Recomendador de Hábitos para reducir el riesgo de padecer Sobrepeso y Obesidad
Habit Recommender to reduce the risk of Overweight and Obesity

Trabajo de Fin de Grado
Curso 2023–2024

Autor
Fan Ye, Daniel Martínez

Director
Jose Ignacio Hidalgo Pérez

Grado de Ingeniería Informática y Grado de Ingeniería de Software

Facultad de Informática

Universidad Complutense de Madrid

Recomendador de Hábitos para reducir el riesgo de padecer Sobrepeso y Obesidad
Habit Recommender to reduce the risk of Overweight and Obesity

Trabajo de Fin de Grado en Ingeniería Informática e Ingeniería de Software

Autor
Fan Ye, Daniel Martínez

Director
Jose Ignacio Hidalgo Pérez

Convocatoria: *Septiembre 2024*

Grado de Ingeniería Informática y Grado de Ingeniería de Software

Facultad de Informática

Universidad Complutense de Madrid

13 de septiembre de 2024

Dedication Page

A nuestros amigos de la carrera, por enseñarnos que todo el mundo puede conseguirlo

Y a los que se quedaron por el camino. Ellos no lo consiguieron :c

Acknowledgements Page

On behalf of Fan, to the friends that accompanied him during these 4 years and to his family for their help and support.

On behalf of Daniel, to every friend he's made throughout the degree, for bearing with him throughout these 4 years, and to his family who believed in him without doubt. And of course, to Fan Ye and his initiative, without him this would have all been impossible.

Resumen

Recomendador de Hábitos para reducir el riesgo de padecer Sobrepeso y Obesidad

En este Trabajo de Fin de Grado se ha diseñado un recomendador de hábitos saludables basado en computación evolutiva. El algoritmo evolutivo se encargará de encontrar valores de un conjunto de variables, es decir, hábitos modificables, para reducir el riesgo de padecer obesidad, teniendo en cuenta una selección de valores que son fijos, como la edad o el sexo.

Para evaluar las diferentes combinaciones de valores que genera el algoritmo evolutivo,utilizamos una serie de modelos obtenidos mediante técnicas de Machine Learning. Los datos utilizados para entrenar los modelos fueron obtenidos en el marco del proyecto Genobia y corresponden a más de 1100 personas, a los que se les ha realizado una encuesta sobre hábitos de vida y situación socioeconómica y un análisis genético.

El objetivo del trabajo presentado en este documento es proporcionar una herramienta práctica que ayude a las personas a realizar cambios en sus hábitos y rutinas con el fin de prevenir o reducir el riesgo de padecer obesidad.

Palabras clave

Computación evolutiva, obesidad, fitness, recomendador de hábito, cromosoma, algoritmo genético

Abstract

Habit Recommender to reduce the risk of Overweight and Obesity

This Bachelor's Thesis presents the design of a healthy habits recommender system based on evolutionary computing. The evolutionary algorithm is tasked with finding values for a set of variables, namely modifiable habits, to reduce the risk of obesity, taking into account a selection of fixed values such as age or gender.

In order to evaluate the different combinations of values generated by the evolutionary algorithm, we employed a series of models obtained through Machine Learning techniques. The data used to train the models was collected as part of the Genobia project, involving over 1100 individuals who completed a survey on lifestyle habits and social-economic status and a genetic analysis.

The objective of the work presented in this document is to provide a practical tool to assist individuals in making changes to their habits and routines in order to prevent or reduce the risk of obesity.

Keywords

Evolutionary computation, obesity, fitness, habit recommender, chromosome, genetic algorithm

Contents

List of figures

Chapter 1

Introduction

"Cuanto peor mejor para todos y cuanto peor para todos mejor, mejor para mí el suyo beneficio político"
— M. Rajoy

In this first chapter we will explain the motivation and objectives of the thesis, along with the work plan.

1.1. Motivation

Overweight and Obesity is a significant global public health issue, linked to numerous chronic diseases such as diabetes, heart disease, and certain cancers (World Health Organization (2024)). As the incidence of obesity continues to rise, the need for effective preventive measures becomes increasingly critical. Traditional approaches to obesity prevention often fail to consider the personalized nature of risk factors, which vary greatly depending on age, lifestyle, and other individual characteristics.

In recent years, evolutionary computation (Ryan et al. (2018)) has emerged as a powerful tool for solving complex optimization problems across various domains. Inspired by natural evolutionary processes, this method has shown considerable promise in addressing health-related challenges by optimizing multiple variables simultaneously. The application of evolutionary algorithms to personalized health recommendations offers a novel approach to mitigating obesity risk, allowing for tailored advice that adapts to the unique needs of each individual.

The motivation behind this work stems from the potential of evolutionary computation to create a more effective, personalized habit recommender system. By optimizing lifestyle factors such as diet and physical activity, the system aims to provide individuals with actionable guidance that can significantly reduce their risk of obesity.

1.2. Objectives

The main objective of this work is to design and implement a habit recommender system that uses evolutionary computation to optimize lifestyle factors to reduce the risk of obesity.

The keys to achieve this objective are:

- **Training** of logical models of the risk of being overweight or obese to evaluate the individuals and quantify how good they are.
- **Optimization** of the values for the selected habit's variables.
- **Development** of a friendly graphical user interface.

1.3. Work plan

We detailed some milestones in the work plan to the keys to achieve the final objective.

1.3.1. Training

- Getting used with the models training with Pancreas Model Tool.
- Train models with PMT.
- Sort the models and select a set number of models for the habit recommender.

1.3.2. Optimization

- Familiarization with the meaning of each variable.
- Development of habit recommender able to run several individuals to collect data.
- Execute the recommender to obtain data, analyze them and modify the values random generation for the variables.

1.3.3. Development

- Development of a graphic user interface connected to the algorithm as a form.
- Implementation of a load function to facilitate filling in the form.
- Offer tailored recommendations as output of the execution.

1.4. GenObIA-CM

The datasets we employed in the development of the habit recommender system for models training and testing are from the GenObIA Project.

The GenObIA Project focuses on the development of predictive algorithms using artificial intelligence to identify individuals at risk of developing overweight, obesity, and associated pathologies.

The Consortium GenObIA is constituted of Research Groups and Associated Groups. One of the Research Groups is the Complutense University of Madrid, specifically the Faculty of Medicine and the Faculty of Computer Science.

Regarding the data we used, Gutiérrez-Gallego et al. (2024):

> In the inclusion study, all participants accepted and signed an informed consent form. The study was approved by the Regional Clinical Drugs Research Ethics Committee of the Community of Madrid (Comité Ético de la Investigación con Medicamentos Regional de la Comunidad de Madrid CEIm-R, Approval Code: 06/2018. Approval date: 28 June 2018) and conducted in accordance with the Declaration of Helsinki. This project is part of the GenObIA consortium of the Madrid Community (GenObIA-CM.B2017/BMD-3773).

For more information, check out their website genobia.es.

1.5. Organization

The rest of this document is organized as follows.

- Chapter 2 emphasizes our approach and presents other similar applications.
- In chapter 3 we described the main contributions to the project.
- Chapter 4 compiles the main technologies we employed in this work.
- In chapter 5 we explained our conclusions and the improvements and future works that can be done.
- In 5.2.5 are stated the contributions each of us made.

Chapter 2

Related Work

"Some things I can do easily, but you can't do at all. And some things you do effortlessly, I could never manage. That's all there is to it"
— Jean RF Magott

Throughout the years, health has been a constant topic of discussion, with numerous theories and recommendations emerging about which habits contribute to a healthier lifestyle. From dietary guidelines to the risks (or benefits, as some studies point to) of alcohol and tobacco consumption, the pursuit of well-being has led to a diverse array of approaches, each claiming to offer the best path to longevity and quality of life.

However, the advancements of digital technology have revolutionized how we understand and manage our health, offering personalized insights that were previously unfathomable.

Our objective is to take a innovative approach, making a program that is able to give customised advice depending on the user's habits by leveraging evolutionary computing. Our application takes the user's habits into account and makes changes relative to the previous ones so that the user does not feel overwhelmed by them.

Recommenders are everywhere in our day to day life, from the ones used on e-commerce to show us related items to the ones we have already bought, to more hidden ones, such as the ones social media use to show us whatever content is related to what we have liked before.

While there are plenty of websites that offer health advice, these only provide generic recommendations that are mostly common knowledge, such as 'Aim for eight hours of sleep a night' or 'Eat a nutritious breakfast every morning', which are not bad suggestions, but they are too standard and non-specific as to make a difference in someone's life if followed one by one, and too overwhelming to implement all at once.

On the other hand, there are other applications that take a more specific approach into a single topic with an emphasis on the user achieving these goals.

Some of these applications are:

- **MyFitnessPal**

  This MyFitnessPal mobile application keeps track of the calories consumed everyday by maintaining a meal diary and informing the user of their nutritional values and components. It also has a more fitness oriented section wherein workout routines are suggested for the user (though it seems more of a secondary function, as it is more hidden). When compared to our application it is a similar approach, yet with a more narrow set of habits, which allows it to focus more on the specifics of the suggestions provided. Laing et al. (2014)

- **Holly Health Habit Coach**

  Holly Health Habit Coach is more oriented towards the creation of healthy habits, divided between fitness related habits (exercising and eating), sleeping habits and mental health habits. Upon login, an AI takes your information and suggests one or two habits so that you can get accustomed to them progressively. According to our tests this application provides less personalized advice, with more focus towards the upkeeping of these habits. In studies like "Small steps, long-term outcomes", "How does Holly Health impact health and well-being" and "Holly Health impacts pain, mood, and self-management skills", the efficacy of this application is shown in terms of habit creation.

- **SmokeFree and Drinkaware**

  SmokeFree and Drinkaware are not mobile apps, but rather websites that provide free advice on how to stop respectively smoking and drinking habits, information of their effects on the user's health, various replacement treatments, and live chats with agents to talk about their day to day problems resulting from these habits. Smoking and drinking are widely known for being noxious towards the user's health, as seen in Dai et al. (2022) and Barnett et al. (2014).

  These applications, instead of giving plain advice, aim to make the user aware of their problem and try to be more helpful towards getting them to quit it by providing temporary replacements and support plans.

With limitations as lack of customization, being limited to a certain scope, being either too insistent with their reminders or users forgetting about them too easily present on the most similar programs, we present a different approach by giving more concrete advice and providing more possibilities to improve health related habits by gathering a broader spectrum of data.

Chapter 3

Work Description

"It ain't much, but it's honest work"
— Dave Brandt

This chapter contains the main contribution of this project. We will go through an introduction, where we will explain the first steps we took, understanding the objectives and getting started with the Pancreas Model Tool (PMT). Then, we will explain how the application works at a high-level and after that, we will describe the development of the habit recommender system.

We developed 2 different versions. The first one does not have a GUI (Graphic User Interface) but can run the program with more than one individual as input. This is the version in which we collect results to analyze and then further improve the algorithm. The second version is the one meant for the user. It has a GUI and utilizes the same logic implementation as the first version.

3.1. First steps

The objective of this project is to create a health advisor, that can offer changes in the user's lifestyle with the goal of finding an improvement in their quality of life.

In order to do that, the first step we took towards was getting acquainted with the usage of the Pancreas Model Tool, a versatile program provided by ABSys-Group (Adaptive and Bioinspired Systems) in order to train the models which will constitute the core of the evaluation system in our application.

We use a group of models to define the bodily status of the user, and strive towards a more healthy state. Our models are logical functions that predicts whether the Body Mass Index (BMI) of a person would be greater than 25 or not. Based on Nuttall (2015), BMI is a metric used to estimate the amount of body fat by using the height and weight. Values above the normal range (18.5-24.9) are considered overweight, and significantly higher values indicate obesity.

