NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

POSTGRADUATE STUDIES
“DATA SCIENCE AND INFORMATION TECHNOLOGIES”

MASTER THESIS

Wellbeing Recommendations using Genetic Algorithms

Artemis V. Ntountoulakis

Supervisor: Alexandros Ntoulas, Assistant Professor

ATHENS

July 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
“ΕΠΙΣΤΗΜΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΠΛΗΡΟΦΟΡΙΑΣ”

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προτάσεις Ευεξίας χρησιμοποιώντας Γενετικούς Αλγόριθμους

Αρτέμης Β. Ντουντουλάκης

Επιβλέπων: Αλέξανδρος Ντούλας, Επίκουρος Καθηγητής

ATHENS
July 2021

MASTER THESIS

Wellbeing Recommendations using Genetic Algorithms

Artemis V. Ntountoulakis
R.N: DS1190002

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προτάσεις Ευεξίας χρησιμοποιώντας Γενετικούς Αλγόριθμους

Αρτέμης Β. Ντουντουλάκης
Α.Μ: DS1190002

Επιβλέπων: Αλέξανδρος Ντούλας, Επίκουρος Καθηγητής

Εξεταστική Επιτροπή: Αλέξης Δελής, Καθηγητής
Μέμα Ρουσσοπούλου, Καθηγήτρια

ABSTRACT

Recommender systems aim at suggesting one or more specific items to the users, given a specific history of the user and/or some other relevant context. Although this works well in many cases, when we need to recommend bundles of items the problem becomes increasingly complex. Such bundles are, for example, a diet plan or a workout routine for a given time interval. In such cases, we don’t need to recommend one meal or one workout but, instead, we need to optimize for a bundle of items that the user will likely prefer over a given time interval (e.g., one month). In such cases, considering all possible combinations of items is prohibitively expensive. In this thesis, we build a recommender system that provides users with a healthy diet plan containing meals and exercises that covers both their needs and preferences. While the state-of-the-art recommender systems typically use collaborative filtering (e.g., matrix factorization) or/and content-based filtering methods, we use Genetic Algorithms to efficiently explore the space of possible recommended combinations. To this end, we also propose a fitness function that measures how suitable each suggestion is for a specific user. Finally, we performed an experimental study with both synthetic data as well as real users with different preferences, needs and goals in order to fully understand our system’s properties and performance.

SUBJECT AREA: Recommender System

KEYWORDS: recommender system, genetic algorithms, bundle

ΠΕΡΙΛΗΨΗ

Τα συστήματα προτάσεων στοχεύουν στο να προτείνουν ένα ή περισσότερα συγκεκριμένα στοιχεία στους χρήστες, λαμβάνοντας υπόψη το ιστορικό του κάθε χρήστη ή/και πληροφορίες παρόμοιου περιεχομένου. Αν και αυτό λειτουργεί καλά σε πολλές περιπτώσεις, όταν πρέπει να προτείνουμε μια δέσμη αντικειμένων, το πρόβλημα γίνεται αρκετά περίπλοκο. Τέτοιες δέσμες είναι, για παράδειγμα, ένα πρόγραμμα διατροφής ή προπόνησης για ένα δεδομένο χρονικό διάστημα. Σε τέτοιες περιπτώσεις, δεν χρειάζεται να προτείνουμε ένα γεύμα ή μία προπόνηση, αλλά, αντίθετα, πρέπει να βελτιστοποιήσουμε το αποτέλεσμα ώστε να μας προτείνει μια δέσμη στοιχείων που πιθανότατα θα προτιμούσε ο χρήστης σε ένα συγκεκριμένο χρονικό διάστημα (π.χ., ένα μήνα). Σε τέτοιες περιπτώσεις, η λήψη όλων των πιθανών συνδυασμών των στοιχείων είναι απαγορευτικά κοστοβόρα. Σε αυτή τη διπλωματική, φτιάχνουμε ένα σύστημα προτάσεων που παρέχει στους χρήστες ένα πρόγραμμα υγιεινής διατροφής που περιέχει γεύματα και ασκήσεις που καλύπτουν τις ανάγκες και τις προτιμήσεις του κάθε χρήστη. Ενώ τα περισσότερα συστήματα προτάσεων συνήθως χρησιμοποιούν collaborative filtering (π.χ., matrix factorization) ή / και content-based methods, εμείς χρησιμοποιούμε Γενετικούς Αλγορίθμους για να διερευνήσουμε αποτελεσματικά τον σύνολο των πιθανών λύσεων. Για το σκοπό αυτό, προτείνουμε επίσης ένα fitness function που μετρά πόσο κατάλληλη είναι κάθε πρόταση για έναν συγκεκριμένο χρήστη. Τέλος, πραγματοποιήσαμε διάφορα πειράματα τόσο με συνθετικά δεδομένα όσο και με πραγματικούς χρήστες με διαφορετικές προτιμήσεις, ανάγκες και στόχους, προκειμένου να κατανοήσουμε πλήρως τις ιδιότητες και την απόδοση του συστήματός μας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Συστήματα Προτάσεων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: συστήματα προτάσεων, γενετικοί αλγόριθμοι, δέσμη αντικειμένων

To my family

CONTENTS

List of Figures

List of Table

1. INTRODUCTION

1.1 Introduction

Most people are getting into decision dilemmas everyday, about simple stuff, like what to put on for work or what movie they should watch. Before deciding, they look into the available options and sometimes ask a friend to help them decide. However, even friends sometimes can not help them due to the lack of expertise in a certain field. Recommender systems are basically experts that provide one or more recommendations that fits best to each user’s interests or needs. The increasing importance of the Web, led most companies to use a recommender system in their applications. Netflix and YouTube are only some of the companies that provide personalised recommendations in the form of similar videos/items that you may enjoy or users that watched this movie also watched these movies. They have all built a successful recommendation system in order to accurately predict each user's preferences. An accurate prediction by a company, translates to a satisfied user that will engage more with this platform and increase the company's profit.

The general idea of recommendations made by experts, or from a recommender system is not something new. Even when ancient civilizations were around, the king used to take suggestions from his ministers for economic matters or war tactics. While this was not a recommender system, the main idea was the same as several experts provided their recommendation on subjects with more than one possible solution. So even when computers were not around people were always looking for a recommendation from an expert when they were facing a dilemma. But with the Industrial Revolution, the choices in various fields increased and created a need for a system that would choose the product that fits best each user’s needs. Duke university was probably the first university that created an independent research area for recommender systems in the mid 1970s. Also the first recommender system that came into existence was created in the 1990s at the Xerox Palo Alto Research Centre. During that time users received a big number of unnecessary emails and they were having hard time to find the ones that they considered important. Tapestry was created in order to fill this need and sent the unnecessary emails to the spam folder. Using collaborative filtering methods, a mailing list for each user was created that contained users that they know or want to hear from while the others were sent to spam lists. This main idea for this categorization is similar to the one that Google or Yahoo use today in order to filter each user’s inbox. [13]

The recommender systems keep evolving in order to better predict user’s preferences and ask as little information as possible in order to provide personalized recommendations. One of the most famous competitions that involved recommender systems was the Netflix competition in 2006. During that time Netflix was a DVD-rental and video streaming company which initiated a competition that asked its participants to create a recommendation algorithm that is at least 10% from the one they were already using. The winner or the winning team of this competition would receive a prize of 1,000,000 US dollars. BellKor's Pragmatic Chaos team won the competition and the prize, but science also won as new recommendation algorithms were created. Later in Chapter 2 we will refer to this competition and to the latent matrix factorization algorithm that became famous through this competition and was used by Simon Funk [7]. So recommendation algorithms are probably the important thing in recommender systems but later in Chapter 2 we will see that apart from algorithms, human computer interaction is also vitally important. A research that was conducted in the 2000s states the importance of HCI and provides some suggestions to specific companies in order to provide a more friendly environment to users. The significance of this paper comes from the fact that almost all companies followed its author’s suggestions and that these suggestions are followed by all companies today in the form of basic rules for a successful recommender system.

All the companies stated above belong to the Entertainment industry but recommender systems are used in various application areas like Social-networking, E-commerce, Health industry etc. For example, in the entertainment industry i-tunes, Spotify and disney+ are only some of the companies that deploy collaborative filtering and content-based methods in order to provide accurate recommendations. Social networking sites like Facebook or LinkedIn also make users or video recommendations that the target user may know or enjoy watching. [13]. Also due to the considerable amount of clinical data that is collected through the years, recommender systems are used in the Health Industry in order to better support medical suggestions or provide some recommendations. In this thesis we build a recommender system that provides a personalized healthy diet plan. This diet plan takes into consideration each user’s preferences and provides a meal-exercise recommendation that will satisfy the target user.

The structure of this thesis is the following: in Chapter 2 we present preliminaries, basic concepts that will help the reader to better understand this work. Then in Chapter 3 we present related work regarding genetic algorithms and bundles. In chapter 4 we introduce a recommender system that provides a diet plan recommendation based on the user's needs and preferences. In chapter 5 we conduct some experiments in order to better understand the properties of our algorithm and finally in chapter 6 we conduct a user study with 50 volunteers. For this study we create 3 different diet plans for each user and we ask them to rate them based on their preferences. Last but not least we will find out if our diet plans satisfy these volunteers and we will see which plan fits better for each user based on their weight goal and athletic condition.

1.2 Problem Statement

This thesis aims to solve two problems. One problem in the recommender systems area and one in the health area. Recommender systems are famous for suggesting one or more items to users based on users’ history or another relevant context. Although this works well in many cases, when we want to recommend bundles the problem becomes increasingly complex to solve. Such bundles are, for example, a diet plan or one workout for a given week or month. Recommending several meals or exercises for a number of days and taking into consideration several rules in order to keep users excited is not something easy. For instance, it is important to avoid recommending similar foods on consecutive days and also assure that their meals contain all the necessary nutrients in order to keep them healthy in the long run. One way to solve this problem is to consider all possible combinations of items and find the one that ticks the above boxes. However, this procedure is computationally expensive considering all the possible options that can be created. If we are looking for a 10-item combination in 100 items there are 17 trillion possible answers, considering that the position of every item in this combination matters. Different position means that they will eat this food item at different times of the day. However, by using Genetic Algorithms we are able to efficiently explore the space of possible recommended combinations. As explained below through a fitness function and two genetic operators they are able to explore the search space even with a small population. So we also introduce a fitness function that measures each bundle's goodness for a specific user and two genetic operators that through the evolution process take us one step closer to an optimal recommendation. However, there is also a health-related issue that we are also trying to solve.

Nowadays people live in a high-speed world, where work and personal life are often rushed in order to be able to cope with their obligations. Due to these circumstances they tend to consume fast food and avoid doing physical exercises. As a result, more and more people suffer from obesity which comes from the imbalance between energy consumption and energy expenditure. Obesity is a serious problem with a negative impact not only on health conditions but also on social life. In addition, obesity's rates are rising rapidly, so it is important to take responsible actions to prevent the consequences [11]. In this thesis we build a recommender system which is using bundles and genetic algorithms to recommend to users a healthy diet plan. This plan will contain 5 small meals and one exercise per day. However, this plan will not focus only on users that want to lose weight but also on all users despite their target goal. Each user can choose whether they want to lose, maintain or gain weight. Our goal is to build a recommender system that based on their target goal, needs and preferences will recommend a personalized diet plan to each user for one or more days.

1.3 Recommender System

A recommender system (RS) tries to predict the rating for specific items, or suggest possible items that a certain user will enjoy. There are two main categories for recommender systems. The first category is about prediction and we need at least one user that has rated or bought a number of items. Taking into consideration what they (dis)like or what they have bought the recommender system learns their preferences. So using these rated items as a training set, a RS can predict either their ratings on other items or suggest items that they will probably enjoy. For example, in matrix completion problems each cell is a different item and each column is a different item category. Then by combining user's ratings and items information the RS is able to provide recommendations that they will probably rate positively. The second problem that a recommender system can solve is the so-called top-k recommendation problem.

This is a similar problem but instead of trying to predict an exact value for each item, it recommends to users the top k (k is an integer) item that they will most likely prefer. All recommender systems have some things in common which is to follow certain rules in order to achieve their goal.

These systems are designed for one main purpose: to get our attention. By getting our attention we either watch more videos or buying more products and increasing each company's profit. But beside the business value of these methods there are also some technical rules that they should follow in order to be successful. First and foremost a recommender system should provide users with items relevant to the ones they like. Relevance is the first thing that comes into our mind when we think about these models. Most of us do not have very complicated preferences. We watch two or three different types of movies or videos. So, providing relevant items is the safest way for a recommender system in order to find at least one item that the user will certainly enjoy. For instance, when the user watches a significant amount of action movies or movies that Brad Pitt starred in, a recommendation could be "Once Upon A Time...In Hollywood" which tick both boxes. On the other hand, "The Proposal" is a romantic comedy and is certainly not relevant to the movies the user watches. However, could this irrelevant recommendation be something that a user will enjoy?

The answer is yes because relevance is not the only factor that these systems should take into consideration. It is also important to suggest items that this user has not watched or bought before. Especially in recommendations for new buys, it is important to not recommend items similar to the ones they already have. For example Amazon should not recommend to users a table if they have already bought one from their site some days ago. It should recommend a few chairs or a couch that matches with this table. Considering that couches and tables both belong to the living room class it is important to suggest items from the same class but different category. This does not apply only to furniture but also to movies and songs. For instance a user that enjoys action comedy movies (category) would probably prefer an action superhero movie. Both categories belong to the Action class. However, seeing only the top sellers or things the user has already bought or watched will probably provoke users to check for a different source of information. So it is easy to understand that there is a need for novel suggestions. But what is a novel recommendation? Based on [16], "novel recommendations are recommendations for items that the user did not know about". A novel recommendation is about an item that target users have certainly not watched (or bought), and sometimes even not heard of. So it is expected that the public is not aware of these items but they have high changes to satisfy the user. This type of recommendation has higher complexity than the ones we discussed above. Liang Zhang [16] pointed out three important factors that make it difficult to find a general novel recommendation. A movie may be popular in a certain year (year of release) or a certain season (Christmas, The Grinch) which is mentioned as a time effect of popularity. And the other two are based on the user's characteristics like sensitivity to popular and general activity. If the target user does not follow trends then we could suggest to them new movies that most people enjoy even if it does not belong to the class that they frequently watch.

It is also equally important to provide the user with unexpected suggestions that will surprise them in a positive way. For instance there are users who watch only science fiction movies. However the algorithm could recommend to them a new, well-structured crime - mystery movie. Whiley they may, at first, dislike the suggestion, they will definitely think about trying something new. Using this technique in the long term will probably work very well to most users and they may discover new interests and hobbies. So it is important to suggest products of different types and different backgrounds. If all products are similar then there is a chance to search for another recommendation engine. Diversity increases the possibility of providing at least one extremely good suggestion and also enhances users' attachment to the company's product because they are not seeing repeated recommendations of similar items[2]. An interesting research took place in 2001 by Kirsten Swearingen and Rashmi Sinha. They compared six recommender systems (3 for movies and 3 for books) and while it may seem outdated, as we will see a lot of their conclusions are considered as basic rules in today’s RS.

Lots of companies used to think that algorithm’s accuracy is the most important factor in a RS and that's why a lot of companies, in the past, used to invest a lot in evolving their algorithms while neglecting to evolve human-computer interaction. But what exactly do we mean by saying human-computer interaction? It is basically the communication or the interface between humans and computers. The user's interaction with the RS is divided into two categories: input and output. Input part consists of the number of rated items or time spent on watching a video. The output part contains information provided by the system like: number of recommendations, information regarding each recommendation or even the logic behind each recommendation. In this experiment 19 participants evaluated recommendations made by 3 of their friends and recommendations made by either 3 book or 3 movie systems. They evaluated both input and output elements of each system. In the input part they evaluated the number of items to be rated, information about these items, rating scales and filtering by Genre. Moreover, they evaluated system outputs like: accuracy of algorithm, recommendation of popular and unpopular items, new-unexpected items, interface issues and others. We will skip directly to the results and conclusions of this research [14].

