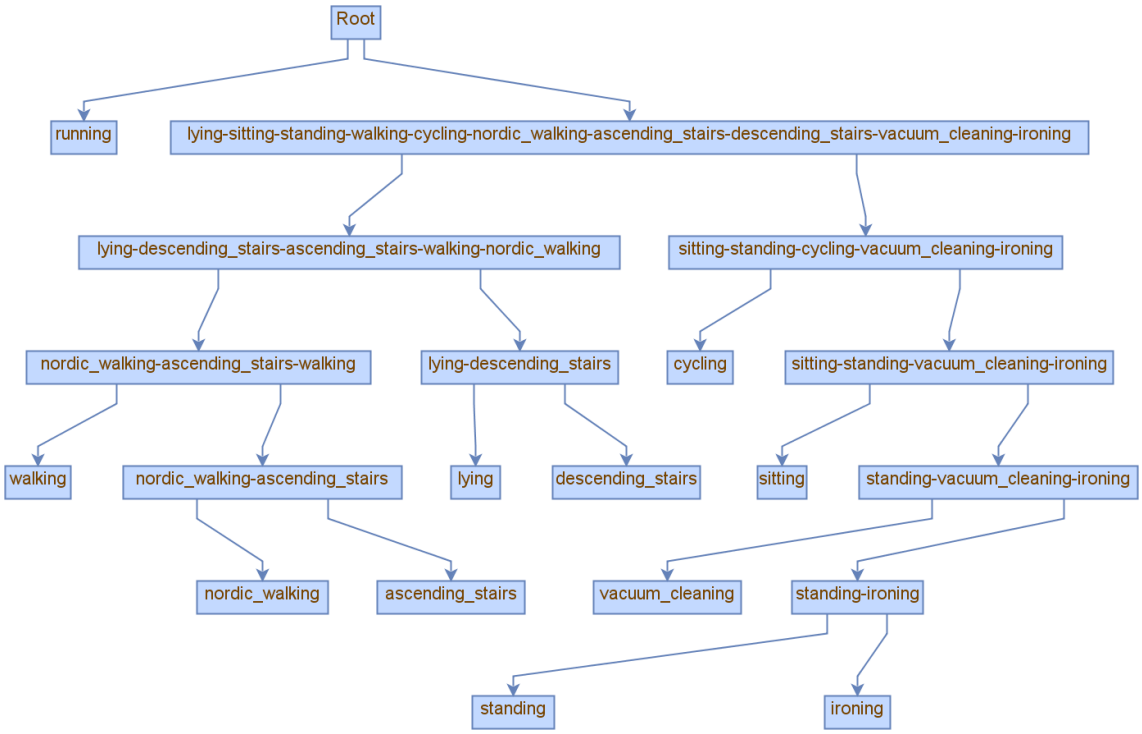

Figure A.11: User 7 tree after second pruning iteration. Pruning was accepted

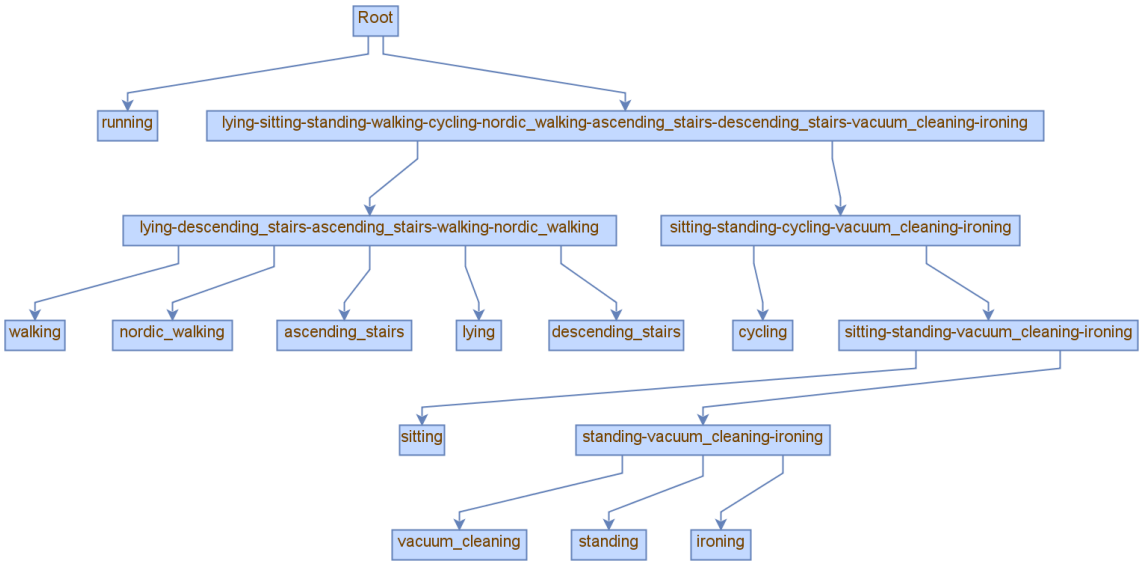

Figure A.12: User 7 tree after third pruning iteration. Pruning was accepted

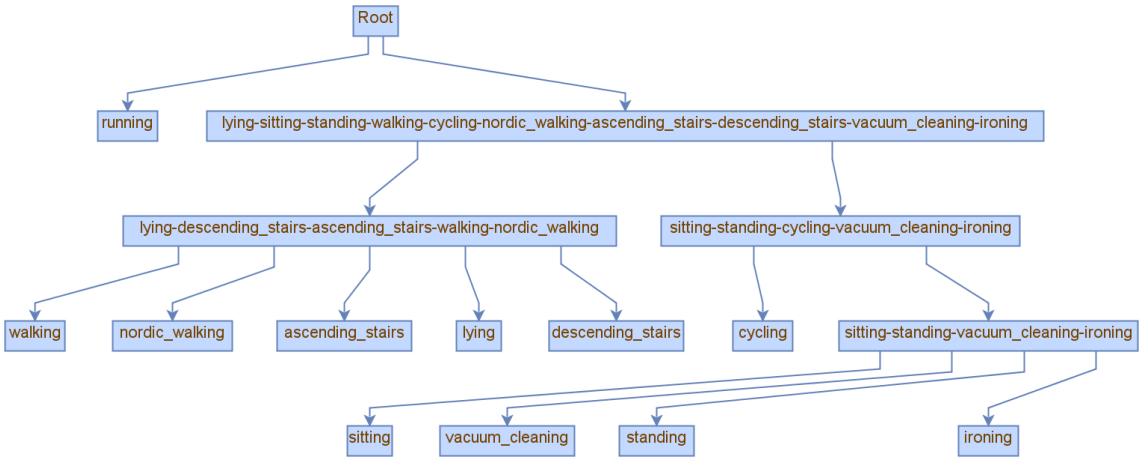

Figure A.13: User 7 tree after fourth pruning iteration. Pruning was not accepted