After getting used to work with PMT, we made a great number of executions with different parameters, reaching the conclusion that the higher the number of

individuals and generations the more accurate the models became. We also settled other parameters: the crossover rate would be around 70 to 80%, the mutation rate around 5 to 10%, and an elitism of 2%, along with tournament selection, facing 3 randomly chosen individuals and letting the best one go on to the next generation.

Once the training parameters were established, we could finally start the training of the models. However, the means that were available to us at the moment were far from enough to run such heavy programs for long amounts of time. With this in mind, we were given access through ssh to a server owned by ABSysGroup , which we could use to run PMT with more population and iterations, thus getting models with higher accuracy.

3.2. Habit recommender overview

Before getting into the development of the habit recommender system, we will explain how we worked, how does the program work and the elements of the program. We can see in Fig 3.1 how the application is structured and the elements it has.

First of all, we had to train the models. To do so, we used the datasets provided by GenObIA Project and Backus Naur Form (BNF) in PMT. As described in Ryan et al. (2018), BNF is a convenient formal notation for grammars comprised of tuples {N, T, P, S}. N is the set of non-terminal symbols, which are mapped to the set of terminal symbol, T, according to P, the set of production rules. S is a special non-terminal *start* symbol. From the trained models, we select the best one to use as the evaluation system in the algorithm. We will get into the evaluation system later on.

After that, we implemented a functionality to initialize the population based on the initial features the user submits and the different phases of the Genetic Algorithm.

The way the recommender system works is that it receives the users features as input and initializes the population based on that input. Then it evaluates the new population and starts the cycle of the Genetic Algorithm. The population goes through the different phases of the algorithm in every iteration, from the generation of the elite, the selection, the crossover, the mutation, the evaluation and finally the introduction of the elite. When the looping ends, the best individual from the population is selected and outputted as list of recommendations.

3.2.1. Fitness evaluation

The models are logical expressions that vary in length and number of variables. To evaluate the fitness of the individuals, we replace the variables in the models with the values of the respective variables in the individual and evaluate the expressions one by one. If the expression is true, it adds one to the fitness, otherwise, we go to the next model (Fig 3.2).

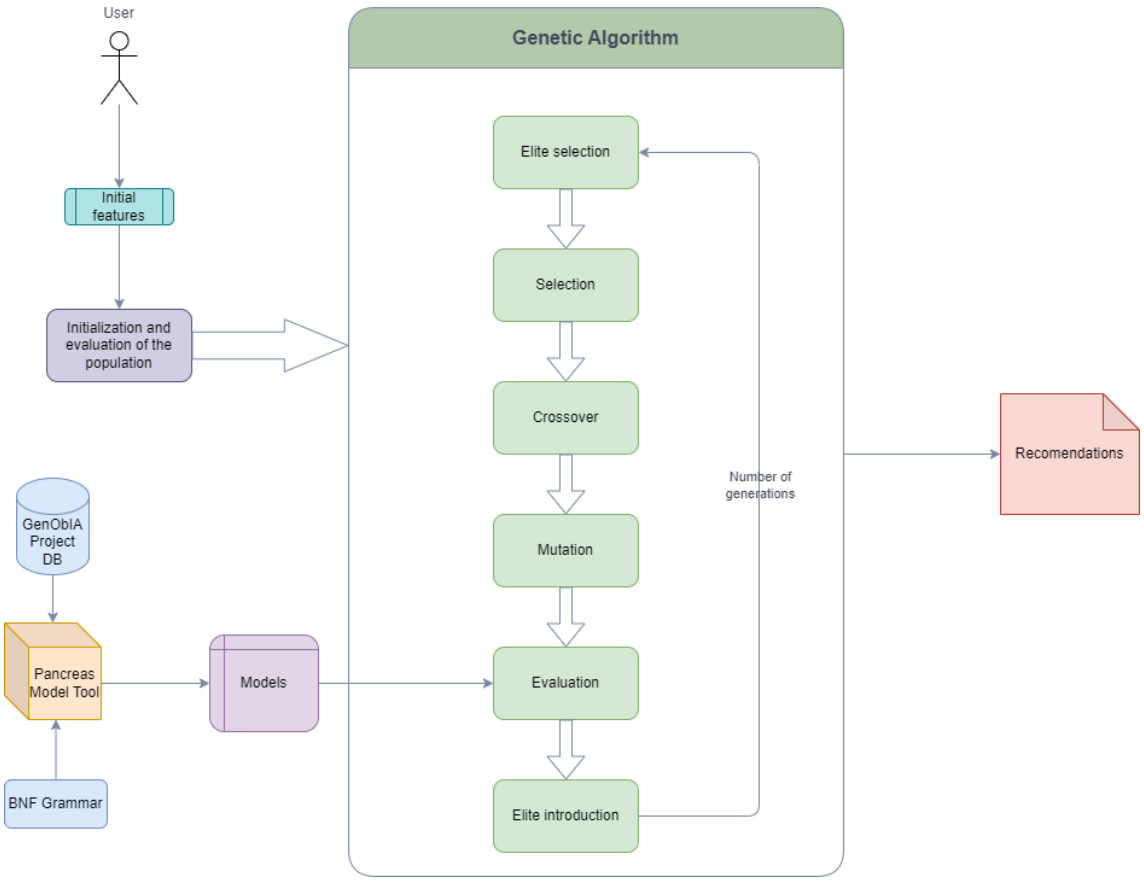

Figure 3.1: Diagram of the program

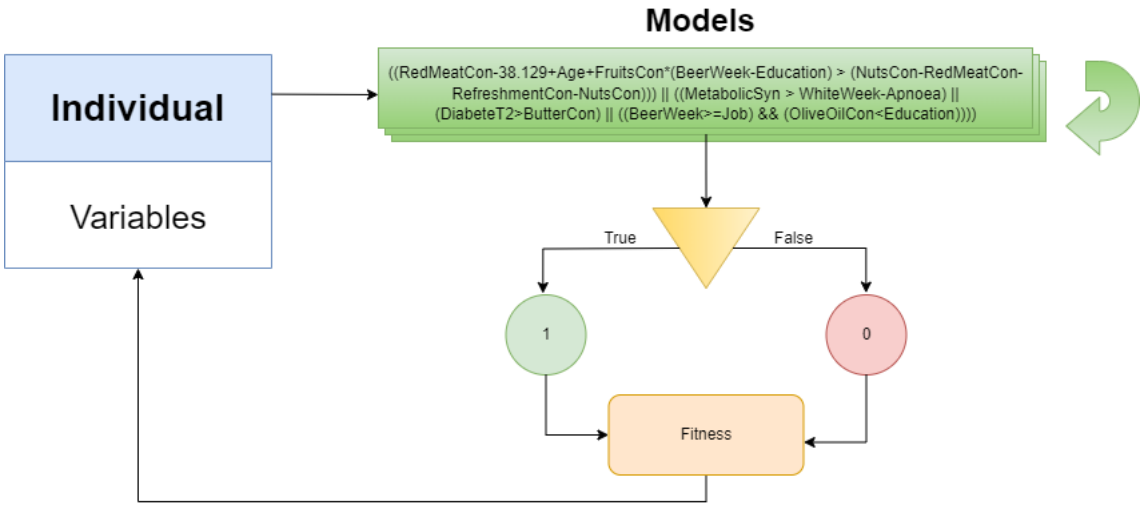

Figure 3.2: Diagram of the fitness evaluation

3.2.2. Workflow

We first decide on which dataset we will work on, then we train the models and execute the program to obtain results. After that, we analyze the results and apply modifications to improve the algorithm. Most of the time there are either modifications of the variables of the dataset or we chose a different one. In both cases, we need to go through the process shown in Fig 3.3 again.

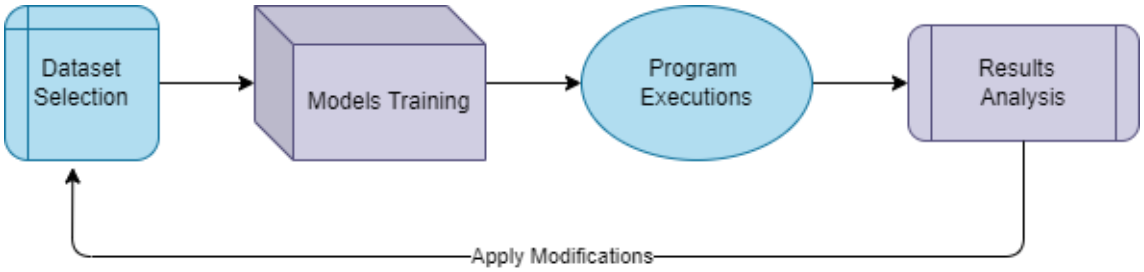

Figure 3.3: Diagram of the workflow

3.3. Variables

The variables we used in our program changed through the development of the habit recommender system but from the start we divided the variables in 2 different groups: the variables that will not change during the execution of the algorithm and the ones that will. The reason is that even though every variable will be taken into account in the evaluation, some elements can be changed but others can not.

3.3.1. Iteration 1

We started using the 37 variables on the filtered datasets because it was easier to manage. Below are the variables with a brief definition of each and the distribution of both groups and shown in Fig 3.4 .

Constants	
Sex	Cancer_col
Age	Cancer_pros
Population	Cancer_lung
Education	Cancer_other
Earning	Heart_attack
Job	Heart_angina
Ex-smoker Year	Heart_failure
Ex-smoker unk	Diabetes
Cancer	Metabolic_syn
Cancer_mam	Apnea
COPD	Asthma

Variables	
Spirit	Smoke
Spirit week	Num smokes
Wine_beer	Pipe
Beer week	Cigar
Wine week	ADH
White week	IPAQ
Pink week	Sleep 8
Stress	

Figure 3.4: Distribution of the variables 1

1. **Sex**, whether the individual is male or female at birth.
2. **Age**.
3. **Population**, the number of people that live in the individual's locality, the ranges are less than 2.500, between 2.500 and 20.000, between 20.000 and 50.000 and more than 50.000.
4. **Education**, the level of education of the individual, it can be no education, elementary education, secondary education or university education.
5. **Earning**, the economic situation of the individual, the ranges are less than 1.000€ per month, between 1.000€ and 2.000€ per month and more than 2.000€ per month.
6. **Job**, the type of job the individual has, from a list of 16 types of jobs.
7. **Stress**, whether the individual is stressed out or not.
8. **Sleep 8**, whether the individual sleeps at least 8 hours.
9. **Spirit**, whether the individual drinks spirit.
10. **Spirit week**, the number of glasses of spirit per week.

11. **Wine beer**, whether the individual drinks wine or beer.
12. **Beer week**, the number of glasses of beer per week.
13. **Wine week**, the number of glasses of red wine per week.
14. **White week**, the number of glasses of white wine per week.
15. **Pink week**, the number of glasses of pink wine per week.
16. **Smoke**, whether the individual smokes cigarettes or not.
17. **Number of smokes**, number of smoked cigarettes per week
18. **Pipe**, number of pipes or pipe charges smoked daily by the user.
19. **Cigar**, number of cigars smoked daily by the user.
20. **Ex-smoker years**, represents how many years ago the individual stopped smoking.
21. **Ex-smoker unknown**, the individual has stopped smoking, but doesn't know for how much time.
22. **Cancer**, whether the individual suffers from cancer.
23. **Cancer mam**, breast cancer.
24. **Cancer col**, colon cancer.
25. **Cancer pros**, prostate cancer.
26. **Cancer lung**, lung cancer.
27. **Cancer other**, other type of cancer.
28. **Heart attack**, whether the individual suffers from myocardial infarction.
29. **Heart angina**, whether the individual suffers from angina pectoris.
30. **Heart failure**, whether the individual suffers from heart failure.
31. **Diabetes**, whether the individual suffers from type 2 diabetes.
32. **Metabolic syn**, whether the individual suffers from metabolic syndrome.
33. **Apnoea**, whether the individual suffers from apnoea.
34. **Asthma**, whether the individual suffers from asthma.
35. **COPD**, whether the individual suffers from chronic obstructive pulmonary disease .
36. **ADH**, whether the individual follows the Mediterranean diet.
37. **IPAQ**, the International Physical Activity Questionnaire is a metric used to measure health-related physical activity.