Firstly, they concluded that the number of items to be rated did not seem to play a vital role in choosing the best recommender system. Nowadays most applications ask new users to rate only a few items of their liking (Netflix) or they do not ask for a rating at all (Instagram, Facebook) but due to their well-tuned algorithms and lots of data they predict extremely well our taste on their offering items. These accurate predictions are what make these applications so successful. This success can be measured by the number of users, time spent by each user and of course their revenue. However when the platform asks users to rate some items it is important that they can easily understand the rating method. There is not a general rule about rating methods. Some users may prefer a binary rating and some others a continuous scale rating, both of them can be easily understood. YouTube, Netflix have very simple rating methods with a like/dislike button as discrete scales and Amazon, TripAdvisor and others have also simple continuous rating methods by rating from 1 to 5 stars.

In order to evaluate algorithms accuracy, it is important to explain the reason for this recommendation. And by saying reason there is a certain phrase that we will probably find familiar: “more like this”. This is something that we see very often today in lots of applications like Amazon, Netflix, YouTube and others. That way they make it easier for the user to understand what to expect from this suggestion as it points to the item that they have already watched or liked. However, this information is not enough for users to try this suggested item. They would also like to see the item's description, like the genre it belongs to, something that it was not in all the RS that they evaluated. Furthermore, they would like to see new items that belong within their favorite movie or book category but they also need to see recommendations from different categories in order to broaden their horizon. Do all of the above seem familiar? They are basic rules that almost all of today's RS follows. While these rules seem obvious to today's readers, they were pretty important observations made in this paper that time. Important enough that 3 out of 5 evaluated RS sites changed their interfaces in order to provide a better and more user-friendly recommendation. One of the sites that made these adjustments was, well-known to all, Amazon.

Most base models of RS systems work with data coming from two sources. Either user-item interactions (ratings, time watching a video) or properties (features) of the users. There are several types of recommender systems and here we will talk about two of the biggest: Collaborative filtering and content-based methods. Content based methods focus on user item interactions while collaborative filtering methods user-user, item-item and user-item interactions in order to provide a more accurate suggestion. The second difference between these methods is that collaborative filtering methods use information of all users in order to provide a suggestion while content-based methods need only target user’s data. Nowadays most companies use systems that use both data types in order to get better results. These systems are called hybrid systems and their goal is to keep each method's advantages and provide accurate suggestions [2].

1.3.1 Content-Based Methods

Content based methods make suggestions by using keywords or attributes assigned to each item. They use all items’ properties but not all users’ properties. They need information only for the target user. For example, if we want to suggest a new movie to John then we only need John's movie ratings and not all users’ ratings. So these methods focus on data from two different sources: target users’ preferences and items’ attributes. Nowadays most items have lots of attributes that characterize them. For instance, movies provide us with a lot of information like genre, actors, directors and description. So, if John rated positively a lot of action movies then we do not need George's ratings in order to understand that John will probably want to see a new action movie. This method is highly effective on new items that have a few ratings. User's profile is made by the user's feedback on several items. This impact can be either explicit or implicit. Implicit feedback is when users’ preferences can be obtained without asking them any questions. For instance, people that watch football highlights of a certain match would probably enjoy watching highlights of several other matches. These systems work very well in text-rich domains or with items that have lots of features. For example it can be a useful tool in web page recommendation. Web pages usually contain text, images and sometimes even music. This type of information is called unstructured. These data types are not in specific columns or categories and most of the time they contain more information than the structured ones. More information needs more time in order to be processed and to get them in a structured state in order to be useful. Structured information is obtained from a database or a (csv, parquet etc.) file where every feature is in a specific column and every item has its own row. Providing any of these two data types is essential in order to train a model and be able to make suggestions [2].

There are 3 main steps in order to build an accurate content-based model. Firstly, it is necessary to find data and convert them into a "keyword-based vector-space representation"[2]. Secondly, we need to obtain each user's preferences. These preferences are either their ratings on different items, or items that they previously bought, watched or read. As said above both implicit and explicit feedback can be used in this model if they are pre-processed correctly. Combining information that was obtained from users and items then we are able to train a content-based model in the same way we would have trained a classification or regression model. These steps are done offline due to their time complexity. Finally, by using this pre-trained model we can recommend to users’ specific items based on their preferences.

Another big advantage of these methods is that they explain the reason why they are able to provide users with each specific recommendation as it uses item-item similarity methods in order to provide a recommendation [2]. The above experiment made by Kirsten Swearingen and Rashmi Sinha [14] showed the importance of gaining the user's trust. Users can find suggestions almost anywhere but they will not know if that's a good or a bad suggestion until they see the suggested movie or read the suggested book. However, users do not want to spend time watching something that they may not enjoy. Explaining the reason behind a recommendation can probably satisfy their curiosity and help them decide if they want to watch this movie or not. Suppose that "The Lord of the Rings" has 80% similarity with "The Hobbit" due to the actors and director. Mary has rated with 5 stars the first trilogy and now is looking for a recommendation. The Recommendation system will probably suggest them watching "The Hobbit" and could also explain this suggestion with the phrase "movies like The Lord of the Rings". This phrase in addition with movie description, actors and directors is a well-structured suggestion as stated in [14].

1.3.2 Collaborative Filtering

As stated above, content-based methods need each item description and ratings of the target user in order to predict what they may like. On the other hand, collaborative filtering methods gather information from many (or all) users in order to suggest to target users a number of new items (collaborating). While content-based methods use item attributes to make a prediction, collaborative filtering methods sometimes even ignore this information. They could use only the correlations in the ratings patterns across users in order to make a prediction [2]. Collaborative filtering methods have two different approaches: Neighborhood-based (memory-based) and model-based.

Memory-based Algorithms

While memory-based algorithms, or as otherwise called neighbourhood-based collaborative filtering algorithms, are the earliest developed algorithms for recommendation, they are still useful in a variety of applications. They are called neighborhood-based because in order to make a prediction they have to use information from items or users similar to a specific item or user. A famous classification neighborhood algorithm that we all probably know is k-means algorithm. It finds the nearest k (integer) items in order to predict in which class the target item belongs to. There are two types of neighborhood-based algorithms in recommender systems: user-based and item-based. Let's say that a company wants to suggest to users a new set of movies. If they use a user-based collaborative filtering model then by using ratings of similar users they suggest a weighted average value of these ratings for each item. Similar users have similar preferences with target users meaning similar ratings in most movies. Using users' ratings and a similarity metric like cosine similarity, we could find users similar to target users. On the other hand, in item-based they first find the k (integer) most similar items to the target item. Similarly, here any similarity metric or dissimilarity metric can be used like Euclidean or Manhattan distance in order to find those items. After having found how similar an item is to the target item, the system could use a weighted average (based on item similarity) of those ratings as a prediction for a new item [2]. After stating the above methods, the first question that comes to my mind is which method is better and why?

In general item-based methods provide more accurate recommendations to users. The reason is that it takes into consideration only the target's user preferences and uses similar items in order to give a weighted average rating. For example, if the target user rates highly action movies, then an item-based model will probably suggest another action movie to this user. On the other hand, user-based methods find similar (to target user) users and suggest movies that they have seen. Although they have some things in common, it is not necessary that they like exactly the same things. For instance, they both rated several action movies positively while target user's neighbours also like romantic comedies. That does not necessarily mean that the target user also likes this movie genre. In addition, item-based methods provide an explanation for each recommendation. Due to item similarity a reasonable explanation could be: "because you watched movie x these are your recommendations". However, providing similar information is not possible when user-based methods are used. Due to privacy concerns another user's information can not be provided to the target user. So, it's irrelevant for a target user information like: "User1 likes this movie" without mentioning User1's name. Last but not least item-based systems are less vulnerable to changes in ratings. The number of users is way larger than the number of movies. As a result, a movie will probably have lots of ratings and a few more from new users will not change its rating dramatically. On the other hand, a user may start watching a new genre of movies so the result from the function that measures similarity would change in a significant way. Another reason is computation frequency. Due to newcomers and change in tastes, in user-based models, we have to recompute user-user similarity quite frequently. On the other hand, new users will not drastically affect the already existing movies that have, most of the time, a large number of ratings [2].

While item-based methods seem to provide better results, this is not completely true. Item-based recommendations are similar to each other and sometimes are expected by users. If we are looking to diversify movie genres, actors and other movie attributes in a recommendation list we definitely have to use user-based methods. As said above, novelty is important in order to keep users excited for a new list of recommendations. Suggesting movies taken by users with similar tastes (neighbours of target user) will probably provide to target user some unexpected movie suggestions. Are they going to like them all? Probably not but they will definitely think of trying something new and will maybe ask for more similar suggestions due to curiosity[2].

In conclusion, memory-based algorithms are simple and easy to implement. They also provide an explanation for each suggestion and they work efficiently even when more users and/or ratings are added to the system. Also combining user and item-based methods will provide some expected but also accurate predictions and some that are novel to target users and could satisfy their needs. These methods' main disadvantage is the way they handle data sparsity. Some movies have only a few ratings or some users have rated a few movies. For instance, there is no neighbor user that has rated "Goodfellas" then it can’t be suggested to the target user. If a movie is not suggested to the target user, then either is not a good suggestion for this user or it is just not having enough ratings. So, these methods do not handle unpopular or new movies.

Model-based Algorithms

As stated, neighborhoods-based methods were the earliest methods used in recommendation systems. However, as machine learning was becoming more and more popular, methods that were popular for classification or regression, also used in recommender systems. Examples of popular classification methods that contribute to RS are decision trees, rule-based methods and Bayes classifiers. Neural networks, support vector machines etc. are also used which are popular regression methods [2]. Lots of people may ask how these methods work then?

Basic rules of those methods are similar. At first data has to be pre-processed and cleaned in order to transform them in a structured form. In order to test a model's accuracy, it’s important to split the above data into train and test set, and choose an appropriate model to fit training data. After the model has been trained, we proceed to the prediction phase. It is important to find the correct values for the model’s parameters in order to get a low training error. The goal is to have small training errors and avoid over-fitting. In the prediction phase the model either predicts ratings for movies that exist in the test set or directly suggests k items to users. Using these ratings (or suggestions) and the test set we are able to calculate prediction error and if this error is smaller than a certain threshold then a model that recommends suitable items to users is found. However, if error is larger than this threshold, we have to either adjust model parameters in order to find the optimal solution, or repeat the whole process from start. It is important to be able to understand what went wrong and the appropriate result wasn’t reached. This low prediction error could be due to several reasons like wrong model choice or model over fitting. Solutions for problems like the ones stated above is choosing a different method, training less our model or with different parameters and even trying a different training set (k-fold cross validation).

In order to understand better model-based algorithms we will briefly discuss decision trees, which is a known classification model, and about latent factor models. Latent factor models are considered as state of the art and became popular through the Netflix contest. While we probably know the way decision trees operate in classification tasks, there are a few possible problems that may need to be overcome in order to function properly in recommendation tasks. For example, it is highly possible that several users would not have rated all available and as a result rating matrix contain null values. Is there a way to deal with null values? Also, what variable does this system try to predict? [2]

In decision trees, if N is the number of items (movies) then we have to create N decision trees, one for each item. So, by creating N independent trees we have to predict each user rating for each movie. We provide to our system information about its users and movies as attributes in order to train our model and be able to predict users' ratings. After collecting our data, we have to decide which attributes correspond as dependent or independent variables. Dependent variable or variables are the ones we are trying to predict while independent variables are the ones we manipulate or change in order to achieve this. But what happens if one or more independent variables contain null values? In order to address the missing values problem, we have to use dimensionality reduction techniques. Supposing that we have M users, N movies and we want to predict the rating of movies in column j. Rating vector's shape is $M * N$and by using a dimensionality reduction technique, like PCA, we create a vector $M * D$ where *D<<N-1* and all attributes as specified. So, we use this new rating vector, which does not contain null values, in order to construct a fully functional decision tree and find the rating of this specific movie in column j. So, by reducing the dimensionality of these vectors we transform this from a recommendation problem into a classification or regression task. In order to predict ratings of all N movies we have to repeat this process N times [2].

However, there is a faster and more accurate method that became famous through the Netflix competition and it's called Latent Matrix Factorization. While previous RS methods used dimensionality reduction techniques in order to fill null values and create a more convenient data representation, latent matrix factorization uses them to directly estimate the data matrix at once. We will briefly explain this method by discussing the way Simon Funk used this method in the Netflix competition [7]. So the participants of this competition had ratings for 17,000 movies from 500,000 users. So while there is a rating matrix with 8.5 billion entries, most of them are blank. As he states this matrix contains only 100 million ratings and 8.4 billion null values. So roughly 2% of the rating matrix contains information while the rest 98% is blank. This creates a need for a connection between these movies and all those users. He states that meaningful generalities can help in representing data with less information as expected. There are some movie's properties that can be generalized. For example, how much action, romance or crime investigation scenes a movie contains and for users if they like action, romance or crime investigation movies. Supposing that "Rocky" is rated as action = 1.2, romance = 0.1 and crime = 0.01 and John's preferences are action = 2, romance = 4, and crime = 1 then combining this information we get$1.2 \cdot 2 + 0.1 \cdot 4 + 0{,}01 \cdot 1 + \ldots = 3.5$ which could be a rating prediction. These attributes can be something meaningful like movie genres or something that does not make sense to us humans but provide more information to our model and in overall better results. Suppose that each movie is described by 40 values explaining how much each movie satisfies each aspect and each user has 40 values that explain how much he likes each aspect. Then the original matrix has been decomposed in two matrices $17{,}000 \times 40$, call this P, for movies and $500{,}000 \times 40$, call this Q, for users. Multiplying these two matrices will bring back our original matrix $17{,}000 \times 500{,}000$. This is called singular value decomposition (SVD) and it makes calculations simpler by getting meaningful insights about users and movies. So, in order now to get the values of these two matrices we either initialized them randomly or by using a Gaussian distribution. Then we can optimize their values by using Stochastic Gradient Descent. When we have the final two matrices by taking the inner product of 1 user (1 line of Q) and 1 movie (1 line of P) we can predict the rating of this user in this specific movie. So, this simple computationally algorithm provided exceptionally good results and was ranked #3 at one point in this competition [7]. So while these models became famous in 2006, now they are considered state of the art in RS.

1.3.3 Content-based or Collaborative Filtering

After presenting both methods, there is one question in our mind: Which method is better? Both methods present significant advantages and disadvantages. Collaborative filtering systems have difficulty recommending new items to users that have a few to none ratings. While content-based methods do not take into consideration ratings from other users and are focusing on item's attributes. So, they can easily recommend a new item to an already existing user. In addition, as stated above, content-based methods provide meaningful explanation where in many cases memory-based algorithms cannot. Moreover content-based methods are easier to use and need less computation time due to using one user's information. On the other hand, collaborative filtering methods need both user's and item's information in order to recommend an item. In addition, collaborative filtering methods are vulnerable to changes as more and more users are registered to the platform and adding new ratings. Due to those changes, it is essential to re-train their model offline in order to use the added ratings and provide better prediction. On the flip side, due to collaborative filtering methods properties less re-training needed as having a few more ratings for each user will not dramatically change their possible suggestions. [2]

Collaborative filtering methods have also some advantages versus content-based systems. At first content-based methods tend to recommend items that are similar to each other. Due to item similarity all recommended items will probably belong to the same category or genre than the ones they have already rated, watched or searched. However collaborative filtering methods provide novelty recommendations as they take into consideration the preferences of similar users. These recommendations will probably surprise positively target users and will provide them greater satisfaction than getting expected recommendations. Furthermore, collaborative filtering methods handle bigger amounts of unstructured data while content-based methods need to pre-process these data in order to convert them to an appropriate structured form. [2]

By combining content-based methods with collaborative filtering RS a hybrid model will be created. The purpose of this union is to keep each method's advantages and create a model that will avoid overfitting, avoid cold start problems and provide efficient and meaningful recommendations. Using both methods as an important component of a hybrid model will provide efficient suggestions and reduce computational cost.

1.4 Genetic Algorithms

Genetic algorithms (GAs) are numerical optimization algorithms inspired by Charles Darwin's theory of natural evolution. Darwin's theory states that all organisms develop through natural selection mechanisms that increase their ability to survive and reproduce. In genetic algorithms a population of individuals are evolved through a number of generations in order to minimize or maximize a fitness function. The individual with the largest or smallest fitness, depending on the problem, is the solution to the stated problem. Like chromosomes in organizations each individual has a set of features that can be mutated or changed by genetic algorithm operations. These are very general methods and have been used in a wide range of problems through different scientific fields like science, engineering, social sciences etc. These algorithms are pretty simple to understand and also easy to implement due to the lack of complex mathematical operations and heavy computations [5]. Based on [3] we will use these operations in order to help the users maintain a healthy lifestyle through meal and exercise recommendations.