A.5 User 8 Results

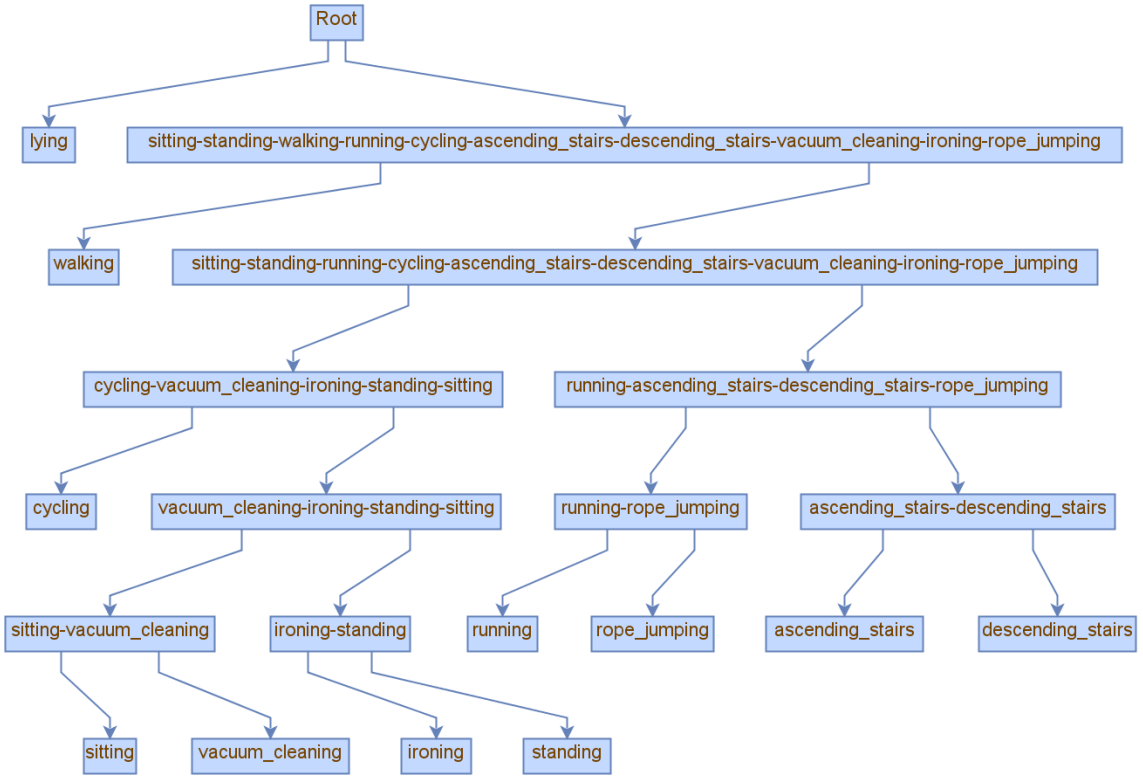

Figure A.14: User 8 initial tree

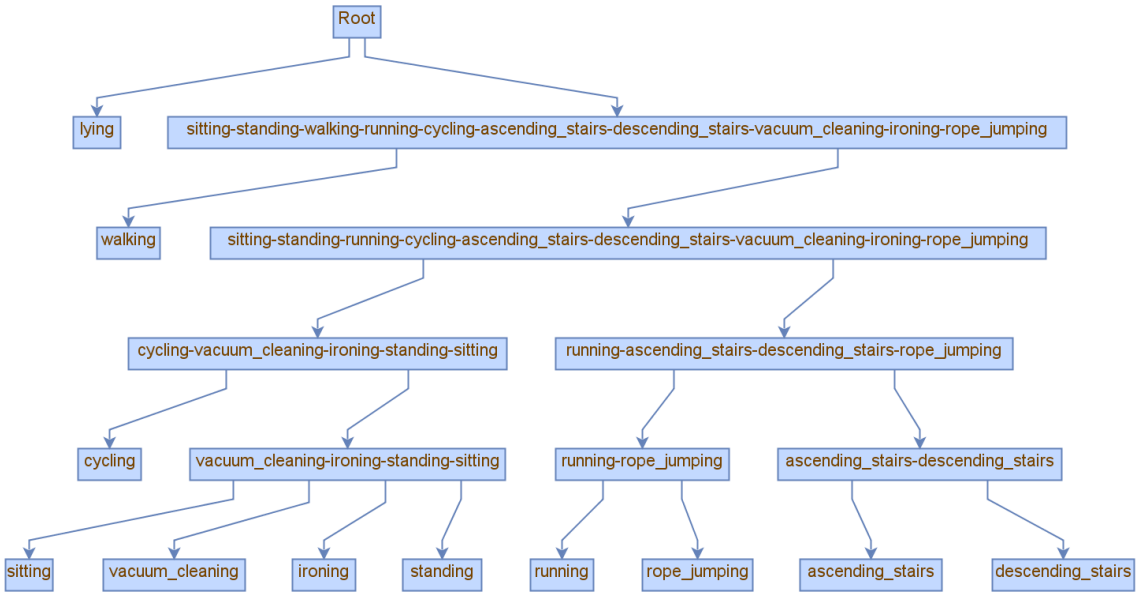

Figure A.15: User 8 tree after