3.3.2. Iteration 2

During the analysis of recommendations we collected, we noticed that the values for ADH and IPAQ were too broad and hard to grasp, so we took the variables from the unfiltered version of the dataset, which has ADH and IPAQ broken down to more specific variables. Since this is the unfiltered version, there were some variables that we did not need so we cut them off. In the end, there were 21 additions to the variable list.

The distribution of the 56 variables are shown in Fig 3.5. The new variables are all on the **Variables** group since they are replacing ADH and IPAQ and we changed Stress to the **Constants** group because it did not make sense that we would recommend someone to be stressed out or not.

Constants	
Sex	Cancer_col
Age	Cancer_pros
Population	Cancer_lung
Education	Cancer_other
Earning	Heart_attack
Job	Heart_angina
Ex-smoker Year	Heart_failure
Ex-smoker unk	Diabetes
Cancer	Metabolic_syn
Cancer_mam	Apnea
COPD	Asthma
Stress	

Variables	
Spirit	Olive oil cons
Spirit week	Vegetables cons
Wine_beer	Fruits cons
Beer week	Red meat cons
Wine week	Butter cons
White week	Refreshment cons
Pink week	Legume cons
Smoke	Fish cons
Num smokes	Industrial bakery cons
Pipe	Nuts cons
Cigar	White meat cons
Sleep 8	Sauté cons
Intense exer days	Dairy cons
Intense exer min	Skimmed cons
Moderate exer days	Walking days
Moderate exer min	Walking min
Min seated	

Figure 3.5: Distribution of the variables 2

1. **Olive oil consumption**, tablespoons of olive oil consumed per day.
2. **Vegetables consumption**, portions of vegetables eaten per day. Garnish counts as half of a portion.
3. **Fruits consumption**, pieces of fruit eaten per day, including natural fruit juice.
4. **Red meat consumption**, portions of beef or pork, including burgers, sausages, and cold meats, per day. Each portion would be the equivalent of 100 to 150 grams.

5. **Butter consumption**, portions of butter or cream per day. Each portion would be equivalent to 12 grams.

6. **Refreshment consumption**, glasses of carbonated or sweet drinks per day.

7. **Legume consumption**, portions of legumes per week. Each portion would be equivalent to 150g.

8. **Fish consumption**, portions of fish or seafood per week. Each portion would be equivalent of 100 to 150 grams of fish or 4 to 5 pieces of seafood.

9. **Industrial bakery consumption**, times the individual consumes industrial bakery per week.

10. **Nuts consumption**, portions of nuts per week. Each portion would be equivalent to 30 grams.

11. **White meat consumption**, whether the individual prefers eating white meat such as chicken, turkey or rabbit instead of red meat.

12. **Sauté consumption**, times the individual consumes sauté as accompaniment of pasta, rice or other dishes per week.

13. **Dairy consumption**, times the individual consumes dairy products per day.

14. **Skimmed consumption**, whether the dairy products are skimmed or not.

15. **Intense exercise days**, the days in the last week the individual has done intense exercises.

16. **Intense exercise minutes**, the average time the individual spends doing intense exercises per day.

17. **Moderate exercise days**, the days in the last week the individual has done moderate exercises.

18. **Moderate exercise minutes**, the average time the individual spends doing moderate exercises per day.

19. **Walking days**, the days in the last week the individual has gone for a walk.

20. **Walking minutes**, the average time the individual spends walking per day.

21. **Minutes seated**, the average time the individual has spent sit in the last week.

3.3.3. Iteration 3

After analyzing the results with the new group of variables, we decided to fuse the number of days per week of a type of exercise and the minutes per day of said exercise because as they are related, they should be evaluated together. To do so, we modified the dataset by multiplying the column of the days per week with the one with the minutes per day of each type of exercise.

As we modified the dataset, we spotter some irregular values in the columns of walking and sitting. There were many cells with values between 1 and 9 and we assumed the reason being the original value is unknown. We calculated the average value of minutes of walking per week (435 minutes) and the minutes of sitting per week (264 minutes) and replaced the irregular values with these average values. 248 cells were changed for walking and 63 for sitting. The new dataset contains 53 variables, its distribution is shown in Fig 3.6 and the final list of variables is listed below:

Constants	
Sex	Cancer_col
Age	Cancer_pros
Population	Cancer_lung
Education	Cancer_other
Earning	Heart_attack
Job	Heart_angina
Ex-smoker Year	Heart_failure
Ex-smoker unk	Diabetes
Cancer	Metabolic_syn
Cancer_mam	Apnea
COPD	Asthma
Stress	

Variables	
Spirit	Olive oil cons
Spirit week	Vegetables cons
Wine_beer	Fruits cons
Beer week	Red meat cons
Wine week	Butter cons
White week	Refreshment cons
Pink week	Legume cons
Smoke	Fish cons
Num smokes	Industrial bakery cons
Pipe	Nuts cons
Cigar	White meat cons
Sleep 8	Sauté cons
Intense exer min week	Dairy cons
Moderate exer min week	Skimmed cons
Walking min week	Min seated

Figure 3.6: Distribution of the variables 3

1. **Sex**, whether the individual is male or female at birth.

2. **Age**.

3. **Population**, the number of people that lives in the individual's locality, the ranges goes from less than 2.500, between 2.500 and 20.000, between 20.000 and 50.000 and more than 50.000.

4. **Education**, the level of education of the individual, it can be no education, elementary education, secondary education or university education.

5. **Earning**, the economic situation of the individual, the ranges are less than 1.000€ per month, between 1.000€ and 2.000€ per month and more than 2.000€ per month.
6. **Job**, the type of job the individual has, from a list of 16 type of jobs.
7. **Stress**, whether the individual is stressed or not.
8. **Sleep 8**, whether the individual sleeps at least 8 hours.
9. **Spirit**, whether the individual drinks spirit.
10. **Spirit week**, the number of glasses of spirit per week.
11. **Wine beer**, whether the individual drinks wine or beer.
12. **Beer week**, the number of glasses of beer per week.
13. **Wine week**, the number of glasses of red wine per week.
14. **White week**, the number of glasses of white wine per week.
15. **Pink week**, the number of glasses of pink wine per week.
16. **Smoke**, whether the individual smokes cigarettes or not.
17. **Number of smokes**, number of the cigarettes per week
18. **Pipe**, number of pipes or pipe charges smoked daily by the user.
19. **Cigar**, number of cigars smoked daily by the user.
20. **Ex-smoker years**, represents the years that the individual stopped smoking.
21. **Ex-smoker unknown**, the individual stopped smoking, but didn't know for how much time.
22. **Cancer**, whether the individual suffers from cancer.
23. **Cancer mam**, breast cancer.
24. **Cancer col**, colon cancer.
25. **Cancer pros**, prostate cancer.
26. **Cancer lung**, lung cancer.
27. **Cancer other**, other type of cancer.
28. **Heart attack**, whether the individual suffers from myocardial infarction.
29. **Heart angina**, whether the individual suffers from angina pectoris.
30. **Heart failure**, whether the individual suffers from heart failure.

31. **Diabetes**, whether the individual suffers from type 2 diabetes.
32. **Metabolic syn**, whether the individual suffers from metabolic syndrome.
33. **Apnoea**, whether the individual suffers from apnoea.
34. **Asthma**, whether the individual suffers from asthma.
35. **COPD**, whether the individual suffers from chronic obstructive pulmonary disease.
36. **Olive oil consumption**, tablespoons of olive oil consumed per day.
37. **Vegetables consumption**, portions of vegetables eaten per day. Garnish counts as half of a portion.
38. **Fruits consumption**, pieces of fruit eaten per day, including natural fruit juice.
39. **Red meat consumption**, portions of beef or pork, including burgers, sausages, and cold meats, per day. Each portion would be the equivalent of 100 to 150 grams.
40. **Butter consumption**, portions of butter or cream per day. Each portion would be equivalent to 12 grams.
41. **Refreshment consumption**, glasses of carbonated or sweet drinks per day.
42. **Legume consumption**, portions of legumes per week. Each portion would be equivalent to 150g.
43. **Fish consumption**, portions of fish or seafood per week. Each portion would be equivalent of 100 to 150 grams of fish or 4 to 5 pieces of seafood.
44. **Industrial bakery consumption**, times the individual consumes industrial bakery per week.
45. **Nuts consumption**, portions of nuts per week. Each portion would be equivalent to 30 grams.
46. **White meat consumption**, whether the individual prefers eating white meat such as chicken, turkey or rabbit instead of red meat.
47. **Sauté consumption**, times the individual consumes sauté as accompaniment of pasta, rice or other dishes per week.
48. **Dairy consumption**, times the individual consumes dairy products per day.
49. **Skimmed consumption**, whether the dairy products are skimmed or not.
50. **EIMS**, minutes per week of intense exercise.
51. **EMMS**, minutes per week of moderate exercise.

52. **ECMS**, minutes per week of walking.

53. **Minutes seated**, the average time the individual has spent sit in the last week.

3.4. Models training

In this section we explain the process followed for the training of all the models used throughout the development of the program. We have divided it into three iterations, which are the same as the previous section, due to the tight entanglement between these two topics.

3.4.1. Iteration 1

After getting acquainted with the workings of the PMT, we ended the generation of models with the GUI and started using the virtual machine.

The main difference between running the PMT on our hardware and on the server was that we couldn't use its graphic interface, we had to set up a property file and run it on the console. Regarding the property file, we had some issues at first because there were some fields we didn't know. However, with the help of Daniel Parras Rodríguez, a doctorate student who had worked previously on the PMT and had more profound knowledge on it, we managed to solve the issues and get the program running on the server.

At first, we used 30 datasets that had been processed to train our models, running the algorithm 30 times with a 50% mutation rate (which is higher than normal because only one gene was able to mutate each time on each individual), a 50% crossover rate (which is lower than normally used ones because here, crossover is more destructive, so we want it to happen less comparatively), and 100 generations with 500 individuals on each generation. The type of selection used is tournament, with 3 individuals competing between them.

3.4.2. Iteration 2

Once knowing how to set up the properties file properly, there was not much more to the training of the models, the biggest difference would be that we were using a single file of raw data (though slightly modified, in order to eliminate gene variables) instead of 30.

Same as last time, we used a 50% mutation rate, a 50% crossover rate, and 100 generations with 500 individuals on each generation. The type of selection used is tournament, with 3 individuals competing between them.

3.4.3. Iteration 3

We first decided that with just 50 models the program would have enough accuracy without sacrificing efficiency on the time consumption.

The dataset we used is a modification of the version mentioned in the second iteration before, running the algorithm 3000 times with a 20% mutation rate , a 70% crossover rate, and 500 generations with 500 population size on each generation. The type of selection used is tournament, with 3 individuals competing between them, and the maximum depth of the models is of 5 nodes. After that we selected the top 50 models sorted by fitness and used them to evaluate the individuals.

Later on, we performed the analysis of the time the application took on each part of the execution and realized that the evaluation of the models took less time than expected, thus we decided to increase the number of models. To do so, we trained a new batch of models and selected the 50 best ones to add to the existing models.

3.5. Individual design and implementation

The Individual Class contains the chromosome with the information of the individuals and evaluates the quality with the fitness evaluation system.