Using Genetic Algorithms for solving mathematical problems is not something new. In the 1960s John Holland invented genetic algorithms and since then several people have used them in multiple fields. But how do we define a GA? A genetic algorithm consists of 4 basic components. A number of possible recommendations which are called population and each one of them separately is called individual. It also contains a function that defines how good or bad an individual is based on user needs and preferences. The last component that we need in order to design a system that uses genetic algorithms is genetic operators. We will use two genetic operators and they are the ones that transform each individual’s features in order to evolve through generations. The first genetic operator is called crossover and it mixes two individuals in order to create a new one. The other one is called mutation and needs one individual to happen and it basically alters its features. Both operators are used in order to provide a wider search space and diversify problem's possible solutions. As a result, they increase the probability to find the optimal solution or at least a better solution than the ones already existing in the population [4].

1.4.1 Search Space

In numerical search optimization problems search space can be of infinite length. The number of dimensions is defined by the number of attributes that each item has. The point in the search space that has the largest value of a function is called global maximum. In maximization problems the goal is to either find the local or global maximum. When it is difficult to find the global maximum, the evolution could stop if a value larger than a certain threshold has been found. Similarly in minimization problems the goal is to find the global minimum which is the smallest value in the search space. By saying search space, we mean all the possible solutions to a specific problem. For example, in IMDB the search space is all the available movies in the platform. It is easy to find the movie with the highest rating but how do we deal with a similar problem, but this time with more dimensions (beside rating)? A problem with more than one dimension would be to recommend a movie for a certain user. Users are complicated individuals and apart from movie ratings it's important to take into consideration their favourite movie genre, actors etc. While in simple problems like the one that was discussed before the search space is constant, this is not the case in GA problems. GA are capable of solving complex problems and dealing with individuals with more than one dimension. Choosing the individual with the smallest value in each dimension does not necessarily mean that it is the optimal solution. So genetic algorithms are able to diversify the population and avoid “stucking” in a local minimum away from the optimal solution. While genetic operators are able to diversify the population, it is equally important to have a general initial population. There are several ways to initialize a population. Creating enough items in order to cover the whole search space assures us that we have a diversified search space. However, in large data sets this could be impossible due to memory limitations [1]. Random initializations avoid memory limitations and provide a diversified population without getting affected by external forces (like author's preferences). After initializing each individual, we have to create a fitness function in order to be able to search for the optimal recommendation.

1.4.2 Fitness Function

A fitness function takes as input a possible solution to a problem and returns as output a score that explains how close this solution is to an optimal one. It is a way to measure how "good" or "bad" an individual is for a specific problem. Every genetic algorithm problem is either a minimization or maximization problem. The individual with the largest or smallest fitness is the solution to the problem. A good fitness function should be complex enough in order to provide an accurate answer and simple enough in order to run in a small amount of time. Even a few seconds per iteration are important due to the fact that this function iterates over $N \cdot generations \cdot iterations$, where N is the number of individuals, which in most recommendations is a large value. GA are mainly used in situations in which either the mathematical functions needed for solving a specific task are very complex or when a direct mathematical approach does not even exist. So it is important to create a function with the correct parameters in order to provide an efficient and accurate result [1]. However, all parameters do not carry the same significance, and it is important to find the correct weight for each parameter. For example John and George want to find the movie to watch tonight. John takes into serious consideration movie ratings while George chooses a movie based on its actors. So our parameters are actors and movie ratings. If we had to solve John problems we had to add larger weight to rating parameters than actor parameters and the opposite for George. Each person adds a different weight in each parameter. As a result, weight settings should be taken into consideration when a fitness function is created. A correct fitness function should be able to satisfy a large number of users by providing them an accurate recommendation that fits their preferences.

1.4.3 Crossover

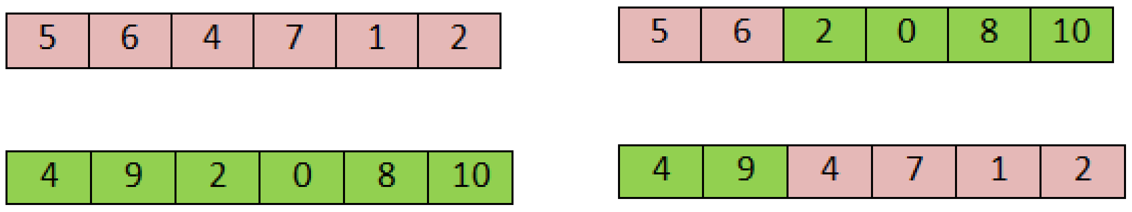


Figure 1: One point crossover. A random index is selected, and these two items exchange their values after this index

There are two main genetic operators that are used in order to diversify the population: crossover and mutation. Crossover needs two individuals to take place while mutation needs one. Crossover or recombination is a genetic operator that combines information from two items in order to create a new one. There are several crossover algorithms; some simple and some a bit more complex. At one point crossover, a random crossover point is selected and the two parents change tales in order to create two new items (Figure 1). Similar to this we have multi point crossover, in which several points are selected (two or more) and change their position. For instance, if two points are selected the two parents exchange tails, heads and also the values between the two points respectively. In uniform crossover each gene or feature is treated separately and it is randomly chosen if two genes will change place or not. Of course the probability of this happening or not, does not have to be fifty-fifty and it can be more biased towards change or stability. There are also more complicated algorithms that are used in crossover but we do not cover them in the scope of this analysis.

1.4.4 Mutation

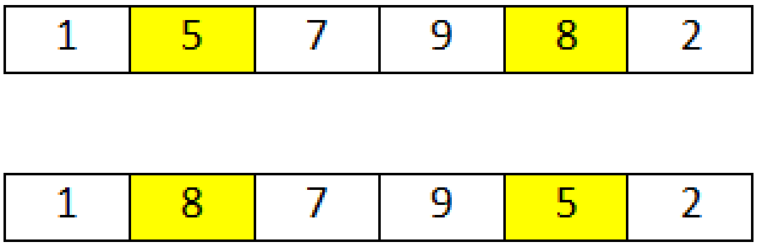


Figure 2: Swap mutation. At random the second and the fifth feature of this individual interchange values.

Mutation can be defined as a small change in an individual in order to get a possible new solution. If we think of an individual as a binary vector, every now and then one bit position in some individuals will change from 0 to 1 or vice versa. The probability as said in [4] is problem dependent however a typical value is 1/N where N is equal to population size. Here we will present three pretty famous mutation operators; bit flip mutation, swap mutation and scramble mutation. Bit flip mutation is similar to what we did in uniform crossover, meaning that one random bit or feature of the individual is selected and changes value. In swap mutation two random positions from an individual are selected and interchange their values as seen in Figure 2. Lastly, scramble mutation is the generalization of the previous method meaning that a subset or the entire individual is chosen and its values are randomly shuffled [1]. As said above in our implementation mutation needs only one item in order to take place. As a result, we will use the swap mutation similar to the one [3] did.

1.4.5 Individual Selection

In a crossover algorithm, individual selection is an important process that defines which two individuals will take place in this operation. In [3] they select two individuals randomly. All individuals have the same probability to be selected in order to participate in this operation. This general method assures that we will avoid premature convergence as we still expand our search space. However, individuals with low and high fitness value have the same probability to evolve. Picking randomly many “bad” individuals for crossover operation will lead to needing way more generations than expected in order to find the optimal solution. Also, this is not how crossover in nature works, in nature only the fittest survive. So, in genetic algorithms there should be a similar selection in which only the best individuals will evolve through generations. A simple method would be to keep the best 50% of the population.

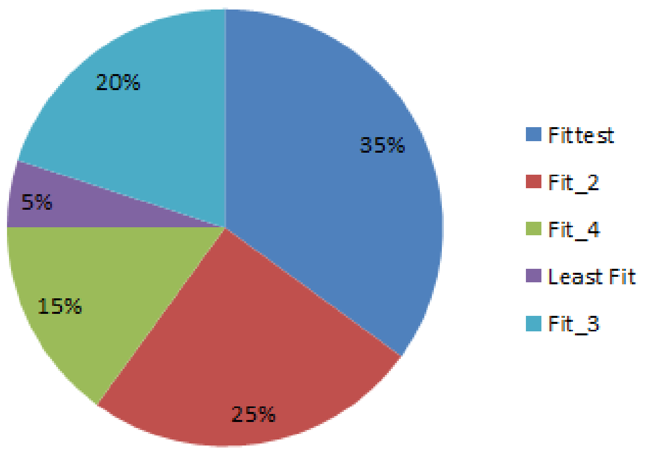


Figure 3: The fittest individual (blue) has the larger probability to be selected (35% in this occasion) and the least fit the lowest. This selection operation is called roulette selection or fitness proportional operator

This is probably the easiest method to implement, however it does not distinguish the individuals between good, better and best and reduce by a lot the genetic adversity of the population. In addition, this method is not similar to Darwin's theory of natural selection. Fitness proportional selection or as otherwise called roulette wheel selection is a known method for selecting individuals that does not create the above problems. Using this method each individual's probability to be selected is proportional to its fitness value. Meaning that the better the solution, the more probable this individual is to be selected to pass its features to the next generation. Imagine this as a big roulette where the fittest individual covers a larger portion of the roulette than the others. As seen in Figure 3 the probability of the ball to fall into the blue segment is larger than a segment that represents the least fit individual (purple). In order to implement this operator, we add the fitness value of each individual and a random number gets chosen between 0 and SUM(Fitness). In order to find the selected individual, each population member's fitness is added until the sum is greater from the random number. The last individual is the one selected from this method. This selection happens at least twice in order to have two individuals for the crossover operator [4].

1.5 Basic Statistics

In this section we refer to some basic statistics that we use in order to explain experimental and survey results. Supposing that we want to measure the height of a basketball team, we measure each player separately and the average height is equal to 1.95 meters. However, this number only does not explain the height distribution of the basketball team. The missing information in this experiment is called Confidence Interval. Assuming that our experimental results follow a normal distribution, confidence interval measures the degree of certainty in a sampling method [8].

$$CI = \overline{x} \pm z \cdot \frac{s}{\sqrt{n}} \tag{1.1}$$

There are two common values referring to this measure: 95% and 99%. Confidence intervals provide an upper and a lower bound from the mean value, in which assuming data follow a normal distribution, it is highly probable a result to be between this value range. Using our previous example, let's say that for 95% confidence intervals the upper bound is 2.05 meters and the lower bound is 1.85 meters. If we take 100 random samples from the above population then the mean value for approximately 95 of them should be between 1.85 and 2.05 meters. Confidence interval formula is shown in Equation 1.1 where $\underline{x}$ is the mean value of a sample, z is the confidence parameter (0.95 for 95%), s is standard deviation of the same sample and n is equal to sample size. Standard deviation measures the dispersion of a sample relative to its mean. The further the points from the mean the larger is standard deviation value [8].

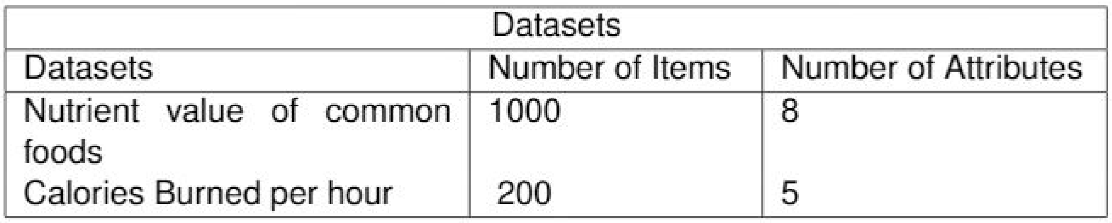
2. RELATED WORK

In this chapter, we present related work regarding bundles and genetic algorithms on recommendation systems.

2.1 Evolutionary approach for ’healthy bundle’ wellbeing recommendations

Hugo Alcaraz-Herrera and Iván Palomares [3] introduced a recommender system that using genetic algorithms promotes a healthy lifestyle by suggesting meals and physical activities to each user based on their preferences. They created a meal-exercise bundle which evolves through generations in order to suit each user’s preferences. Unlike other recommender systems domains such as movie, hotel or sightseeing recommendation where items are static entities in this approach bundles are highly configurable items.

Table 1: Data sets, number of items and number of attributes in [3] approach.



Datasets		
Datasets	Number of Items	Number of Attributes
Nutrient value of common foods	1000	8
Calories Burned per hour	200	5

So, for this approach they use one food information dataset [10] and one physical activity information dataset [9]. Food items have the 8 following attributes: Food id, Name, Type, serving size, Calories, Protein, Carbohydrate, Sugar, Fat with all nutritional values measured in grams. Physical activity items have: Exercise id, Name, Type, Intensity, Burned calories. A meal plus an exercise item is called a “bundle”. Each bundle is basically an individual and also a possible solution in the problem domain. Individuals are randomly initialized in order to broadly cover the search space. They also define a fitness function ff in order to evaluate each bundle accuracy through the evolutionary process. So, for each user $u_i \epsilon\ U$ with a goal G_i and preferences ψ_i , they set a fitness function ff_i in which $R = \rho_{1,}\rho_2, \dots \rho_N$ are a set of possible restrictions.

$$FF_i = FF(G_i) = \phi(\Psi_{u_i}; \Gamma_i(\rho_{i,1}), \Gamma_i(\rho_{i,2}), ..., \Gamma_i(\rho_{i,M})) \qquad (2.1)$$

with $G_i = \rho_{i,1}, \rho_{i,2}, \dots, \rho_{i,M}$" $\Gamma_i(\rho_{i,j})$ is an aptitude function describing the degree to which restriction $\rho_{i,j}$ is satisfied by the individual, Ψ_{u_i} is a restriction that measures how much u_i preferences are met, and Ψ_i is a combination function, e.g. an averaging operator" [3]. For example, if a user's u_1 goal G_1 is diabetes control, their model suggests moderate exercising, reduced sugar consumption and limited number of calories per meal. So user’s fitness function is defined as:

$$\phi(\Psi_{u_1}; \Gamma_1(\rho_{sugar}), \Gamma_1(\rho_{ex-mod}), \Gamma_1(\rho_{cal-lim}))$$

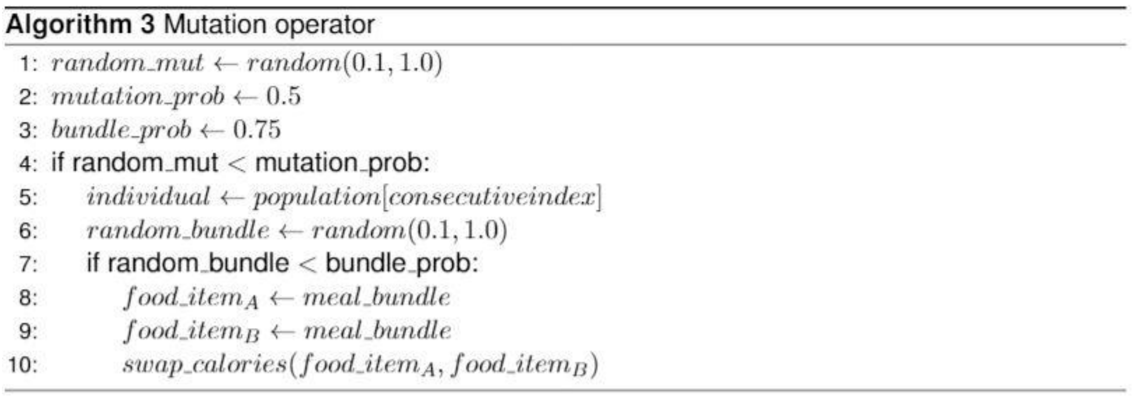
In order to modify the individuals' properties, they use two genetic operators: crossover and mutation. As stated above in crossover operation parts of two individuals (meals) are combined in order to create two ones in order to expand the search space. Each new meal inherits the exercising suggestion of its ancestor meal and as a result both exercises will move to the next generation. The individuals that are needed in this operation are randomly selected in Algorithm 1. Mutation on the other hand needs only one individual in order to happen and all individuals will proceed to this operation. For both operations there is a probability to not happen if the random number is not smaller from crossover and mutation probability respectively as seen in Algorithm 1 and Algorithm 3.