first pruning iteration. Pruning was accepted

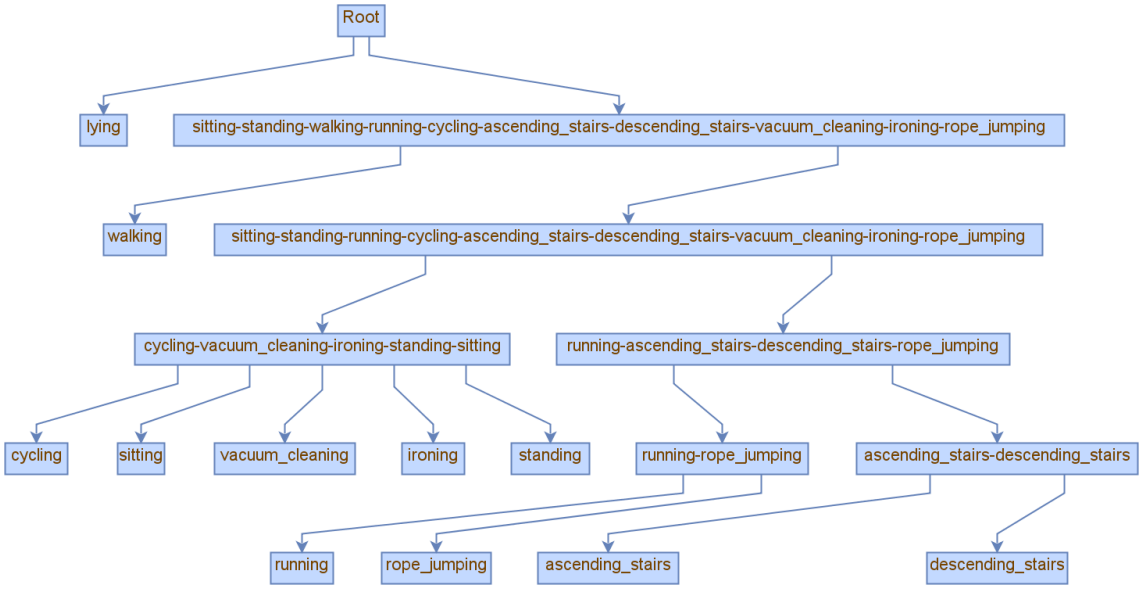

Figure A.16: User 8 tree after second pruning iteration. Pruning was not accepted