3.5.1. Chromosome

Initially, we decided to use BigDecimal as the type of the elements of the chromosome because they store decimals with a higher accuracy than double. However, during the testing with JEXL, we found out that BigDecimal type number are not capable of using basic arithmetic operators, instead they use functions: add, subtract, divide and multiply, thus making it impossible for us to work with the functions we got from PMT. For that reason, and because of the fact that we did not need such a high accuracy for the variables in our program, we decided to use double instead of BigDecimal in our chromosome array.

Each allele of the chromosome of the individuals would represent a element of the variable list. As some elements can not be modified due to their nature, we implemented an auxiliary boolean array to go through the chromosome, accessing only to the variable ones.

3.5.2. Initialization

One of the most challenging parts of our program is the randomization of the alleles. Each variable represents a different feature so we group them based on their magnitude. Every time the dataset changes, we also modified the groups as we understood better the variables. Even then, the categorization and the range limitations we made are mostly based on what we considered more logical after

reviewing the results as we lack medical knowledge. The function that returns a random value based on the variable is called getLimitedRandom and contains the following groups:

1. The value is binary. This group includes **Sleep 8** and **White meat consumption**. Both are yes no questions with no strings attached.

2. The value is binary, it can only be higher or equal than the original. This group only includes **Skimmed consumption**. If the individual takes skimmed products, means that they probably can not eat lactose, so if the original value is yes, that is 1, it should stay that way.

3. The value is binary, but it can only be lower or equal to the original value. This group includes **Spirit**, **Wine beer** and **Smoke**. It does not make sense to recommend drinking or smoking to the individual if they were not doing it in the first place.

4. The value can only be equal or lower than the original. This group includes **Spirit week**, **Beer week**, **Wine week**, **Pink week**, **Number of smokes**, **Pipe** and **Cigar**. We considered that the values higher than the original could not contribute positively to the fitness because these variables are considered bad habits, so we cut them off. We also added a condition that is if the original value of the yes or no question related to these variables is no, that is 0, these variables' value is automatically set to 0 as well.

5. The value can only be equal or lower than the original. This group includes **Butter consumption**, **Refreshment consumption**, **Industrial bakery consumption** and **Minutes seated** and shares the same logic than the previous group, but without the extra condition.

6. The value can change up to 2, either increasing or decreasing, from the original. This group includes **Olive oil consumption**, **Vegetables consumption**, **Fruits consumption**, **Red meat consumption**, **Diary consumption**, **Fish consumption** and **Nuts consumption**. We considered that it is difficult to suddenly change the diet so we decided on a smaller range in this group.

7. This group includes **EIMS**, **EMMS** and **ECMS**. The values can change up to 30, 120 and 270, respectively, either increasing or decreasing, from the original. The variation increases as the exercises' intensity decreases.

3.5.3. Evaluation system

In order to implement the evaluation system in our application, we needed to transform the models we had into logical operations. After some research and testing different libraries, we found JEXL. It creates a JexlExpression with a string as an input and evaluates it as a logical operation. The last piece left to finish the parser

was a regular expression that replaces the variables of the models with the actual value of our individual.

We also overrode the compareTo() function to sort the individuals using the sort() function from the Collections library and toString() to show the results on the console.

3.6. Algorithm design and implementation

The evolutionary algorithm we implemented in our app is the standard version. We chose the simpler approach because we wanted to lessen its burden on the efficiency.

For the selection phase, we decided to implement a deterministic tournament (Miller y Goldberg (1995)) with a tournament size of 3. We randomly select 3 individuals from the population and keep the one with the highest fitness, then repeat as many times as the population size. It may not be the simplest method but it is not fully random either so we considered this strength good enough to sacrifice some efficiency

For the crossover phase, we chose the uniform crossover. This method goes through all the variables of the chromosome and each one of them has 50% chance to swap between the two selected individuals. We decided to implement this method because each allele of the chromosome represents a different feature of the individual and it does not make sense to mix values from different features. As for the crossover rate.

For the mutation phase, we implemented the basic mutation with 5% mutation rate. It goes through all the variables and when it mutates, it gets a random value from the function getLimitedRandom(). The reason we implemented this method is because it is the simplest and easiest.

For the evaluation, we recalculate the fitness of the individuals that suffered modifications and then we sort the population.

We also implemented 2% of elitism in our algorithm. It is a technique used to ensure that the best individuals of each iteration are preserved and carried over to the next iteration without being altered. In our case, we select the elite before the selection and then when we introduce them into the population, we check whether the elite is better than the worst individuals in the population. If so, they are replaced and the population is sorted again.

Regarding the population size and the number of generations, we recorded the time the program took with different values in both population size and number of generations and we settled them at 100 in both. We will go into more details below.

3.6.1. Time analysis

We made some modifications to record the time each function takes in each execution and there are clearly two phases that takes most of the time. The crossover takes around the 48% of the total time and the evaluation takes around 50% of the total time. The reason being that the crossover function has to go over the population and each allele of the chromosome and the evaluation function has to sort the population.

We also made some executions with 100 randomly selected individuals from the latest dataset, trying different parameters to test the difference of improvement and the difference of time it took the program run. We tried with 100 as population size and 100 as number of generations, 100 and 150, 150 and 100 and finally, 150 and 150. We utilized the same seed in these four executions and the same input, 100 individuals and 100 models. As in Fig 3.7, the time each execution took, showed in milliseconds, differs but the total improvement are very similar, thus we decided to set the population size as 100 and the number of generations as 100.

A noteworthy detail is that the limitations of hardware affects the time the program takes to execute. The executions made to extract the data showed in Fig 3.7 were run on the ABSysGroup's server.

pop100_gen100						
Total Time	**Original Fitness total**	**New Fitness Total**	**Time/Person**	**Minutes**	**Fitness Improv/Person**	**Total improvement**
12182960	3520	5719	121829,6	2,03049333	21,99	2199
pop100_gen150						
Total Time	**Original Fitness total**	**New Fitness Total**	**Time/Person**	**Minutes**	**Fitness Improv/Person**	**Total improvement**
18507911	3520	5729	185079,11	3,08465183	22,09	2209
pop150_gen100						
Total Time	**Original Fitness total**	**New Fitness Total**	**Time/Person**	**Minutes**	**Fitness Improv/Person**	**Total improvement**
18653127	3520	5727	186531,27	3,1088545	22,07	2207
pop150_gen150						
Total Time	**Original Fitness total**	**New Fitness Total**	**Time/Person**	**Minutes**	**Fitness Improv/Person**	**Total improvement**
27707884	3520	5729	277078,84	4,61798067	22,09	2209

Figure 3.7: Time and Performance Analysis

3.7. Graphic User Interface

In this section we describe the implementation of the user interface present in our program.

3.7.1. Description of the main window

We made the user interface of the application using Java's swing library, along with the custom class ConfigPanel that we had used previously during the degree

on the Evolutionary Programming subject, and is able to write on other classes variables directly from the front end if they have getter and setter methods.

The main window of our program extends from a JFrame, and contains 6 different panels. It also has 2 buttons and a progress bar to show the status of the execution.

Each of these 6 panels has text boxes, combo boxes and check boxes in which the user can to input the data. Every change the user makes update the status of IndividuoOriginal, a class translates the user's data into an individual in the algorithm.

The left button is "Cargar parámetros". It opens a dialog where the user can select a file. It is made to load parameters from a comma-separated values file that represent the data from the individual. We had some problem because being ConfigPanel a custom class as we did not know where the value of the front end was stored, so we had trouble setting it at the moment of loading parameters, but we ended up figuring it out by trial and error.

The right button is "Ejecutar", made to start the execution of the program with the set parameters (or, if there are no set parameters, the minimum of each field).

We also had some issues with the progress bar due to us not knowing at first that Java Swing Utilities only run on a single thread, therefore giving us problems at the moment of updating the progress bar because it conflicted with the update of the run button. To solve this, we tried many things, such as keeping a thread constantly updating the progress bar launched from the main program and launching the same thread from the button and launching simultaneously the run of the program.

The solution to our problem was finally using a SwingWorker class object, but instead of launching both the execution and the updates from the progress bar from it, we launched it like a thread from the MainWindow's builder and kept it running in the background, updating the progress bar every 100 milliseconds for the entirety of the program's execution.

3.7.2. Description of the result window

Finally, when the program's execution is done, a pop-up appears on the middle of the screen displaying the results of the execution, telling the user what changes they should make to their habits. Additionally, there are 2 buttons that allow the user to save the suggestions to a text file, to be able to them later on without keeping the program open.

We also wanted to update the UI to a more pleasant one compared to the default look of Java's UI. To do so, we used Java's UIManager setLookAndFeel function, along with a custom Look & Feel packet called FlatLaf, to change the aspect of our application to a more simplistic and refined one.

3.8. Demo

Upon opening the application, we see 6 different panels in which the user is supposed to input the corresponding data (Fig 3.8).

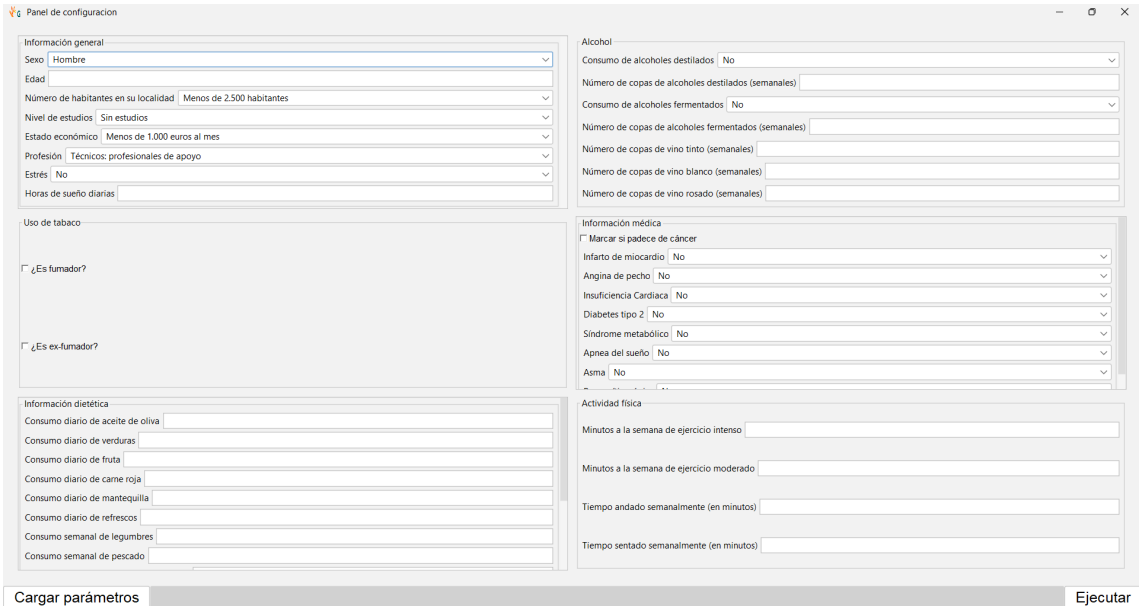

Figure 3.8: Application upon opening

There are three kinds of parameters (Fig. 3.9):

- Choice parameter: consisting on a JComboBox with various options, which are translated into an integer for the back end to process.
- Integers: consisting on a JTextLabel in which only an integer can be input.
- Double: the same as the Integers, but these only accept double values (with a decimal part).