Algorithm 1 Crossover (population, weights, sum_weights)

```
crossover_probability = 0.9
$random\_cross \leftarrow random(0.1, 1.0)$
if random_cross < crossover_probability:
    $individual_A \leftarrow popuation[random\_index_1]$
    $individual_B \leftarrow popuation[random\_index_2]$
    $create(individual_A, individual_B)$
    $create(individual_B, individual_A)$
else
    add $individual_A$ to new population
    add $individual_B$ to new population
```

Algorithm 2 Function create($individual_A, individual_B$)

```
$new\_individual_A$
$bundle\_probability = 0.75$
$random\_bundle_A \leftarrow random(0.1, 1.0)$
if $random\_bundle_A < bundle\_probability$:
    $meal_A \leftarrow bundle_A$
    $meal_B \leftarrow bundle_B$
    $new\_meal \leftarrow build\_meal(meal_A, meal_B)$
    $activity_A \leftarrow bundle_A$
    $new\_bundle \leftarrow Bundle(new\_meal, activity_A)$
    add new bundle to new $individual_A$
else
    add $individual_A$ bundle to new individual
```



Algorithm 3 Mutation operator

```
$random\_mut \leftarrow random(0.1, 1.0)$
$mutation\_prob \leftarrow 0.5$
$bundle\_prob \leftarrow 0.75$
if random_mut < mutation_prob:
    $individual \leftarrow population[consecutiveindex]$
    $random\_bundle \leftarrow random(0.1, 1.0)$
    if random_bundle < bundle_prob:
        $food\_item_A \leftarrow meal\_bundle$
        $food\_item_B \leftarrow meal\_bundle$
        $swap\_calories(food\_item_A, food\_item_B)$
```

They also conducted two studies that showed two different results. At first, they measured average fitness function and likelihood of optimality for 5 users in order to find out their program accuracy on different users' goals. Likelihood of optimality basically measures the number of iterations in which the fitness value of the best bundle was better than the *0.95 * average* fitness of the best bundles from all iterations. From these 5 users they concluded that their program responds well to the users that wanted either to lose, maintain or gain weight or even control their diabetes. However, it needs improvement when it comes to users that want to build muscle as it probably lacks more expert knowledge on the nutrition and exercise requirements.

For their online evaluation they provided to 54 volunteers 4 bundles and asked them to pick their two favorites. They also asked them 3 other questions: to explain the reason for this choice, to rate from 0 to 10 the 4 options that were provided and to also rate the two options that each user selected. Almost 90% of the volunteers selected at least one of the two generated recommendations and 25% of them chose both of their recommendations. The average rating of the 4 bundles was 7.28 and of the two chosen bundles was 8.22. Lastly 66.7% of users stated that they chose these bundles because they liked both meals and exercises of those bundles.

3. OUR IMPLEMENTATION

3.1 Bundles

In general, Genetic Algorithms works through a population of individuals that can be evolved. In our case these individuals are called bundles [3]. Each bundle contains one meal object and one exercise object per day. For example, if a user asks for a 7-day diet program then 7 meals and 7 exercises will be the outcome of the recommendation algorithm. Each meal contains 8 food items in which two of them belong to the main foods category, two to side dishes, 3 to snack category and one from breakfast category. The meal formation is one breakfast dish and one small snack dish for breakfast, one small snack for mid-morning break and one for afternoon. In addition, one main meal dish and one side dish for lunch and the same for dinner. It is a well-structured meal plan that provides enough food items in order to not get hungry throughout the day and also get the necessary amount of nutrients and calories. Genetic algorithms tend to change or/and mutate these items in order to find a bundle that fits best to the target user, while keeping the above meal formation in order to always suggest a balanced meal. Bundles also contain one exercise object that contains one exercise per day. After users have selected the number of days for their diet plan, we then randomly create 10,000 bundles, in order to have a large population to work with. Both genetic operators will focus on the bundle's meal and not on the bundle's exercises due to their high importance and large diversity.

3.2 User Information

3.2.1 General Questions

Our program’s goal is to provide a healthy but also an enjoyable diet plan to each user. In order for this program to work it needs some information to be provided by users. At first it asks users for some general information like their gender, age, height and weight. It needs this information in order to find their basal metabolic rate (bmr)[12]. To calculate bmr we use the revised Harris-Benedict equation (Equation 3.1) in which weight is measured in kg, height in cm and age in years. Basal metabolic rate shows the amount of calories needed to keep the user's body functioning at rest. Apart from the needed nutrients, the algorithm has to take their preferences into account. So we ask for their ratings on several food and exercise categories. They have to rate from 1 to 5, fourteen food and four exercise categories. In addition, it’s important to know the user's goal and their favourable exercise intensity. When someone asks for a diet plan we think that they probably want to lose weight. However this is not always the case, as there are several people that want to maintain or even gain weight. As a result, as a target goal they choose between three possible answers: lose weight, maintain weight and gain weight. Despite their target goal it is always important, apart from a balanced meal, to exercise frequently in order to maintain fitness and control their weight better. That’s why exercise intensity is an important factor as different users have different physical fitness and endurance. So we let them choose one out of 3 different exercise intensities: Low, Moderate and High. Later we will see how this choice will affect the final number of calories needed in a day in order to reach their target goal.

$$\begin{aligned} Men : BMR &= 88.362 + (13.397 \cdot weight) + (4.799 \cdot height) - (5.677 \cdot age) \\ Women : BMR &= 447.593 + (9.247 \cdot weight) + (3.098 \cdot height) - (4.330 \cdot age) \end{aligned} \tag{3.1}$$

3.2.2 Calculations

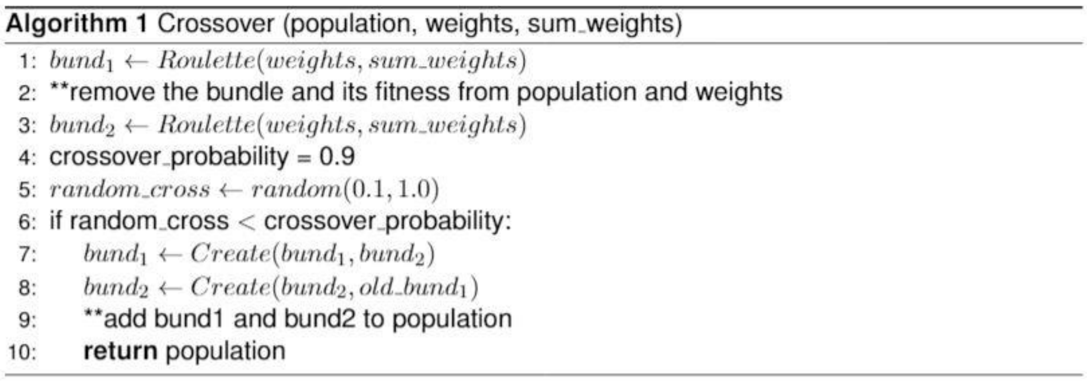
After finding bmr we multiply this result, depending on the user's exercise intensity, by 1.375 for low, 1.55 for moderate and 1.72 for high intensity. The outcome of this multiplication shows the number of calories needed in a day for the user to maintain their current body weight. If they want to lose weight, we subtract 500 calories and if they want to gain weight we add 500 calories to the above result [12]. Subtracting or adding 500 calories will help users to lose or gain around 1 kg per 10 days. Of course, there is a minimum daily calorie threshold that we can't go below in order to keep all users healthy. This threshold is equal to 1900 calories for male users and 1300 calories for female users per day. After finding the user's daily calorie target, we split these calories between three macronutrients: carbohydrate, fat and protein. In order to have a balanced meal they have to obtain certain grams from all of these 3 ingredients. There are two diets that, depending on if they are athletes or not, define the correct amount of each nutrient [6]. The first diet is called 40-30-30 in which 40% of calories come from carbohydrate, 30% from fat and 30% from protein. This diet is not for people with kidney problems or for people that participate in sports that need endurance. The second diet is provided by USDA Dietary Guidelines where 51% of the calories comes from carbohydrates, 33% from fat and 18% from protein. This diet is especially useful for athletes, so for people that are engaged with endurance exercises like walking, running etc.

3.2.3 Restrictions

Before moving on to more technical stuff like the algorithms and the performance metrics that we used we will discuss some restrictions users could set to our model. At first, we remove food or exercise categories that users have provided a rating equal to 1. We want to be sure that we will not suggest those foods that they do not want to taste. Furthermore, users have the option to choose one restriction out of the following: Lactose intolerant, Vegetarian and having Diabetes. For Diabetes and Lactose intolerant options we added a penalty in our fitness function and we will discuss it in 4.2. For vegetarians we remove items that contain meat from the search space. Removing only the food items that belong to the meat category is not enough as there are still several food items that contain meat in snacks, fast foods, prepared meals and other categories. For the second procedure we will take advantage of the names of food items. They are very descriptive and provide information about the ingredients of every food item. So, we remove food items that contain one of the following words in their name: [meat, duck, pork, beef, sausage, lamb, chicken, turkey]. Is there a probability for our model to still suggest an item that contains meat? Yes, there is but now at least it is almost zero. Due to the fact that we remove so many food items from the main meal category we move Vegetables from side dish to main meal for vegetarian users. So there are several options for vegetarians also for lunch and dinner

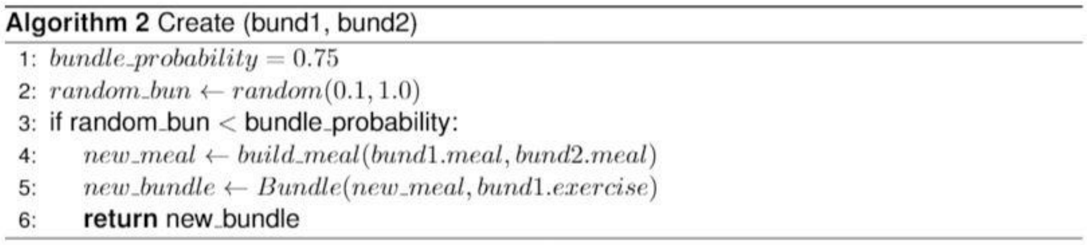
3.2.4 Algorithms

As said above our implementation model is based on [3] where they use two famous genetic algorithms: crossover and mutation. Our implementation of these two genetic algorithms has some small differences from [3] that significantly change the way that these algorithms work. Algorithm 1 and Algorithm 4 explain briefly how these two functions work.



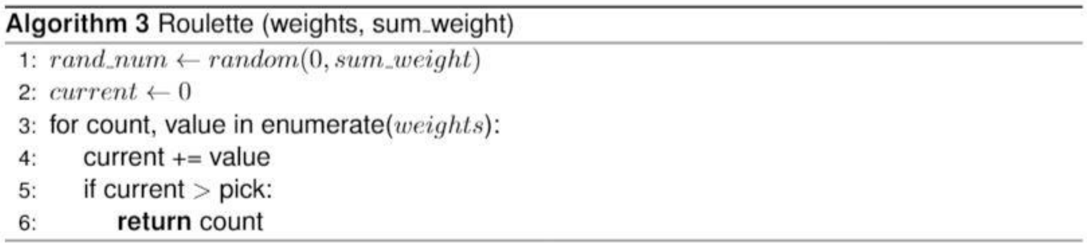
Algorithm 1 Crossover (population, weights, sum_weights)

```
$bund_1 \leftarrow Roulette(weights, sum\_weights)$
**remove the bundle and its fitness from population and weights
$bund_2 \leftarrow Roulette(weights, sum\_weights)$
crossover_probability = 0.9
$random\_cross \leftarrow random(0.1, 1.0)$
if random_cross < crossover_probability:
    $bund_1 \leftarrow Create(bund_1, bund_2)$
    $bund_2 \leftarrow Create(bund_2, old\_bund_1)$
    **add bund1 and bund2 to population
    return population
```



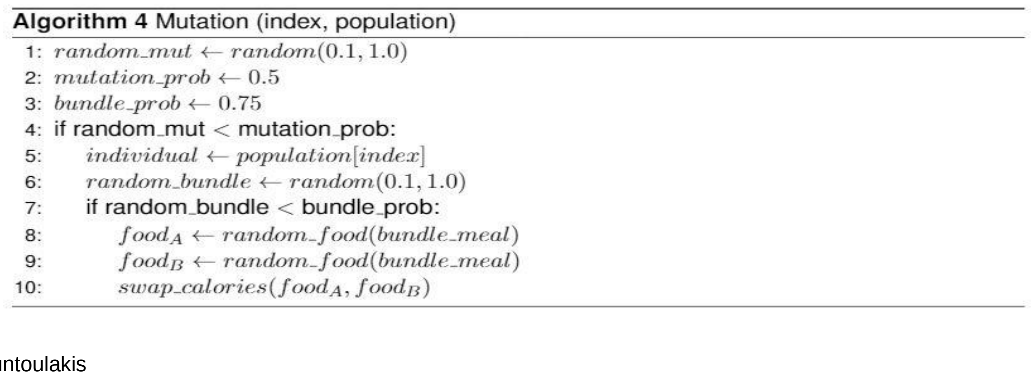
Algorithm 2 Create (bund1, bund2)

```
$bundle\_probability = 0.75$
$random\_bun \leftarrow random(0.1, 1.0)$
if random_bun < bundle_probability:
    $new\_meal \leftarrow build\_meal(bund1.meal, bund2.meal)$
    $new\_bundle \leftarrow Bundle(new\_meal, bund1.exercise)$
    return new_bundle
```



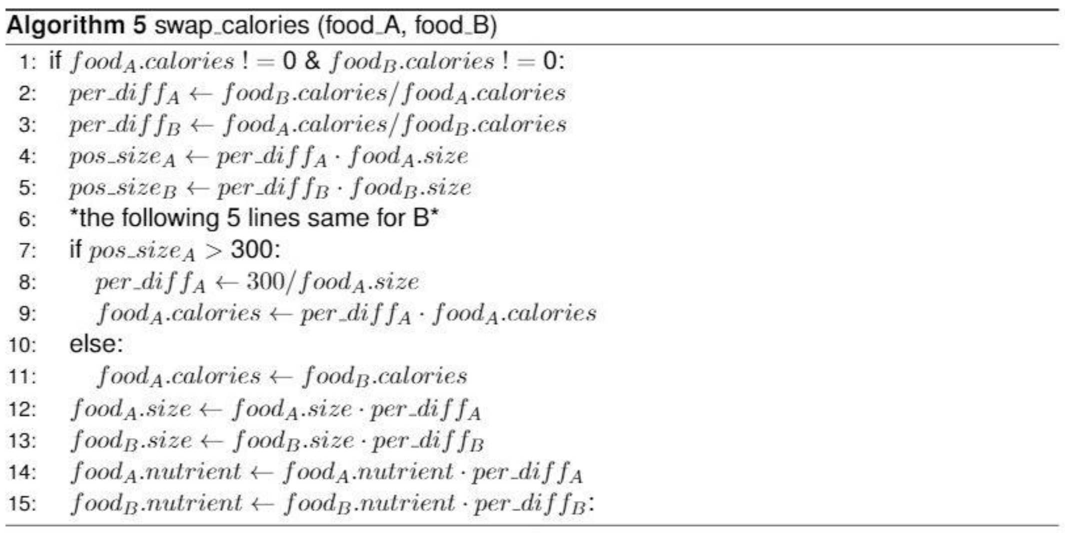
Algorithm 3 Roulette (weights, sum_weight)

```
$rand\_num \leftarrow random(0, sum\_weight)$
$current \leftarrow 0$
for count, value in enumerate($weights$):
    current += value
    if current > pick:
        return count
```



Algorithm 4 Mutation (index, population)

```
$random\_mut \leftarrow random(0.1, 1.0)$
$mutation\_prob \leftarrow 0.5$
$bundle\_prob \leftarrow 0.75$
if random_mut < mutation_prob:
    $individual \leftarrow population[index]$
    $random\_bundle \leftarrow random(0.1, 1.0)$
    if random_bundle < bundle_prob:
        $food_A \leftarrow random\_food(bundle\_meal)$
        $food_B \leftarrow random\_food(bundle\_meal)$
        $swap\_calories(food_A, food_B)$
```



Algorithm 5 swap_calories (food_A, food_B)

```
if $food_A.calories \mathrel{!}= 0$ & $food_B.calories \mathrel{!}= 0$:
    $per\_diff_A \leftarrow food_B.calories / food_A.calories$
    $per\_diff_B \leftarrow food_A.calories / food_B.calories$
    $pos\_size_A \leftarrow per\_diff_A \cdot food_A.size$
    $pos\_size_B \leftarrow per\_diff_B \cdot food_B.size$
    *the following 5 lines same for B*
    if $pos\_size_A > 300$:
        $per\_diff_A \leftarrow 300 / food_A.size$
        $food_A.calories \leftarrow per\_diff_A \cdot food_A.calories$
    else:
        $food_A.calories \leftarrow food_B.calories$
    $food_A.size \leftarrow food_A.size \cdot per\_diff_A$
    $food_B.size \leftarrow food_B.size \cdot per\_diff_B$
    $food_A.nutrient \leftarrow food_A.nutrient \cdot per\_diff_A$
    $food_B.nutrient \leftarrow food_B.nutrient \cdot per\_diff_B$:
```

The combination of Algorithm 1, Algorithm 2 and Algorithm 3 is the well-known genetic operator called crossover. For this operation to happen we need two bundles. In order to find these two bundles, we use the roulette wheel selection algorithm. Having already found the fitness values of each bundle, we subtract their fitness from the one with the maximum fitness value. In this way we make this a maximization problem. So the bundle that has the largest fitness is also the one with the highest probability to be selected by the roulette wheel selection algorithm. After we have found the two needed bundles we ask for a random number, and if the random number is smaller than the crossover probability we proceed to the creation of a new bundle. In order to create the meal section of this new bundle we choose food items from both bundles that were selected above. However, it's important to maintain the same meal formation with 2 main dishes, 2 side dishes, 2 foods for breakfast and 1 for mid-morning and afternoon break, in order to have a balanced meal. Then we assign to the new bundle the exercise activity of the first bundle. Last but not least we add this new bundle to a new population. We repeat this method twice in order to preserve both activities in the original population and the same number of bundles by creating a second new bundle following the same procedure. After repeating this algorithm for CrosNum times, where CrosNum is the number of iterations for the crossover algorithm, we have $2 \cdot CrosNum$ new bundles. In order to keep the same number of individuals we add to the new population the best individuals from the previous generation. Then using the new population, we proceed to the mutation algorithm.

Algorithm 4 is basically a mutation algorithm where instead of interchanging numbers between two indexes we interchange features between two meals. So, this method needs only one bundle and we execute it on every generation for all bundles. At first, we ask for a random number. If this number is below our first threshold which is equal to 0.5, we ask for a second random number. If the second number is below 0.75, a second threshold, two random food items of this bundle will swap calories. If any of these two random numbers is not small enough to pass any of the above two thresholds, mutation will not take place for this bundle. The Swap_calories method does not only swap their calories but also adjusts food's size, sugar, protein and carbon in order to match their new calories. This way we have a different meal that while it provides the same sum of calories, it may provide less sugar or less fat that are not beneficial for our organization. In order Algorithm 5 to take place, calories of both food items can not be equal to 0.

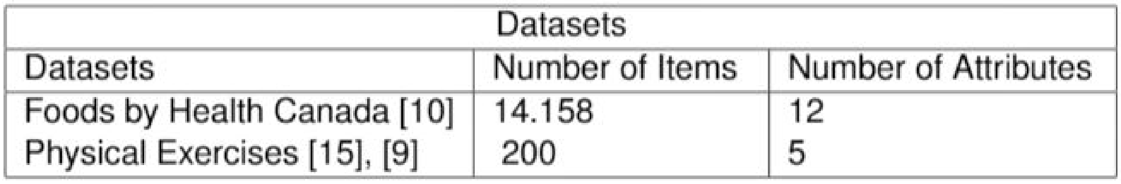
Firstly, in Algorithm 5, the percentage of calorie increase or decrease for each food item is calculated. Then we multiply this percentage with each food size in order to find the new size of each item. We set a size threshold at 300 grams, meaning that no food item quantity can be larger than 300 grams. We want to avoid suggestions like 1-kilogram apples, 1 kilogram orange juice etc. An average meal weighs 600 kilograms so 300 grams per food item is considered a logical threshold. In addition, no food item quantity can be smaller than 40 grams in order to avoid suggestions like 15 grams of chicken etc. As said above some meals like breakfast, lunch and dinner contain 2 food items and we expect the main dish to weigh 200 grams and side dishes close to 300 grams. So if by swapping calories food item size becomes larger than the above threshold we set its calories to a value that would make its food size equal to 300 grams. So then after we have found the percentage difference between final and initial calories of this food item we multiply protein, fat and sugar values by this percentage in order to find the correct food nutrients. For example, if one apple provides 95 calories, 1 gram of protein, 25 grams carbohydrate etc then when we mutate this food item to 190 calories we have to also change protein to 2 grams, carbohydrate to 50 grams etc. By changing the size of these food items, we create new bundles and expand our search space and by expanding our search space we have a higher chance to find the optimal suggestion.

4. EXPERIMENTS

4.1 Data

The two data sets used for these experiments are presented in Table 2. As seen, the food item dataset was provided by Health Canada while the physical exercises dataset was built by using information from both [15], [9] sources. In the two following sections we conduct a small analysis on the food and exercise items used in our experiments.

Table 2: Data sets, number of items and number of attributes in our approach.



Datasets		
Datasets	Number of Items	Number of Attributes
Foods by Health Canada [10]	14.158	12
Physical Exercises [15], [9]	200	5

4.1.1 Food items

Food items are basically parts of a meal in a day and the dataset that we use [10] was provided by Health Canada. So it contains 14.158 food items with lots of nutrient information but we use only the following attributes: Food id, Name, Description, Category (Figure 4.1), size, calories, protein, sugar, fat and carbon, lactose. Food id feature contains unique integer identifications, name is the name of each food, category is the food category where it belongs, and description is a short description. All except food nutrients and calories are of type strings. Calories are basically energy measured in kcal (kilocalories) and all nutrients are measured in grams. In some nutrients instead of a decimal number there is the word ”tr” which means traces of that ingredient. We replace tr with 0.05 which is half of the smallest number in each column. Moreover, we replace nan values in description with the word Unknown and nan in the other columns with zero (0).

As Figure 4.1 shows meat has more than 3000 food items making it the category with the most food items. Some other big food groups are vegetables and baked foods with almost 2300 and 940 food items respectively. As seen, all categories have a different amount of items which creates a different probability for each food item from each food group to be added to a diet program. Food items from meat groups have the highest probability to be found in a random made meal and food items from Spices and Herbs the least. However, besides these 3 large categories and some small ones, most categories contain between 300 and 1000 food items which explains the high diversity that this dataset presents. Despite the high diversity we have to create a meal form in order to present a logical meal structure to users. For instance

23

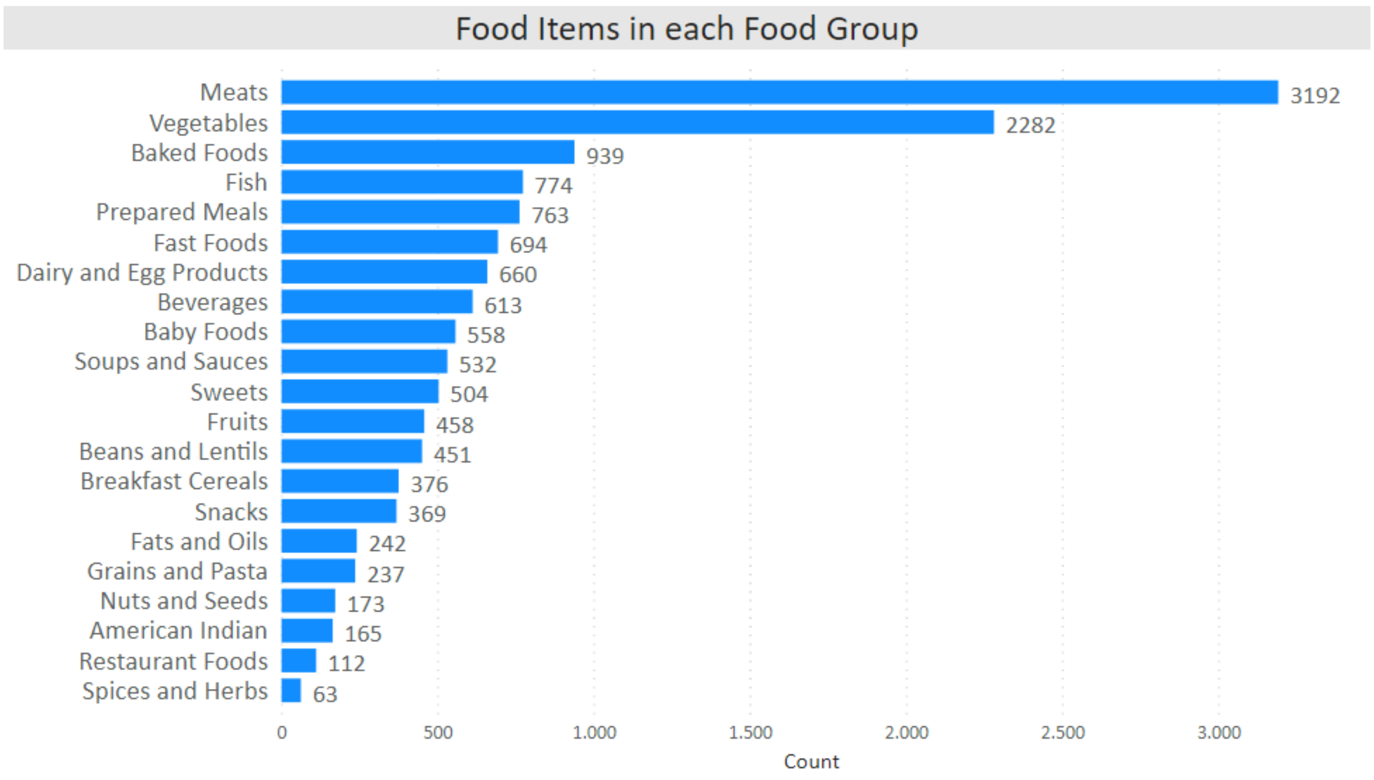


Figure 4: Number of food items from each food category.

it will not be ideal to suggest to users to eat fish or fast food for breakfast as it will not be appreciated by most of them.

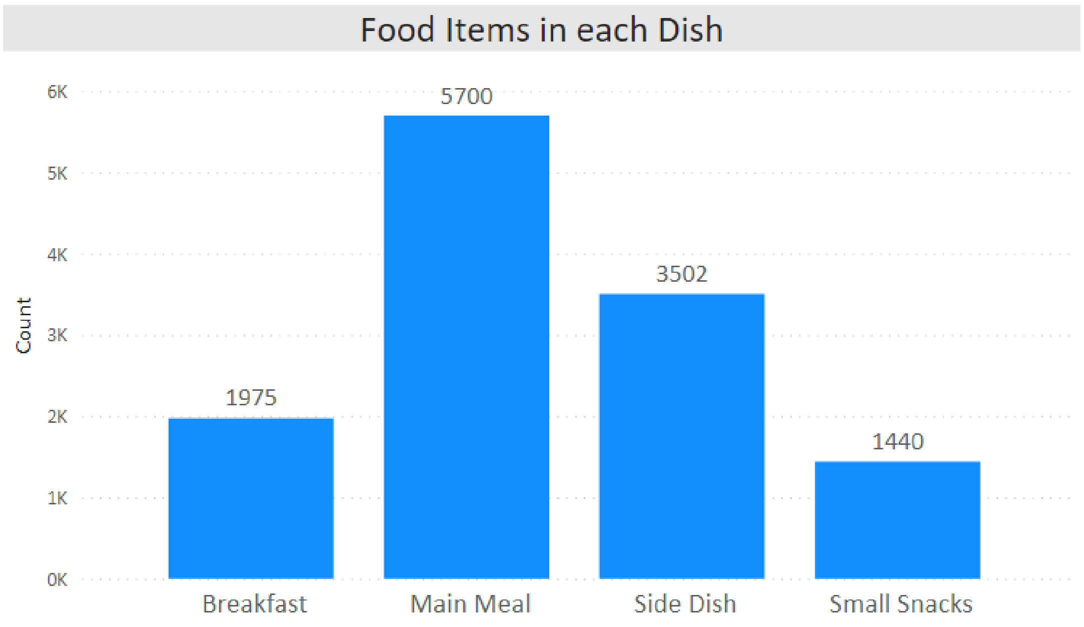


Figure 5: Number of food items from each Dish

We create an extra category, called Dish, that will help us build a specific meal forma- tion. This attribute can take up to 4 different values (Figure 4.2). In snack-breakfast dishes we assign foods from the categories below: Baked Foods, Breakfast Cereal, Dairy and Egg Products. In the main

dish there are food categories that are considered main meals like: American Indian, Restaurant Foods, Meats, Fish and Prepared Meals, Fast Foods. Supple- mentary dishes contain food items that are considered side dishes and belong to the following categories: Vegetables, Soups and Sauces, Beans and Lentils and Grains and Pasta. Lastly small snacks contain snacks that users can eat anytime of the day like Beverages, Fruits and Snacks. There are also some food categories that have not been assigned to none of the above Dish categories. We will not use these food items because either they are too specific (Baby Foods etc.) or they do not add a remarkable amount of nutrients and calories to a meal (Spices and Herbs etc.).
Figure 4.2 shows that more than 35% of food items belong to the main dish. There is then a big diversity of foods for lunch and dinner and can assure that they have something different to eat for these meals each day. This is important so they can have a balanced diet and avoid getting fed up with similar flavours every day. We also have close to 3500 food items that belong to the side dish category and 1440 snacks that will provide them energy through the day. Lastly there are also 1975 food items that belong to the Breakfast category and while most of them are dairy and egg products users still have lots of choices that could satisfy their needs and preferences.

4.1.2 Exercise

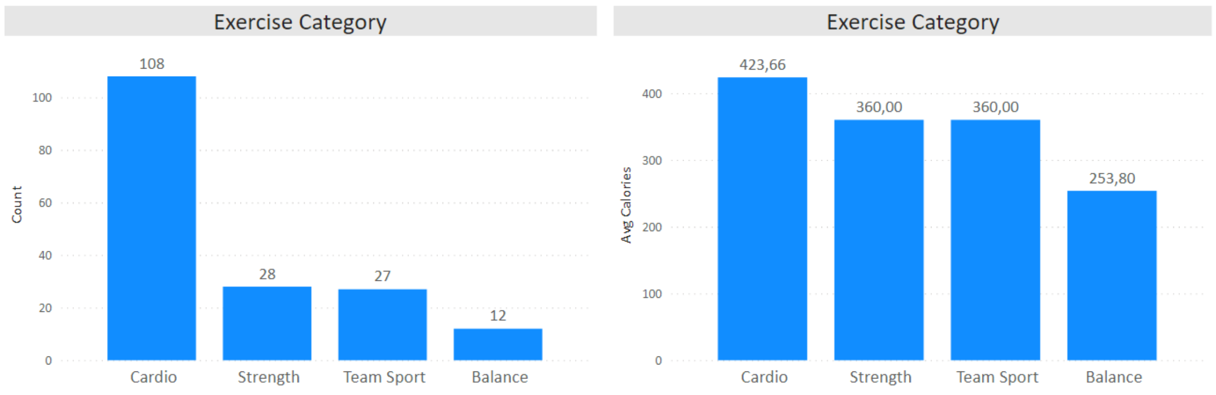


Figure 6: Left Figure shows the number of exercise items per exercise category. Right Figure shows the average calories burned by one exercise in each category for moderate effort

Apart from eating healthy it is also important for users to exercise frequently in order to boost their energy and better control their weight. We obtain the exercise dataset from [15, 9] and create one that meets our requirements. So, there are 175 exercises belonging to 4 different exercise types: Cardio, Balance, Strength and Team Sports. As seen in Figure 6 almost 60% of the exercises belong to the Cardio category. Strength and Team sport have a similar number of exercises and lastly balance has only 12 exercise items. Cardio is one of the most important exercise categories, as almost all exercise programs recommend doing at least one cardio exercise daily. Moreover, they burn, in general, a larger number of calories than the other categories for moderate effort as seen in Figure 6. Strength and Team sports burn almost the same calories while balance exercises burn the least. Each exercise category benefits us differently however we will not go that deep and we will consider only the number of calories each exercise burns. As a result, we understand that the author of this dataset focused on providing exercises that burn a significant amount of calories and not a balanced dataset in terms of exercise category. These items have the following features: Exercise id, Name, Type, Intensity and Calories burned. Calories burned category is referenced for a person weighing 59 kilograms and for 1 hour activity. So, it is just an estimation but it still provides us valuable information. As we will see later, we let users choose the exercise intensity and type based on their preferences. While the final outcome may not contain their most favorable exercise type, as we will discuss below, their preferences matter in a significant way.