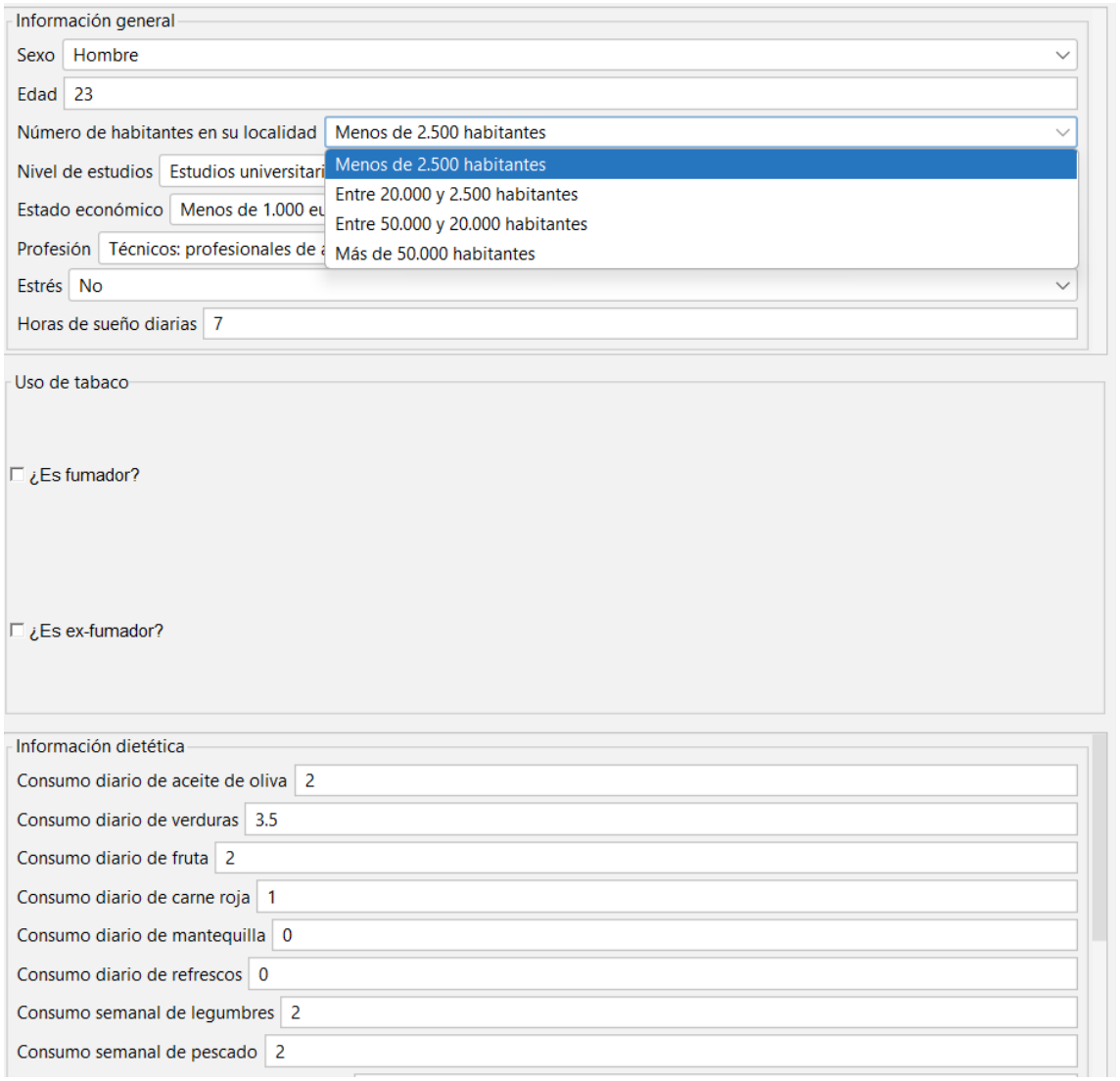

Figure 3.9: The three different kinds of parameters present in our program

Another option is to use the load button on the bottom left to read the values of the parameters off a file chosen by the user on a JFileChooser, thus setting them all on the UI (Fig. 3.10).

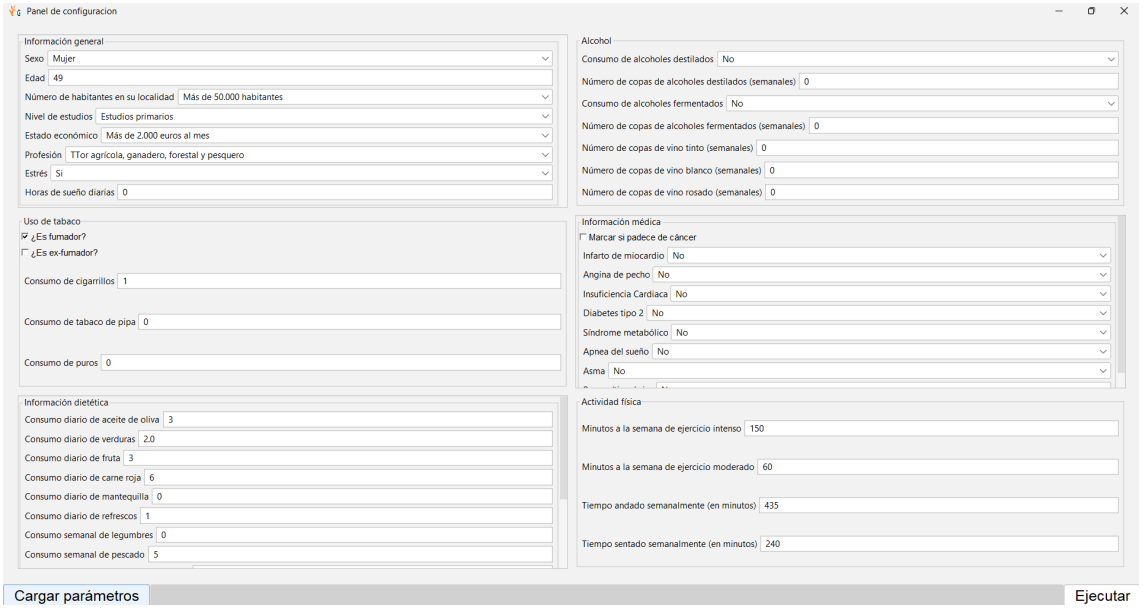

Figure 3.10: Upon pressing the load button, the values are displayed on the front end for the user to see and change

Once all parameters are set, the user can push the run button to start the execution, which also starts the progress of the charge bar in the bottom (Fig. 3.11). When the bar ends, the program is done and the outcome is shown through a smaller window that appears in the middle of the screen (Fig. 3.12).

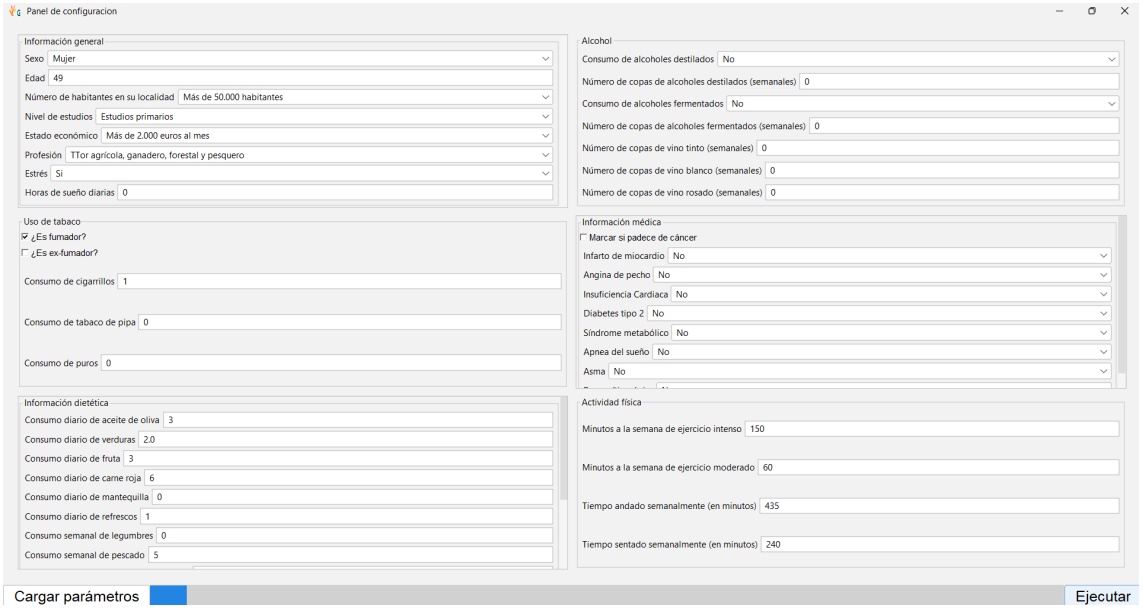

Figure 3.11: After pressing the run button, the program will run in the background, while the progress bar keeps the user informed about the execution's progress

Additionally, we made an .jar that can be executed to try the application (demo version). That said, it is important to execute the file from the console, as it will not work otherwise.

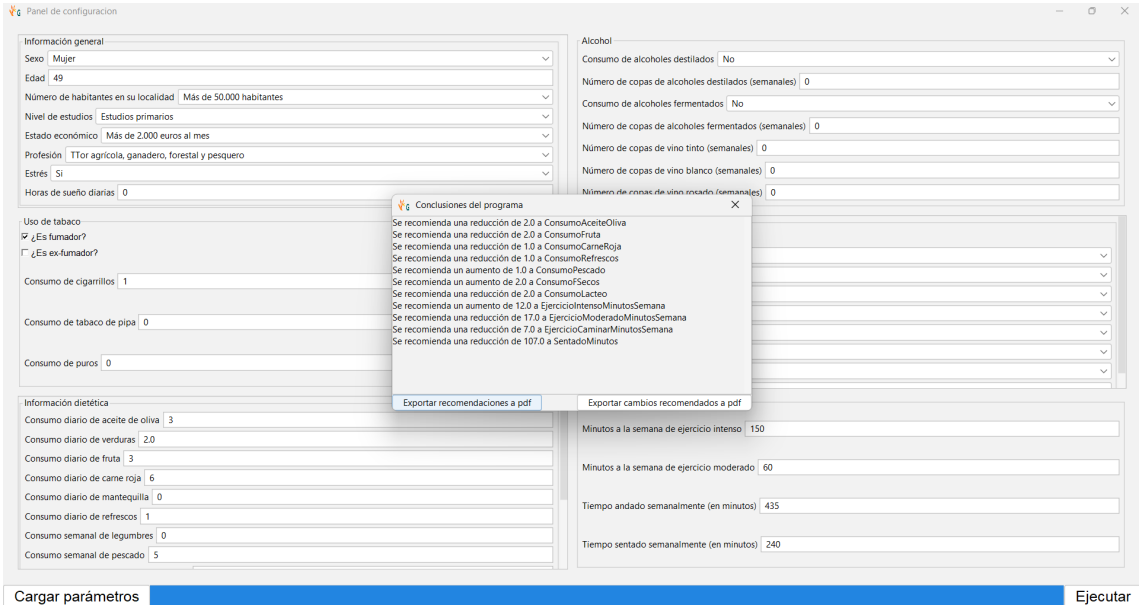

Figure 3.12: When the execution is done, a screen with the suggestions pops up

3.9. Other analysis

3.9.1. Analysis of models

We collected all the models we trained during the third iteration and made a histogram based on the accuracy of these models to prove the robustness of the models. With a sample size of 6.000 models, we have many repeated accuracy values, but each have a different model. Additionally, we can see in the histogram in Fig 3.13 that the data follows a normal distribution (International Encyclopedia of the Social Sciences (2024)). The data and the Python program we used to generate this histogram can be found in our GitHub repository.

3.9.2. Analysis of correlation

Correlation refers to a statistical measure that describes the extent to which two or more variables move in relation to each other. In simpler terms, it quantifies the degree to which changes in one variable are associated with changes in another (Wells (2024)).

We employed Python libraries pandas, seaborn, and matplotlib to calculated the correlation between the variables and the fitness of the individuals and the heat maps.

The data we worked on are the entire dataset and they are biased. Almost half of the samples are from young people from 18 to 30 years old and most of them exercise regularly.