4.2 Performance Metrics

In this section we are going to introduce two performance metrics similar to [3] which are typical in evaluation of genetic algorithms. We introduce a fitness function that counts the performance of each bundle and the average performance of all bundles for each execution. This function is a dissimilarity metric and it shows how different is each bundle versus one that perfectly satisfies the target user's needs and preferences. Genetic algorithms' goal is to make each bundle's fitness score as small as possible. This function adapts to the user's restriction while also keeping the basic formation presented below.

$$FF = \alpha_1 \cdot |target - real| + \alpha_2 \cdot preferences + \alpha_3 \cdot same\_main\_food + \alpha_4 \cdot extra\_restr$$

$$FF = \frac{FF}{days} \tag{4.1}$$

$$\overline{FF} = \frac{\sum_{i=1}^{n} FF_i(k)}{n} \tag{4.2}$$

$$\tag{4.3}$$

where $\alpha_1 = 4, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 50$. Let's explain separately each part of equation 4.3. |*target-real*| is the absolute difference between a target value and a meal value. BMR and each user's diet plan provide us with a calorie and a nutrient target for the user per day. As said above we use three food nutrients: carbon, fat and protein. The smaller the difference between target and bundle's value for each nutrient (and calories sum) the better this meal fits the target user. Preferences contain user's preferences for each food and exercise category. Each bundle contains one exercise and 8 foods per day. Users' rating for each category is from 1 to 5 and we subtract this number from 6 in order to make it a minimization problem. The lower the preference value the more they like this food/exercise type. We multiply the result by α_2 in order to add some more weight to this category. It is important for the user to not eat the same meal every day. As a result, we add a punishment if two consecutive days contain the same main meal for lunch or dinner. This punishment is a 50% increase of the final fitness and is represented by the *same_main_food* parameter. Moreover, if they have rated 2 or more categories with a value larger than 2 then we add a punishment if the food item in the afternoon, in the breakfast and in the mid-morning meal belong to the same food category. This punishment is equal to a 10% increase in fitness. There could also be some extra restrictions depending on users' preferences. Lactose intolerant users should avoid lactose in foods so we multiply by 20 each gram of lactose in foods. In addition, people with diabetic should avoid sugar and fat so we multiply each gram of these two types by 5 as a punishment mechanism. For vegetarians we avoid recommending them food items that contain meat and also add Vegetable’s category in Main Meal Dish. *overline* is the average of the best bundle's fitness value found after k generations and average them by the number of executions. Lastly the values of parameters α have been set this way in order for needs and preferences to have equal effect on the final recommendation.

The second metric is called Likelihood of optimality. k is the number of generations in a genetic algorithm, n is the number of consecutive executions and *m* = *n* is the number of executions in which an optimal solution is found after k generations.

$$Lopt(k) = \frac{m}{n} \tag{4.4}$$

This function basically uses the fitness function stated above. To explain the likelihood of optimality better, imagine that we run these algorithms for 100 executions and 100 generations. Then we find the best bundle's fitness value for each generation and take the average fitness of the best bundles for each execution. So, we have 100 averages of the best bundle's value. Then we again take the average of these values just to have only one average fitness value for all executions. We set an optimality threshold 5% above this value and we call it Average Fitness Value (AVF). After that for each execution we check the average fitness value for all generations if it is above or below the AVF. Finally, we count the number of executions that are below AVF. So for example if 88 executions have average fitness value per execution below AVF then Lopt(100) = 0.88. This metric explains a bit better this algorithm performance because it does not depend on user preferences. For example, we will find smaller fitness values if the user likes most of the food categories or all exercise types. That does not mean that the algorithm works better than another user that likes only 3 or 4 types of foods.

$$\begin{aligned} Nutrient\_Accuracy &= \frac{1 - |target - value|}{target} \\ Accuracy &= \frac{Nutrient\_Accuracy}{4} \end{aligned} \tag{4.5}$$

Equation 4.5 is used in the last section of this thesis. There we analyze the results of a survey we conducted and we want to calculate the nutrient accuracy of each recommendation. So, we use the above equation that shows how close the bundle's nutrient values are to the target's nutrient values. When calculating the fitness of each bundle we use scaled nutrient values while in this equation we use real nutrient values. To make the long story short, fat scaled values range from 1 to 5 while real values could range from 0.01 to 300 grams. So, this metric is more accurate in terms of the meal's nutrient accuracy than the one we use in our fitness function.

4.3 User Feedback

As discussed earlier, lots of organizations ask for users feedback on their products or on their suggestions. It is important to know if users enjoy using the company's product so they can continue evolving it or if there are things that they don't like and need revising. We set a fitness function (equation 4.3) in which both preferences and users affect almost equally the final outcome. But is this what users want? There is only one way to find out. We have to present to users' different diet plans by adding different weight on each parameter of the fitness function. So, we change two α parameters in the above equation:

α_1 and α_2. Parameter α_1 defines the weight that food nutrients will have on the final suggestion. Parameter α_2 affects the importance of users' preferences on food and exercise categories. α_3 parameter can take only two values: 0 and 1. When α_3 is equal to 1 then when a certain bundle contains the same main food in two consecutive days then fitness increases by 50%. That way we avoid providing similar main foods for consecutive days that most probably will discourage users to stick to their diet plan. They provide us their feedback by rating from 1 to 5 where 1 means "Very Dissatisfied" and 5 "Very Satisfied" by the recommended diet plan.

4.4 Time, Fitness vs Number of Generations

For the first experiment we wanted to learn three important things about our recommendation system. The time needed for this algorithm to run for 50 iterations and different numbers of generations. The average fitness distribution of the best bundles through the evolution process results and also the fitness of the best bundles for each generation and 30 iterations. We execute the crossover operation for 20% of population at each generation, so at most 20% of the population will change. Meaning that 1000 crossovers will create at most 2000 new bundles. We say at most because there is a possibility for a crossover to not happen if the random number is larger than the defined threshold.

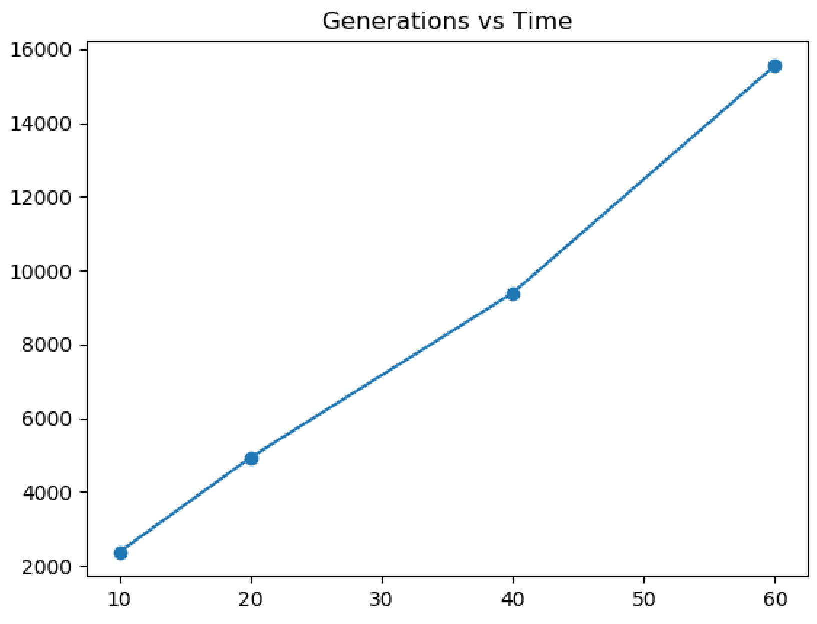


Figure 7: Average time needed for 50 iterations and 10, 20, 40 and 60 generations

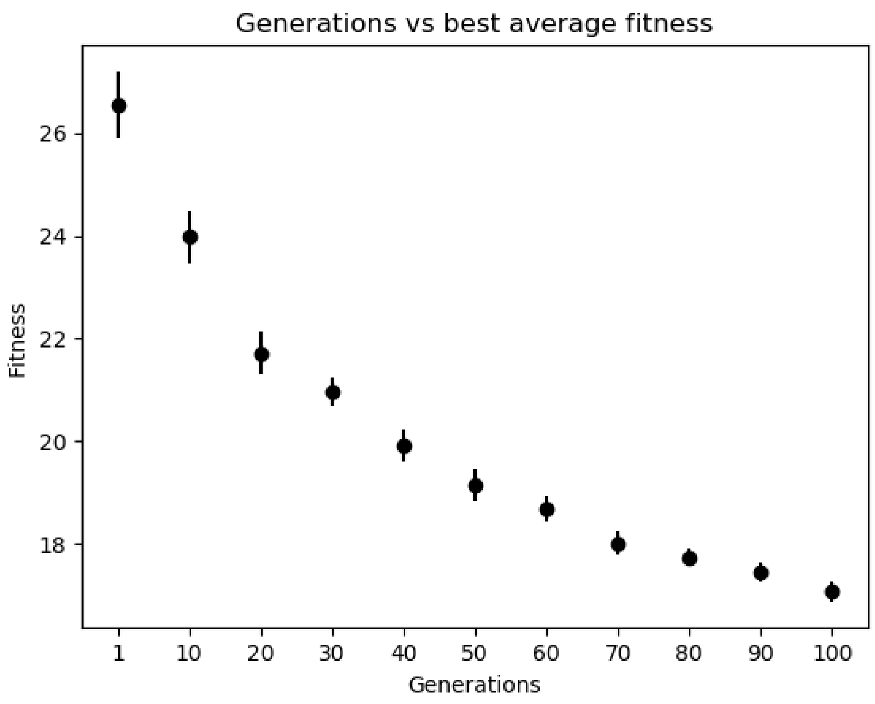


Figure 8: We collect the fitness function value for the best bundle for a different number of iterations. We repeat the experiment 30 times. Dots show the mean fitness value of the best bundles and error bars show the 95% confidence interval

At first, we create 10,000 random bundles that contain random foods from each dish and a random exercise for each day. Despite the bundle's randomness they follow the meal formation we stated above with a specific number of food items from each Dish category. We use the same random population for each iteration and generation number in order to make a fair comparison. We asked for a 1-day plan and identified the user as a male of 82 kg with random food and exercise preferences, a non-athlete that wants to lose weight. We provided random food and exercise preferences and the average rating for food categories is equal to 3.25 and for exercises 2.5 while having fives and ones in both preference types. For every generation 2000 random bundles will go through the crossover function and all bundles will go through the mutation function. For the crossover operation we use the roulette wheel function in order to find two bundles that will participate in this interaction. Let's briefly explain how the roulette wheel function works. Before even our algorithm begins, we find each bundle initial fitness. So, at the first iteration we use the initial fitness of each bundle in order to give them a probability to be selected in a crossover algorithm. So, for every iteration k we use the fitness that each bundle had at iteration k-1. After both crossover and mutation algorithms are completed, we again calculate each bundle fitness. Finally in terms of hardware we use multithreading with 3 threads which was the maximum number our laptop could handle. We conducted this experiment for 100 generations and 30 iterations. Every 10 generations we get the best bundle for each iteration.

As expected, (Figure 7) there is a linear relation between time and the number of generations. It needed 36 minutes to run the algorithm for 10 generations, close to 73 minutes for 20 generations and more than 4 hours for 60 generations. The line equation that fits well to the first 3 points is: $y = 256 \cdot x - 194$. For 60 generations it took a little more time than expected, maybe due to higher hardware temperature. However does a larger number of generations equal a better fitness function result? This should be the case but the randomness existing in both generation algorithms could always provide us a result that is an exception to the above rule.

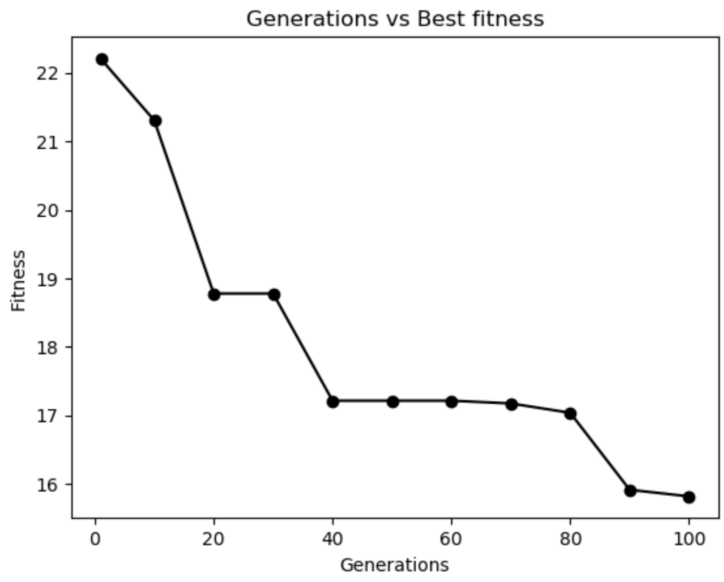


Figure 9: Average fitness function value of the best bundle of each generation. Then average it for the number of generations.

Figure 9 shows bundle fitness for different generations (1-100). As said above we run this experiment for 100 generations and we repeat it 30 times. Every 10 generations we collect the bundle with the lowest fitness value. So at the end we have 30 records for each generation number. Mean fitness values of the best bundles are presented as dots and error bars show the 95% confidence interval in Figure 9. So at first our bundle's average fitness value is equal to 26,6 and confidence interval is just above 0,6. For 10 generations the mean fitness value of best bundles is 24 and 95% confidence interval is 0,5 from mean value. We notice here that the fitness difference between these two generations is considered important because it is larger than the sum of their confidence intervals. This rule still applies until we reach 50 generations. From Figure 9 can be easily noticed that the fitness difference between 50 and 60 generations is smaller than the sum of their confidence intervals. Mean difference is equal to 0,4 and sum of their confidence intervals is equal to 0,5. So the fitness difference between these two generations is not statistically important. However, the fitness difference between 60 and 70 generations is statistically important so our algorithm is still providing us better solutions while the generation number increases but at a slower pace than before. For larger generations we probably need to check every 20 generations in order to find statistically important fitness differences. This could only mean that we are getting closer and closer to an optimal value or to a local minimum. In addition, as depicted while the generation number is getting larger, the confidence intervals are getting smaller. That means that more and more bundles are getting closer and closer to optimal bundles as they evolve. This could also mean that food diversity is also getting smaller and smaller due to the probabilistic nature of the crossover algorithm. However, are the bundles with the lowest fitness values for each generation close to these averaged values? And also having a smaller diversity for a larger generation could it make it possible to be stuck in a local minimum far away from the global minimum? We will try to answer both questions later in this thesis.

Figure 9 presents us with fitness values of the best bundles for the same eleven number of generations. These values are probably not inside the intervals presented in Figure 9. Most of the times the worst and best bundle’s fitness are not inside the confidence intervals range. However, if some bundles with high fitness value are far from the mean (outlier) then there is a possibility for the best value to be inside these confidence intervals. As seen in Figure 8 and Figure 9 there is a similar distribution between these points. For a smaller number of generations fitness values are decreasing fast and as the generation number is getting larger $\frac{dF}{dG}$ (F for fitness and G for generation) is getting smaller. For instance, the fitness difference between the best initial bundle and the best bundle after 20 generations is equal to 3,2. On the other hand, the fitness difference between 40 and 60 generations is almost equal to zero. It is also important to notice that the fitness difference between best bundles and average of 30 best bundles for each generation is pretty big. There is always more than 8% difference which is a pretty significant difference for all generations.

4.5 Crossover Number

In this section we will repeat the above experiment 3 times but each time for a different crossover number. We want to know if a larger crossover number would lead the algorithm to stick to a local minimum. Also, we will learn the effect that different crossover numbers have on the decay rate of the fitness distribution. Our population is made by10^4bundles and we ask for a 1-day diet program. User is again male, weighing 82 kg and has the same random food and exercise preferences for all iterations. We will again check its fitness values for up to 100 generations. We will also repeat this experiment 30 times in order to limit some algorithms' randomness. The first-time crossover will take place 1000 times per iteration, meaning that at most 20% of the population will alter in just one generation. The second time crossover will take place 2000 times and the last time 3000 times. The third time it is possible to change up to 60% of the population through crossover operation in only one generation.

However, there is a benefit-cost dilemma that we have to take into consideration. Altering such a high portion of the population in only one iteration could destroy its diversity. Diversity is very important in a population in order to avoid local minimums. Because with a low diversity we might be stuck with a bundle being in local minimum far away from the optimal solution (global minimum). On the other hand, smaller diversity means that it is easier and also quicker to decrease fitness (up to a certain point of course). As we know crossover operation will use the roulette wheel selection algorithm in order to choose its two bundles. This is a probabilistic algorithm meaning that bundles with lower fitness scores have higher probability to be chosen. So now it's easy to understand and explain our dilemma. Should we use a large crossover number in order to try to get as fast as possible to the bundle with the lowest fitness score, or a small crossover number in order to have larger genetic diversity and larger probability to get closer to global minimum?

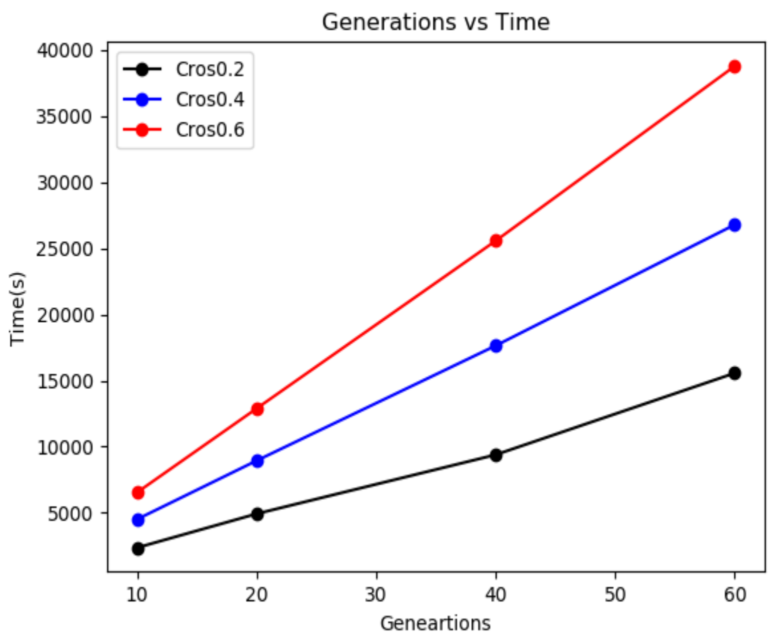


Figure 10: Time needed for different number of crossovers per generation. Cros0.2 is for at most 20% of the population meaning 10^3 crossovers. Similarly, Cros0.4 is $2 * 10^3$ crossovers and Cros0.6 is $3 * 10^3$ crossovers.

Figure 10 shows the time needed for the experiment to happen for each one of the 3 iterations. As also seen from Figure 7 while the generation number gets larger, each iteration needs more time to happen. Figure 10 also the same thing for different crossover numbers. It is also interesting to notice that we don't have 3 parallel lines. While the crossover number gets larger, the slope of these lines also gets larger. While crossover number increases time growth rate is also increasing making more time expensive to have a large number of crossovers and generations.

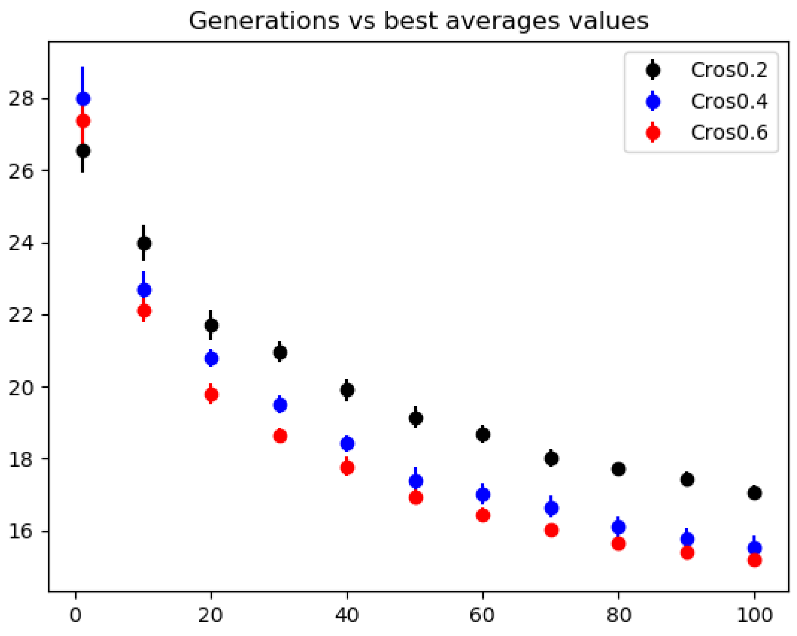


Figure 11: Dots are the mean value of the best bundles and error bars show 90% confidence intervals. For each generation number there are 50 samples meaning 50 iterations.

From Figure 11 we reach to several conclusions. At first for 10 and 20 generations we see that fitness values are smaller while the crossovers number per generation gets larger. While for 20 and 40 generations the difference between cross0.2 and cross0.4 is getting wider. Their confidence intervals also do not coincide which means that they have significant differences. For 60 generations this difference is almost the same. However the difference between cross0.4 and cross0.6 is almost zero. Could this collision mean that we are close to an optimal solution? Or did we get trapped to a local minimum due to the lower diversity of the population? And lastly having more crossovers per generation can this always lead to better solutions for the same generation number?

4.6 Like everything and nothing

In this experiment we test the effect that users' preferences have on fitness distribution. Do lower ratings lead to smaller decay rate or it just moves fitness distribution horizontally? In order to find out we create 2 different users with the same physical attributes. Both users have the same age, weight, height and diet goal. However, they have one main difference. The first user likes all foods and exercises and the other none. This means that the first user rated all food and exercise categories with a 4 or a 5 and the second user with 1 or 2. More specifically the first user rated with 4: Baked Foods, Snacks, Sweets, Restaurant Foods, Beverages, Fruits, Beans and Lentils and Prepared Meals and with 5: Vegetables, American Indian, Meats, Breakfast Cereals, Soups and Sauces, Fish, Nuts and Seeds, Grains and Pasta, and Fast Foods. He also rated all exercise categories with 5. The second user rated with 1: Vegetables, American Indian, Restaurant Foods, Breakfast Cereals and Nuts and Seeds and all the other foods categories were rated with 2. All exercises were rated with 1. From this experiment we will see how users' preferences affect our fitness function through the generations. In this experiment we did not use the rule that removes all the food categories that have been rated with 1. If we have done that we will have only a few items to work with.

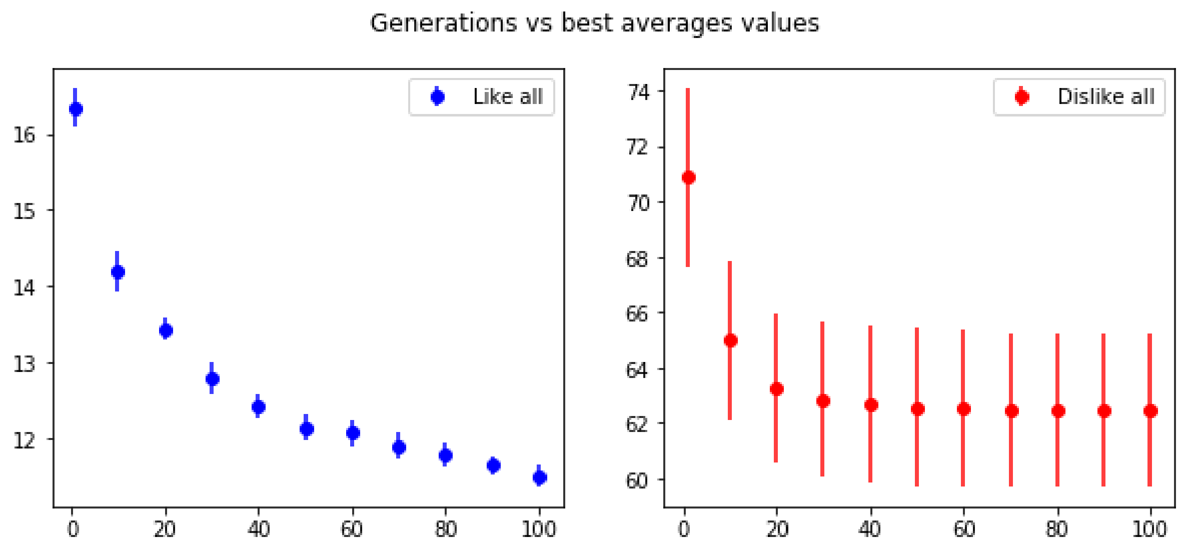


Figure 12: Average of best bundles per iteration for different generation number and two different

users. One that like all categories and one that likes none.

As expected, Figure 12 shows that for a user that likes all categories fitness values are lower than the one that dislikes all categories. This happens due to our fitness function. As said above we take into consideration user's food and exercise preferences. Not only users that like all foods get lower fitness values but the difference between these two users in any generation number is significant. We say that the difference is significant because 95% confidence intervals for both users do not overlap one another. It seems that fitness values from both users follow similar distribution as their average fitness values create two almost parallel lines. Does fitness of the best bundles present similar results?

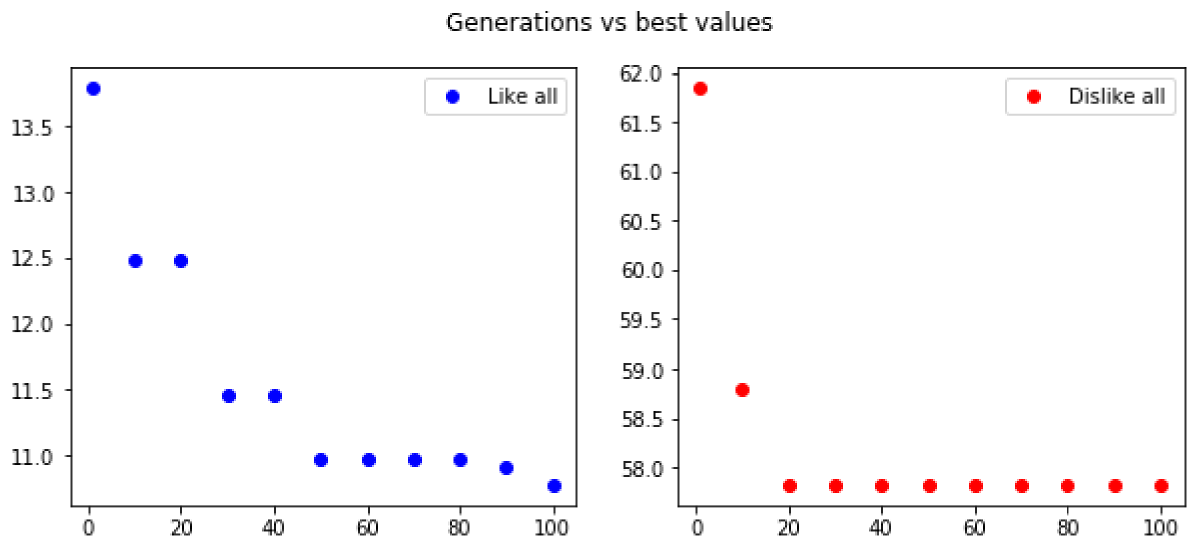


Figure 13: Fitness of the best bundles for two different users.

We see similar results in Figure 13. As seen, the best bundle of the like-all user is always smaller than the other bundle. They follow an almost identical distribution with the average fitness values presented above with similar fitness differences between these two users. The difference between the best bundle of each user in each generation is above 100 fitness. As seen from Figure 9 this kind of difference is considered significant. However, is this difference between these two users the same for all generations? Having the same initial population and physical attributes could it lead to a steady fitness difference between the users?

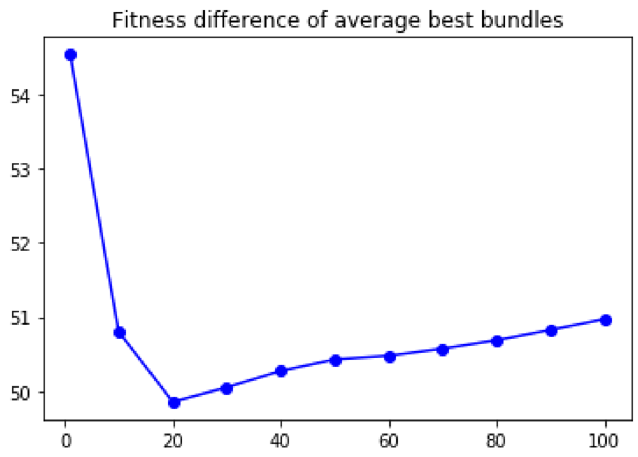


Figure 14: Difference in fitness of a user that likes all categories with a user that likes none

As seen from Figure 14 this difference is between 100 and 180 for these 4 generation numbers. For the first 3 generation numbers, as they get larger, the fitness difference is getting smaller. However, for 60 generations the difference of the best bundles between these two users is 180. We expect this difference to get smaller but not below a certain threshold. Assuming that we have the same identical bundle solution for both users, they have almost totally opposite preferences. So, we have to find the minimum and maximum threshold of one identical solution for both users. The maximum rating difference is 4 (5 for one user, 1 for the other) and the minimum is 2. So, *min_dif* = $10 \cdot 7 \cdot 2 = 140$ and *max_dif* = $10 \cdot 7 \cdot 4$ = *280*. Difference in 60 generations is equal to 180 which is inside these two thresholds. That means that it is possible to be very close to the optimal solution for both users. While there is not a specific number of generations that provide us the best possible solution, we will explore this quest for certain cases. To sum up users' preferences does not seem to affect the number of generations needed to reach an optimal solution and also even for users with lower ratings (not 1) on preferences we are still able to provide them a very good recommendation. But how close were we to an optimal solution? How many generations needed approximately in order to get close to a very good recommendation?

4.7 Near optimal solution

In this section we search for the generation number where fitness values stop decreasing and fitness difference between a number of generations becomes statistically insignificant. As seen in the previous experiment, food and exercise preferences do not affect the fitness value distribution through different generations. It just moves fitness distribution horizontally. So, for this experiment the target user is again male weighing 82 kilograms, having no food restrictions, wants to lose weight and asks for a 1-day diet program. He also likes all foods and exercise categories by rating them with 4 or 5. After initializing 10^4 random bundles we start the evolution process. To stop the evolution and conclude that we are close to an optimal recommendation, the fitness difference of the best bundles must be below 0.1 for 6 consecutive generations. For example, if best bundle fitness for tenth generation was equal to 24.2, for eleventh iteration was equal to 24.15 and for twelfth iteration 24.14 then we have 3 consecutive generations. If this phenomenon is repeated 3 more times the evolution stops and we present the lowest fitness bundle to users as their 1-day diet program. Each iteration could evolve for at most for 180 generations and we repeat this experiment for 30 iterations. In previous experiments we saw that fitness function was decreasing following an exponential decay as the generation number was getting larger. However, we expect this fitness value function to converge to a local or a global minimum and that fitness will stop decreasing in this evolution process. So, we collect the best bundle fitness every 10 generations or when the evolution process stops for each iteration.

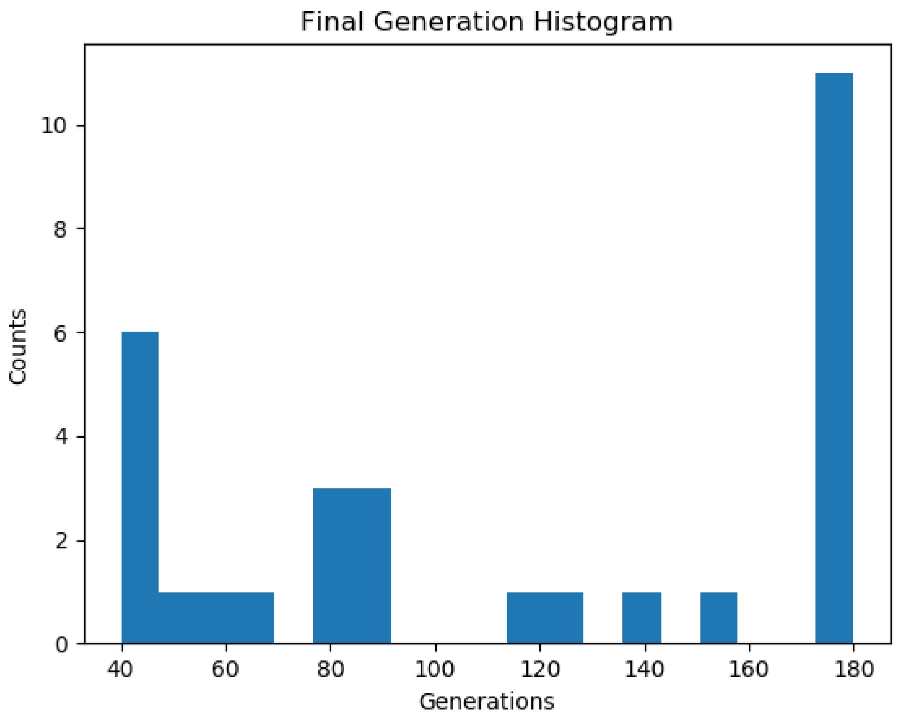


Figure 15: The last generation for each of the 30 iterations. The evolution process stops when it reach to 180 generations.