Regarding the making of the graphics, we modified the output of our program to

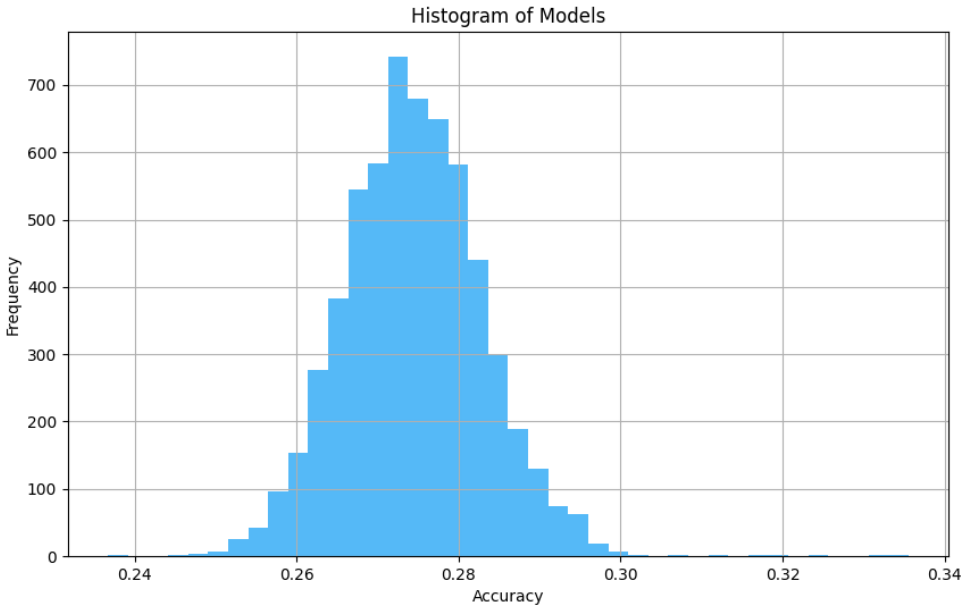

Figure 3.13: Histogram of the Models' accuracy

get the information we needed. We ran it through the latest dataset to calculate the original fitness. Then we made 2 datasets, one with the original data, the original fitness from the output of the execution of the program and the BMI, from one of the crude dataset to the output. The second one is the output, with the new fitness and the new values on the variables. We also removed Spirit, Wine beer, Smoke and Cancer on making the graphics since we considered them redundant.

After that, we also divided each dataset in two parts, one centered on the variables related to the habits and one centered on the consumption. Every data set used to make the heat maps and the Python program can be find in our repository.

In Fig 3.14 we can see the correlation between the BMI, the original fitness of the individuals and the original values of the variables. We can clearly observe that the BMI and the original fitness are inversely proportional which is totally expected as we are maximizing the fitness to reduce the chance to suffer from overweight and obesity, that is, reducing the BMI.

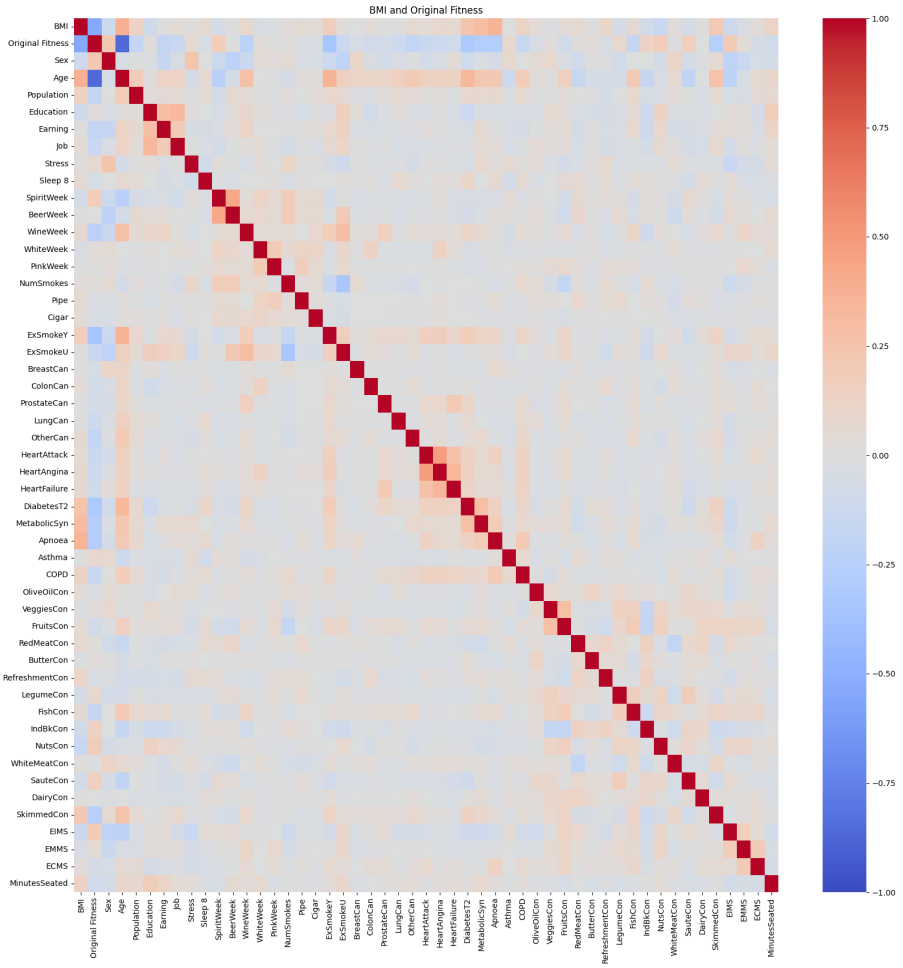

Figure 3.14: Correlation Heat Map From Original Data

We also made 2 more heat maps, Fig 3.15 and Fig 3.16. They are centered on variables associated to habits and variables associated to consumption. We also added the value each box represents for more detail.

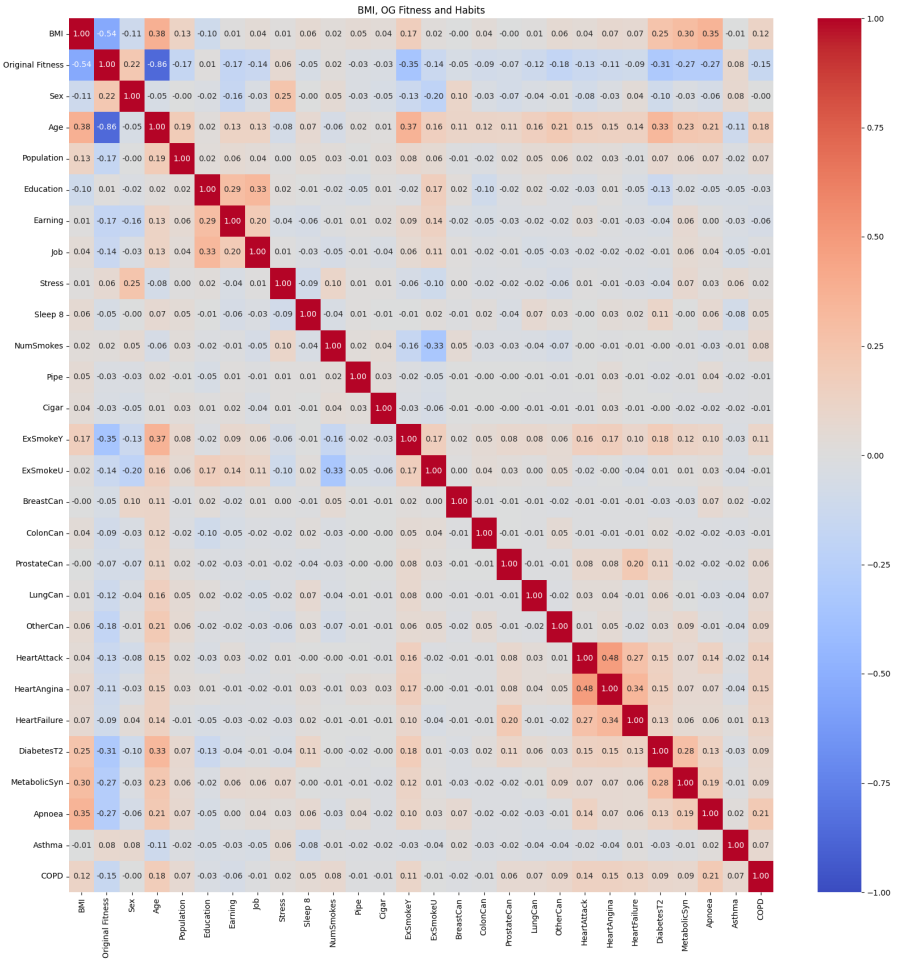

Figure 3.15: Correlation Heat Map with Habits from Original Data

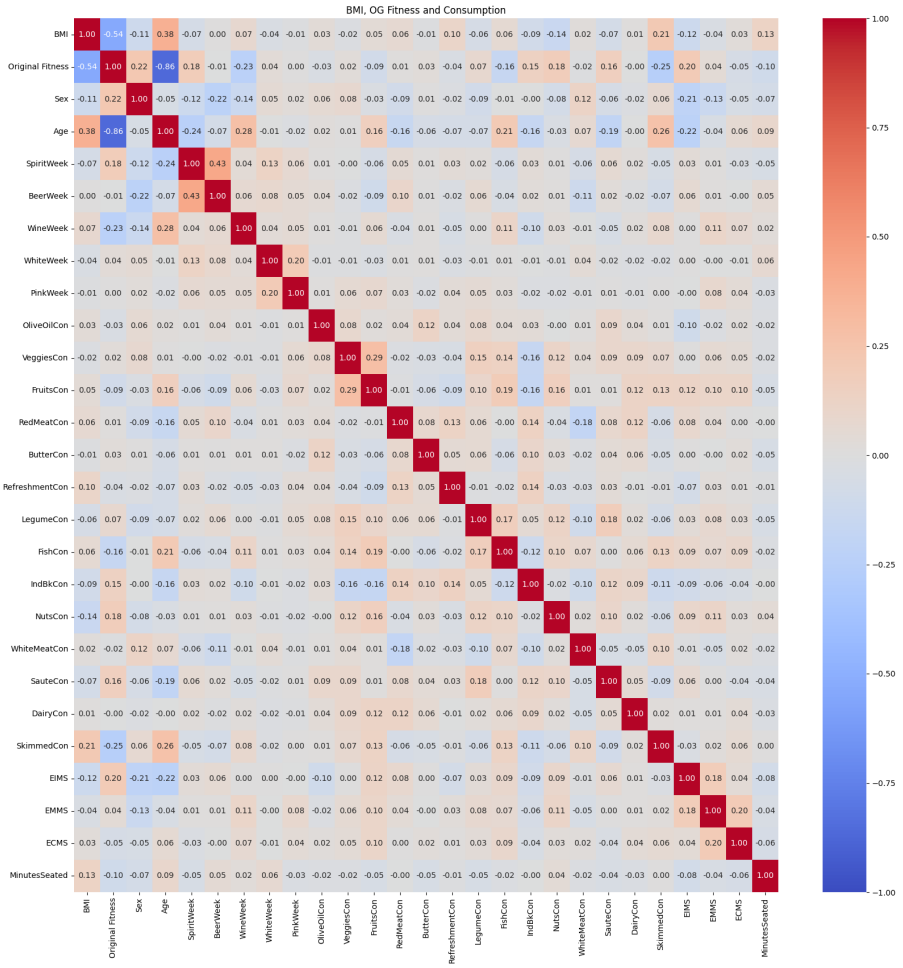

Figure 3.16: Correlation Heat Map with Consumption from Original Data

On the other hand, in Fig 3.17 shows the correlation between the new fitness and the new values of the variables after being improved through our program.

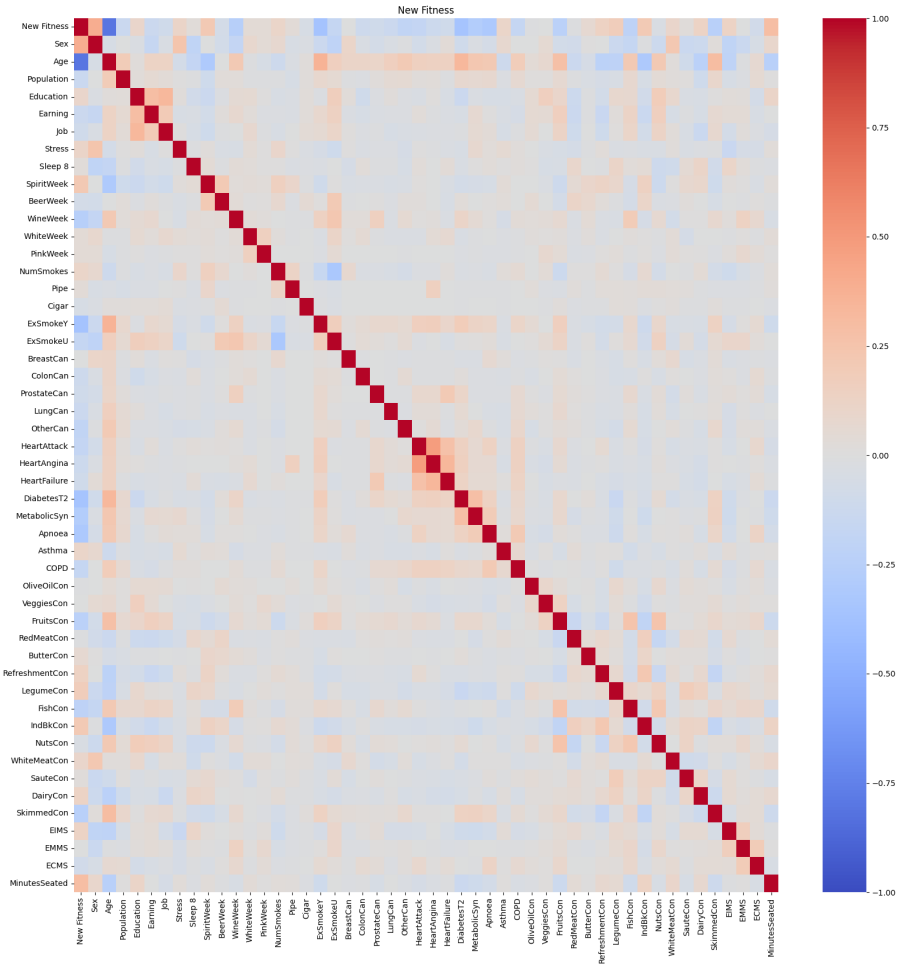

Figure 3.17: Correlation Heat Map After Improvements

We also have the respective maps to habits variables and consumption variables, Fig 3.18 and Fig 3.19 respectively.

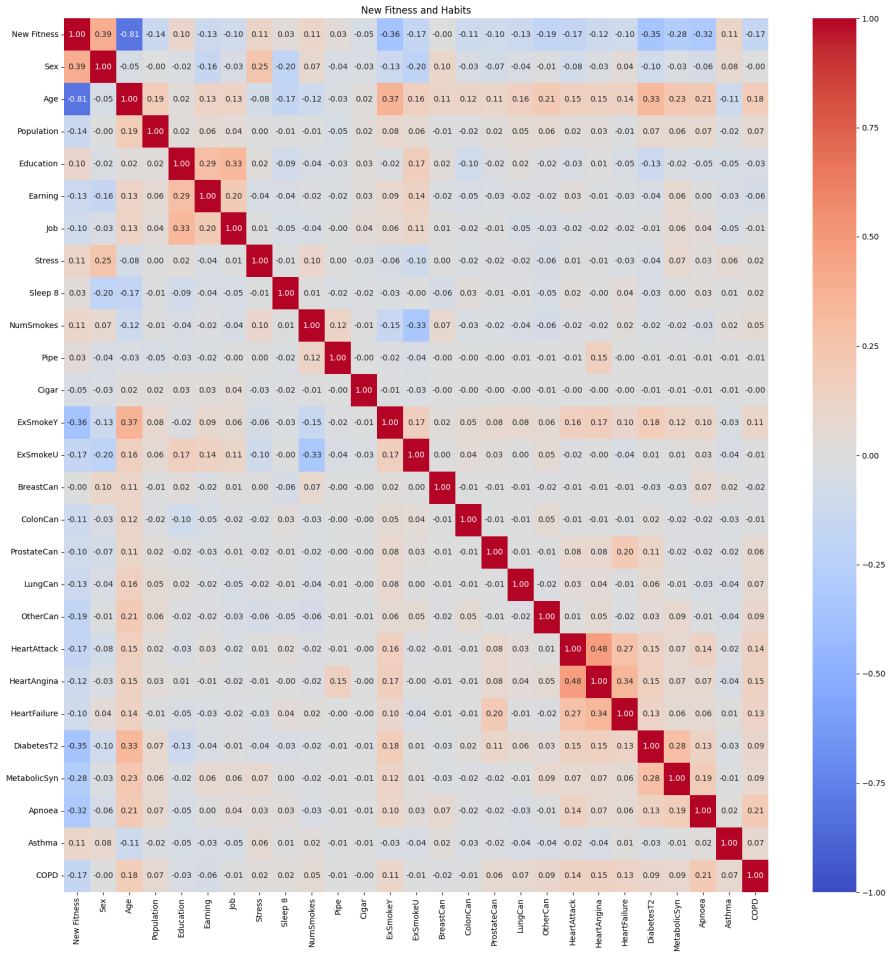

Figure 3.18: Correlation Heat Map with Habits from Original Data

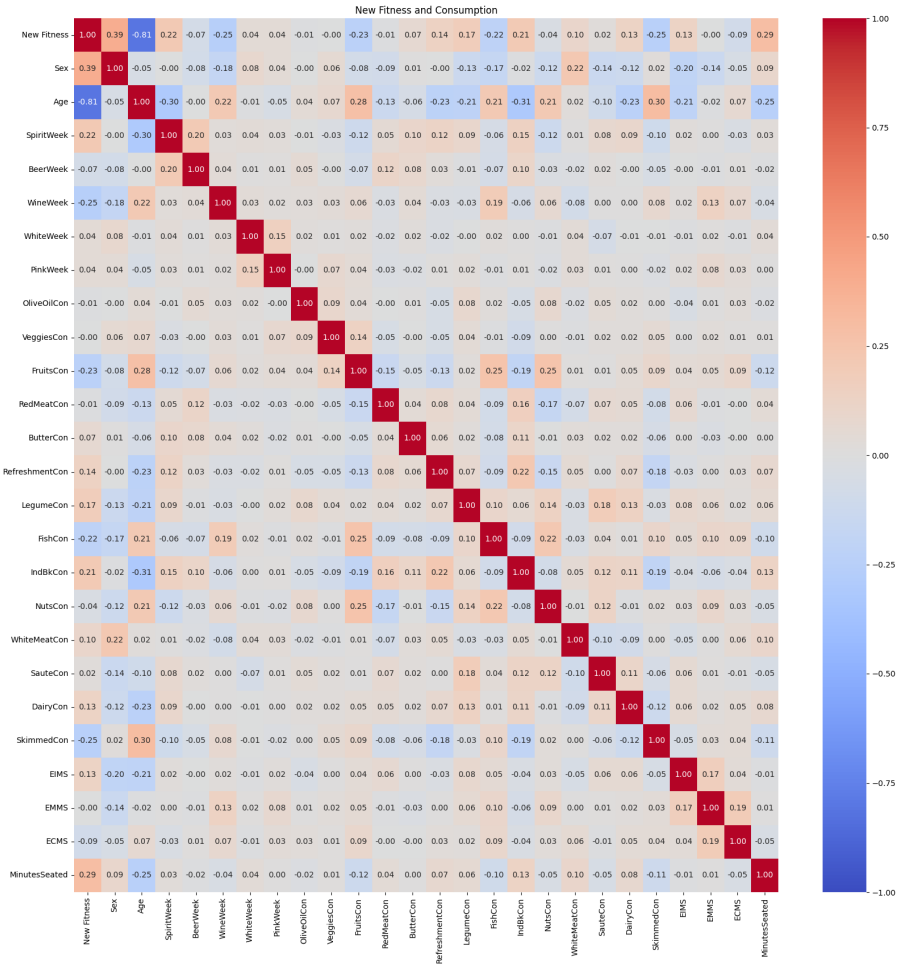

Figure 3.19: Correlation Heat Map with Consumption from Original Data

Chapter 4

Employed Technologies

"The real problem is not whether machines think but whether men do"
— Arthur C. Clarke

We have used many technologies and tools on the development of the project and they will be presented throughout this section.

4.1. Java

We chose Java as the programming language to develop the project. It is a language we are familiar with and it offers numerous libraries and has documentation about everything you can ask for.

4.1.1. Apache Commons JEXL (Java EXpression Language)

JEXL is a library intended to facilitate the implementation of dynamic and scripting features in Java. For more information, check out the documentation.

We utilised this library to transform the models from strings to mathematical equation in our project in order to evaluate the fitness of each individual.

4.1.2. Javax Swing

We used the javax.swing package (documentation) to implement the Graphic User Interface of the application. We already used this package during the degree so we are familiar with it.

4.2. Python

We employed Python to calculate statistical measures because the libraries it offered made the job really easy.

The libraries we used are Pandas (pandas development team (2020)), specialized in management and data analysis, Matplotlib(Hunter (2007)), centered in plot generation and Seaborn (Waskom (2021)), a data visualization library based on Matplotlib.

4.3. Evolutionary Algorithms

The evolutionary algorithms blend random exploration with targeted search through selection mechanism and population transformations with the objective to solve optimization or search problems.

Our objective is to use these algorithms to find improvements on the habits of people. Due to the amount of variables present, we opted to implement a simple and basic evolutionary algorithm.

4.4. GitHub

GitHub is the version control system (VCS) used in this project. We have a repository where all the documentation and data are stored.

In our repository, we divided the 4 different versions we made of the program, each in their own folder: prototype 1, prototype 2, no GUI version and GUI version. The 2 prototypes are not described explicitly but were important in the development process of the habit recommender system. The reason we have a version with an interface and one without it, is because the first one is the version made for the users, and the second one is what we used to generate the data we analyzed, since it can run several individuals.

We also have the original set of data and the PMT environment (the series of files we needed around for it to run), along with bibliography and information files used to store information we needed during the process.

In the analysis folder there are the datasets and the python programs related to the heat maps and the histogram. Demo with the executable and the files needed to run it.

4.5. Pancreas Model Tools

Pancreas Model Tools, also known as PMT, is the tool we used to generate the models that are used to evaluate the habits of the individual. If used as a standalone

application, it allows the user to introduce the parameters of the execution easily, along with importing and editing the grammar from a bnf file and the training data. If used through console it receives a properties file that specifies the parameters of the execution, and the locations of the grammar and training data files. Then it launches the training of the models and stores them on another file if necessary. Hidalgo et al. (2018)

4.6. Secure Shell

Secure Shell (ssh) is an internet protocol used by command line machines to connect between them, and also the tool we used to connect to the virtual machine lent to us, on which we trained the models for our application.

4.7. Tmux

Tmux is a terminal multiplexer, used to get multiple command lines from a single machine. We used it to be able to run multiple instances of the PMT at the same time to make studies on its efficiency and get more models in a fraction of the time. For more information, check out their GitHub repository.

Chapter 5

Conclusions and Future Work

"Being weak is nothing to be ashamed of... Staying weak is"
— Fuegoleon Vermillion

In this chapter we will talk about the conclusions and the improvements and future works that can be done.

5.1. Conclusions

Taking everything into account, the project turn out very interesting. We managed to implement the habit recommender system we had in mind while leveraging evolutionary computation.

This project could pave the way for a myriad of possibilities in the medical field. There are countless options, both for investigation and for practical applications that can take advantage of evolutionary computation.

Some of the limitations we encountered are the following: the data was biased, almost half of the individuals were young people between 18 and 30 years old, due to the lack of medical knowledge on nutrition, the changes we made on the variables were based on common sense and our limited knowledge and our lack of knowledge in the statistic field limited what we could do regarding data collection and display.

We hope this project can create a gateway for others to try and explore other medical fields leveraging evolutionary computation.

5.2. Improvements and Future Work

There is always room for improvement. In this section we will list some improvements that we have considered but could not make.

5.2.1. Redundant variables removal

Spirit, Wine beer and Smoke are redundant and can be removed. If the number of glasses or number of smokes is 0, it means that the individual doesn't drink or smoke. As these variables are implemented so they can never be greater than the original value, there is no need to have these 3 variables in the algorithm. The same can be said for Cancer, as if the individual suffers from cancer, they would have to choose which type of cancer they suffer.

This change implies modifications on both back end and front end. Regarding the back end, new models have to be trained and the Individual class have to be changed. On the front end, we can keep them as a feature to facilitate the user to fill the form but .

5.2.2. Medical and statistic knowledge

As we said before, the limit and conditions we set on the variables on generating random values are based on common sense and our limited knowledge on nutrition, so it would be nice to have feedback and recommendations from an expert on the field. Because something that seems good to us may be actually something bad, or vice versa.

About statistics, if we knew more, we could have extracted more interesting information from the data we produced and draw better conclusions from them.

5.2.3. Inclusion of genes

Regarding the dataset we were provided by GenObIA, there are genetic information as well but we did not make use of them. It would be a nice addition to the algorithm because it can add another depth to the variables but it would be optional because the genes are not information everyone has at hand.

The genes being optional variables implies having different set of models trained with the genes variables and a different chromosome as well. There would be a element similar to a checkbox to notify the algorithm which models and chromosome use.

5.2.4. Friendlier graphical user interface

The GUI we made only has the basic features. It would be great to add more to facilitate the user to fill the form and in general offer a better experience to the user.

A more sequential approach could be taken to the gathering of the data, such as showing different windows (one after the other) for each of the information panels that we have, providing a more interactive experience.

5.2.5. Mobile phone and web application

As for us, we made the program in Java because we were very familiar with it but it would be great to port the program to mobile or web platforms so it could reach and help more people.

Personal Contributions

The contributions each one of us made in this project will be stated below. They will follow a structure similar to chapter 3 to make it easier to keep up.

Fan Ye

In this section we will describe the contributions Fan made in this project. He focused mostly on the algorithm and the analysis of the datasets and the results.

First steps

He fiddled with the PMT, trying the different options this tool offered to compare the results. After deciding on the parameters the PMT would be executed on the ABSyS Group's server, he also worked on the properties file that would be used on said execution.

Variables

Fan worked on every iteration of the variables. He chose which dataset to use, analyzed the results and modified the dataset accordingly.

On the first iteration, he described the variables and added them to the algorithm. Then he analyzed the results and suggested changing datasets.

On the second iteration, Fan took the new dataset and cut off the variables that we would not need and described the new variables. He analyzed the results and suggested the changes made later on the third iteration.

On the third iteration, Fan made the modifications regarding variables of the dataset, normalized its values and described the new variables.

Models Training

Fan worked on the first iteration of the models training. He contributed in the making of the properties file later used on the executions on the ssh server.

Individuals design and implementation

Fan designed and implemented the base of the individuals. He introduced , the auxiliary array and getLimitedRandom(). He also implemented the different constructors the class had, and overrode the compareTo() function. Fan also worked on JEXL and the regular expression that would be used as the fitness evaluator.

As the variables changed in each iteration, he made the modifications necessary on the Individual Class. After analysing the results of each iteration, he has been optimizing getLimitedRandom() the best he could.

Algorithm design and implementation

Fan designed and implemented the different phases of the algorithm: evaluation, selection, crossover, mutation, selection of the elite and introduction of the elite into the population.

He made the adjustments to obtain the time each function of the algorithm takes, the time each execution takes and how much it has improved. He also put the results together and analyzed them.

Graphic User Interface

Fan provided the functional requirements of the GUI and provided support during the implementation, solving questions and bugs.

Other contributions

Fan made the diagrams of the Overview, Fitness Evaluation and Workflow in the Overview section and also the distribution of the variables of each iteration in Variables section in 3.

He also made the python programs and modified the datasets to generate the heat maps of correlation and the histogram.

Regarding the thesis, Fan worked on the structure of the thesis, and the following chapters: 1, 2,4 and 5. As for chapter 3, he worked on everything he contributed, stated in this section. He also made revisions and corrections through all the document.

Daniel Martínez Martínez

Here we describe the contributions made by Daniel in this project, where he mostly delved into the training of the models and the development of the front end.

First steps

In order to start working with the models, Daniel first started tinkering around with PMT, trying to figure out the best properties to get a high accuracy.

Besides that, he also spent some time pondering on how they should make the UI for the application; making a webpage (either on local or on the internet), or using Java's libraries.

Variables

Daniel provided support on the analysis, correct description and interpretation of the variables, which were sometimes mistranslated thus leading to misinterpretations.

Models Training

Daniel worked on training the models for every iteration, trying different sets of parameters and configurations for the PMT.

On each iteration, the models were trained on a virtual machine, and each time some number of variables were changed the models had to be re-trained.

Each set of models had to be trained from a properties file, where the proper values were researched, tried and set through trial and error.

Individuals design and implementation

Daniel suggested using the BigDecimal type at first to store the values of the chromosome in a more precise form, but as it was discovered later, it could not be used because of the lack of traditional arithmetic operating signs needed for the JEXL evaluation.

Algorithm design and implementation

Daniel worked on the connections between the Controller and the different parts on the individual, providing support whenever necessary.

Additionally, the utility classes that transform strings into individuals and read models were made by Daniel.

Graphic User Interface

Daniel worked primarily on the GUI of the program, testing the use of the normal swing utilities, then deciding to opt for the more versatile ConfigPanel, and testing every problem that appeared within.

He also created a class within the ConfigPanel that allowed us to show a legible string value in a JComboBox and provide the back end with its connected integer value.

The execution of the progress bar updates also took an important amount of Daniel's time, as it conflicted with the update of the pressed button "Ejecutar".

Daniel also made the functionality on the MainWindow and AdvicePopUp classes, it's buttons and structure, and looked into the use of a Look&Feel to beautify the front end.

Demo

Daniel made a simple demo demonstrating the capabilities of the front end, what values can be set on each of the parameters and the load process.

Other contributions

On the thesis, Daniel worked on chapters 2, 3 and 4 mainly, also pitching in for chapter 5.

Additionally Daniel looked over the whole document, trying to catch any errors or weird sounding expressions to correct them.

Bibliography

El que lee mucho y anda mucho, ve mucho y sabe mucho.

Miguel de Cervantes Saavedra

Barnett, N. P., Clerkin, E. M., Wood, M., Monti, P. M., O'Leary Tevyaw, T., Corriveau, D., Fingeret, A. y Kahler, C. W. Description and predictors of positive and negative alcohol-related consequences in the first year of college. *Journal of Studies on Alcohol and Drugs*, vol. 75(1), páginas 103–114, 2014.

Dai, X., Gil, G. F., Reitsma, M. B., Ahmad, N. S., Anderson, J. A., Bisignano, C., Carr, S., Feldman, R., Hay, S. I., He, J., Iannucci, V., Lawlor, H. R., Malloy, M. J., Marczak, L. B., McLaughlin, S. A., Morikawa, L., Mullany, E. C., Nicholson, S. I., O'Connell, E. M., Okereke, C., Sorensen, R. J. D., Whisnant, J., Aravkin, A. Y., Zheng, P., Murray, C. J. L. y Gakidou, E. Health effects associated with smoking: a burden of proof study. *Nature Medicine*, vol. 28(10), páginas 2045–2055, 2022. Epub 2022 Oct 10.

Gutiérrez-Gallego, A., Zamorano-León, J. J., Parra-Rodríguez, D., Zekri-Nechar, K., Velasco, J. M., Garnica, , Jiménez-García, R., López-de Andrés, A., Cuadrado-Corrales, N., Carabantes-Alarcón, D., Lahera, V., Martínez-Martínez, C. H. y Hidalgo, J. I. Combination of machine learning techniques to predict overweight/obesity in adults. *Journal of Personalized Medicine*, vol. 14(8), 2024. ISSN 2075-4426.

Hidalgo, J. I., Colmenar, J. M., Velasco, J. M., Kronberger, G., Winkler, S. M., Garnica, O. y Lanchares, J. Identification of models for glucose blood values in diabetics by grammatical evolution. *Handbook of Grammatical Evolution*, páginas 367–393, 2018.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, vol. 9(3), páginas 90–95, 2007.

International Encyclopedia of the Social Sciences. Distribution, normal. International Encyclopedia of the Social Sciences, 2024. Retrieved September 11, 2024 from Encyclopedia.com.

Laing, B. Y., Mangione, C. M., Tseng, C.-H., Leng, M., Vaisberg, E., Mahida, M., Bholat, M., Glazier, E., Morisky, D. E. y Bell, D. S. Effectiveness of a smartphone application for weight loss compared with usual care in overweight primary care patients: a randomized, controlled trial. *Annals of Internal Medicine*, vol. 161(10 Suppl), páginas S5–S12, 2014.

Miller, B. L. y Goldberg, D. E. Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.*, vol. 9(3), 1995.

Nuttall, F. Q. Body mass index: Obesity, bmi, and health. *Nutrition Today*, vol. 50(3), páginas 117–128, 2015.

Ryan, C., O'Neill, M. y Collins, J. Introduction to 20 years of grammatical evolution. *Handbook of grammatical evolution*, páginas 1–21, 2018.

pandas development team, T. pandas-dev/pandas: Pandas. 2020.

Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software*, vol. 6(60), página 3021, 2021.

Wells, G. Correlation coefficient. Encyclopedia of Public Health, 2024. Retrieved September 12, 2024 from Encyclopedia.com.

World Health Organization, W. Obesity and overweight. 2024. Accessed: 2024-09-12.

Mattaku sono toori da. Mattaku motte mu imi da.
Donna ni yume ya kibou wo motte ite mo,
koufuku na jinsei wo okuru koto ga dekita to shite mo,
iwade karada wo uchikuda karete mo, onaji da. Hito wa izure shinu.
Naraba jinsei ni wa imi ga nai no ka?
Somosomo umarete kita koto ni imi wa nakatta no ka?
Shinda nakama mo sou na no ka? Ano heishi-tachi mo mu imi datta no ka?
IYA, CHIGAU!
Ano heishi-tachi ni imi wo ataeru no wa wareware da!
Ano yuukan na shisha! Aware na shisha!
Omou koto ga dekiru no wa... seija de aru wareware da!
Wareware wa koko de shini tsugi no seija ni imi wo takusu! Sore koso yui itsu.
Kono zankoku na sekai ni aragau sube na no da!
HEISHI YO IKARE!
HEISHI YO SAKEBE!
HEISHI YO TATAKAE!

Último discurso de Erwin Smith
Temporada 3 Parte 4 Episodio 4
Shingeki no Kyojin
Hajime Isayama