Figure 15 shows the last generation for each one of the 30 iterations. As seen 18 of them stop before our threshold which was at 180 generations which means that they did not find an optimal recommendation. In addition, 14 of them stopped at a generation number below 100 and 3 of them stopped after evolving for more than 100 generations and before reaching 180. We also notice that 6 iterations stop between 40 and 50 iterations. They evolved through a limited number of generations and they probably did not converge to a global minimum. These iterations probably converge to a local minimum and for 5 consecutive iterations they could not significantly decrease their fitness value. So even for such a small threshold, there are times in which the decreasing rate is getting smaller and it looks like it reached a solution. After this observation we may need to answer several questions. Did the algorithm find a bundle with a very low fitness score below 100 generations or was it just bad luck in our roulette algorithm? Meaning that there is a probability that the roulette algorithm chose bundles with high fitness scores despite being the least probable to be chosen. The only way to find that out is to look at the best bundles for each generation and the average fitness of the best bundles.

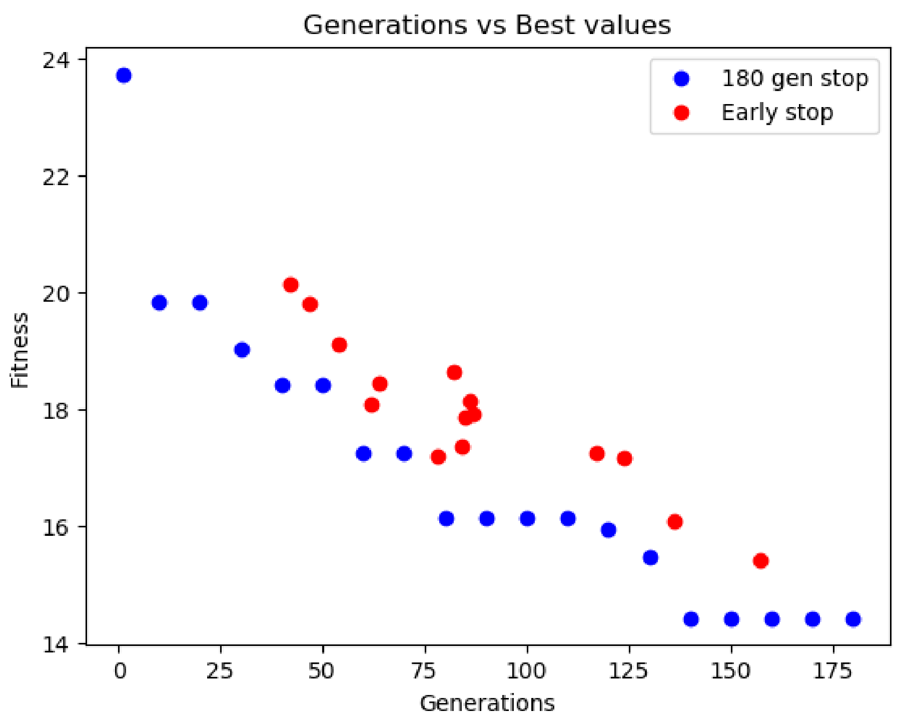


Figure 16: Blue dots depict the best fitness value for each generation and red dots show the fitness value and the generation number where the evolution process stopped before reaching 180 generations for each iteration.

Blue dots at Figure 16 show the best bundle (lowest fitness value) out of all iterations for each generation and each red dot shows the fitness value and the generation number on which 1 iteration stopped. This is the generation number after 6 consecutive generations where fitness value did not increase for at least 0.1 between two generations. As seen for similar generations, red dots are always higher horizontally than the blue dots. That means that these iterations converged to a local minimum and for 6 consecutive iterations they could not decrease their fitness score by more than 0.1 from the previous one. In addition, these two different distributions seem to be moving in parallel while they both lie in an exponential decay. So early stops will always lead to a local minimum and not to an optimal recommendation that can be found for the same number of generations. As the bundles keep evolving through the generations the fitness decrease rate is getting smaller. This could mean that we have found the optimal recommendation at 140 generations as the best fitness value for 40 generations is (almost) the same. However, in order to reach safe conclusions, we have to see the average fitness values of the best bundles for each iteration.

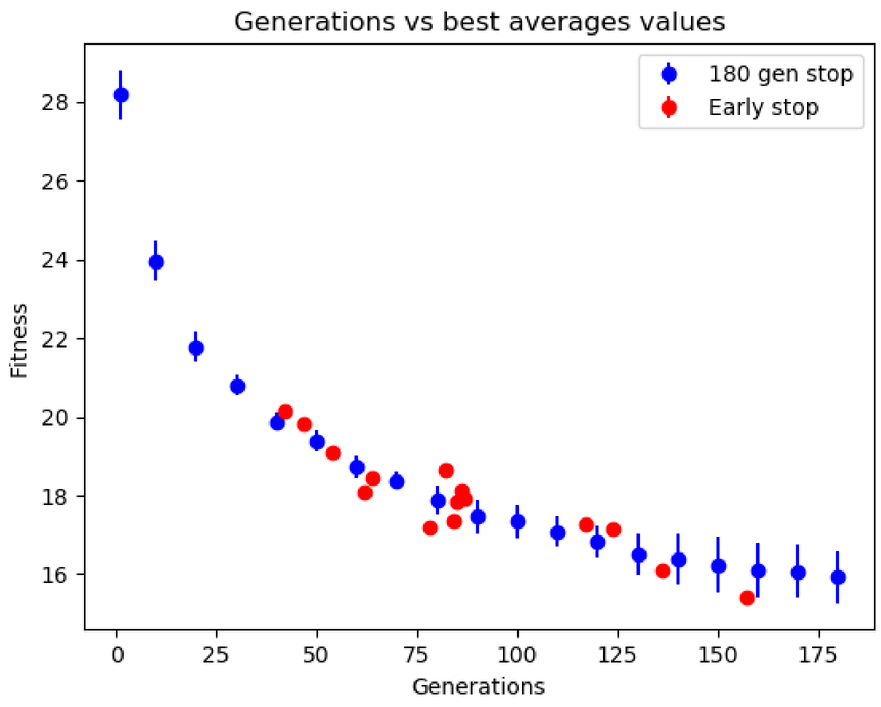


Figure 17: Average fitness of the best bundles for all iterations from each generation. Error bars represent 95% confidence intervals. The first generation has 30 values in its population and the last generation number (180) has 13.

Figure 17 shows the average fitness of the best bundles for 30 iterations and in this figure also the error bars represent the 95% confidence interval of those values. In this figure it is also clear that fitness distribution follows an exponential decay as seen from the blue dots. For the first 80 generations the error bars of those values do not coincide meaning that the fitness decrease is statistically important. However, after 80 generations confidence intervals are getting larger and larger meaning that the fitness difference between the best bundles is increasing. Another possible explanation could be that while the generation number is getting larger the iteration number is getting smaller as 18 iterations have stopped before reaching 100 generations. So, after 120 generations there could be iterations that have found the optimal recommendation and iterations that have probably stuck to a local minimum. So, in order to get the optimal recommendation and not a recommendation very close to it, we either have to be lucky or to repeat the experiment a number of times.

This Figure provides us also important information about the iterations that stopped before 180 generations. As seen, red dots depict the best values in Figure 17 and not the average of the best fitness bundles. This happens because there are not two iterations that stopped in the same generation. So, the best values at each one of these iterations are very close to the average best fitness values for non-stop iterations. So, in general while they have larger fitness than the corresponding fitness of the non-stop iterations, they have similar or even lower fitness value than the average fitness of the best bundles for all iterations. As a result, even when an early stop happens, we notice that this global minimum is not very different from the average best bundle. So, while it may not be optimal, we could run this algorithm and instead of providing a generation number we could provide a fitness difference between two generations in order to stop the evolution process.

5. USER STUDY

5.1 Questionnaire

There is only one way to find the correct weights for each parameter in the above fitness function, by asking the users. So, we conduct a study by asking 50 volunteers to rate 3 different bundles. Each bundle contains 1 exercise and a meal menu for 1 day based on the user's preferences and needs. For this survey 175 exercise items were selected (Figure 18) and 60% of them are cardio exercises. The Strength and Team Sport category contain 28 and 27 exercises respectively and we have 12 balance exercises. While this dataset is not balanced all categories contain at least 1 exercise for each intensity type. In terms of meals, each one of them contains 8 food items from 4 different classes: 1 from Breakfast, 3 from Small Snacks, 2 from Main Meal and 2 from Side Dish. We did not use the whole food data set because a lot of foods are unfamiliar to Greeks. We use 983 food items belonging to 4 different classes and 14 different categories. As Figure 18 shows class Main Meal contains 4 different food categories with the most food items (363) and then Side Dish with the same number of categories and 260 food items. Lastly Breakfast and Small Snacks contain 3 food categories with similar amounts of food items with 158 and 157 respectively. By combining 8 food items we create 3 menus for each user where each menu is made with the same number of generations but different weights on fitness function's parameters. The first menu focuses on the user's nutrients needs, the second on its preferences and the last one is balanced focusing on both needs and preferences. Users are not aware of these changes nor the focus of each menu.

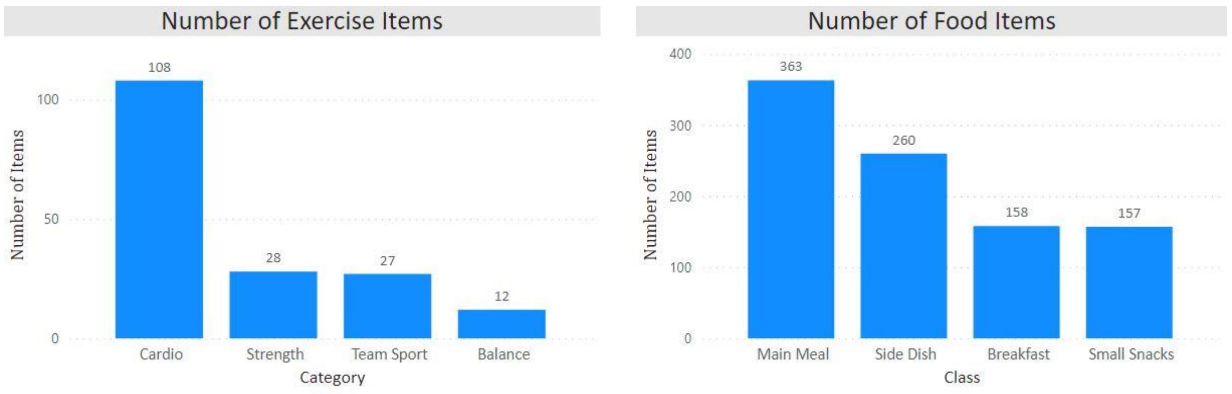


Figure 18: Number of exercise (left) and food (right) items per category that were used in this survey.

As mentioned above we provide users 3 different menus focused on 3 different targets. The difference between these menus is in fitness function parameters. Equation 4.3 at 4.2 have 4 different α parameters. For a balanced menu $[\alpha_1, \alpha_2, \alpha_{3,}\alpha_4] = [4,1,1,50]$. $\alpha_1 = 4$ because |*target* - *real*| values are between 0,001 and 4,999 while preferences values are between 1 and 5. When we tried having both variables equal to 1, we noticed that nutrients did not truly affect each bundle's fitness score and preferences were the ones that mattered. As a result, we add more weight to nutrients in order to build a fitness function that focuses equally on both preferences and nutrients and we set $[\alpha_1, \alpha_2, \alpha_{3,}\alpha_4] = [1,1,1,50]$ for the preference bundle. Lastly, we wanted the final bundle to focus only on nutrients and to be little affected by users' preferences. As a result, we set $[\alpha_1, \alpha_2, \alpha_{3,} \alpha_4] = [30,1,1,50]$ in order to make sure that food nutrients play a significant role in the final bundle recommendation. It is also noticeable that α_3 and $\alpha_4$are equal to 1 and 5 respectively for all occasions. This happens because they do not affect the focus of our bundles and due to their significance, we can't really change their values. Parameters α_3 is a binary variable and we either take into consideration the same main food rule (not eating the same main food for two consecutive days) or we don't. Parameters α_4 affect the amount of weight users' restrictions have on the final outcome. Especially for lactose intolerant and diabetic people we do not want to suggest food items that could lead to unpleasant results. So, it is vitally important for parameter α_4 to take a large value in order to not suggest foods that they should avoid. So, the goal here is to provide them three one-day diet plans and to rate them based on their preferences. But before recommending these diet plans, our volunteers had to fill a google form that contained 28 questions in order to learn more about their goal and preferences.

The first five questions were about their gender, age, height, weight and exercise intensity. We used the information of the first four questions in order to find the user's BMR using 3.1 equation. On the exercise intensity question users had to choose between low, moderate and high. Based on their answers we multiply BMR's result by a certain factor in order to find the daily calories needed to remain at their current weight. In addition, we asked about the user's target goal and they had to choose between losing, maintaining or gaining weight. As stated in 3.2.2 if they want to lose weight, we subtract 500 calories from the number calculated above and add 500 calories if they want to gain weight. Another question is about their involvement in a sport that needs high stamina and endurance. Depending on their answer we choose one of the two available programs as stated in 3.2.2. Then we ask them about any food restrictions and we provide them four possible answers: Lactose intolerant, Vegetarian, Diabetic and none from the above. Moreover, they have to grade all 15 food categories from 1 (not like) to 5 (like very much) so they can show their dislike or restriction in a certain food category by rating it with 1. Last but not least they rate by using the same grading system the four exercise categories: Balance, Cardio, Strength and Team sport exercises.

5.2 Results

5.2.1 General

Doing this survey, we tried to have volunteers of both genders and belonging to multiple age groups and, as seen in Figure 20, out of 50 participants 29 are women and 21 are men. In addition, 60% of our population are between 20-25 years old, almost 24% are between 26-35 and the rest are above 35 years old. While most of the population are around the author's age, however there is some diversity in age and gender. In addition, in two out of three age groups (except from 36+ category) gender percentages are almost 50-50. Seventeen out of thirty volunteers aging from 20-25 and six out of twelve are women.

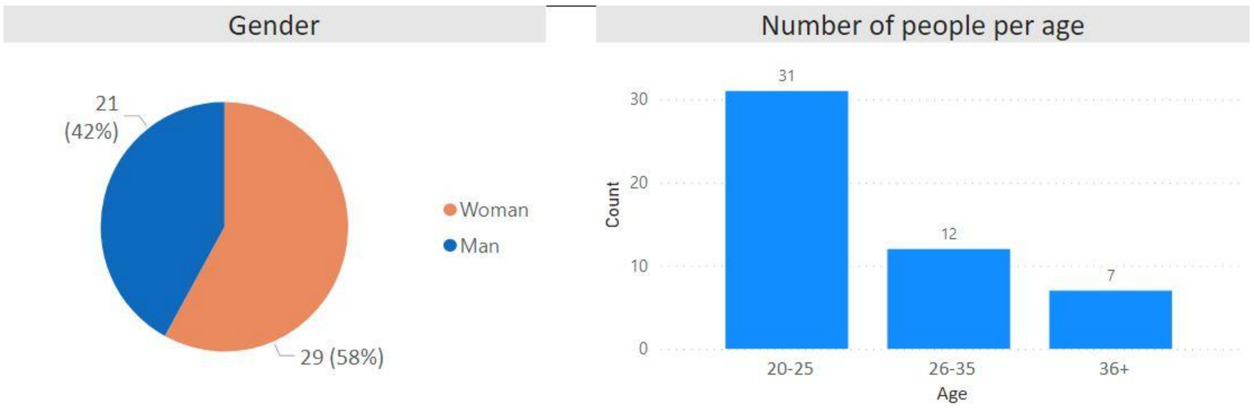


Figure 19: Gender and age of survey's participants.

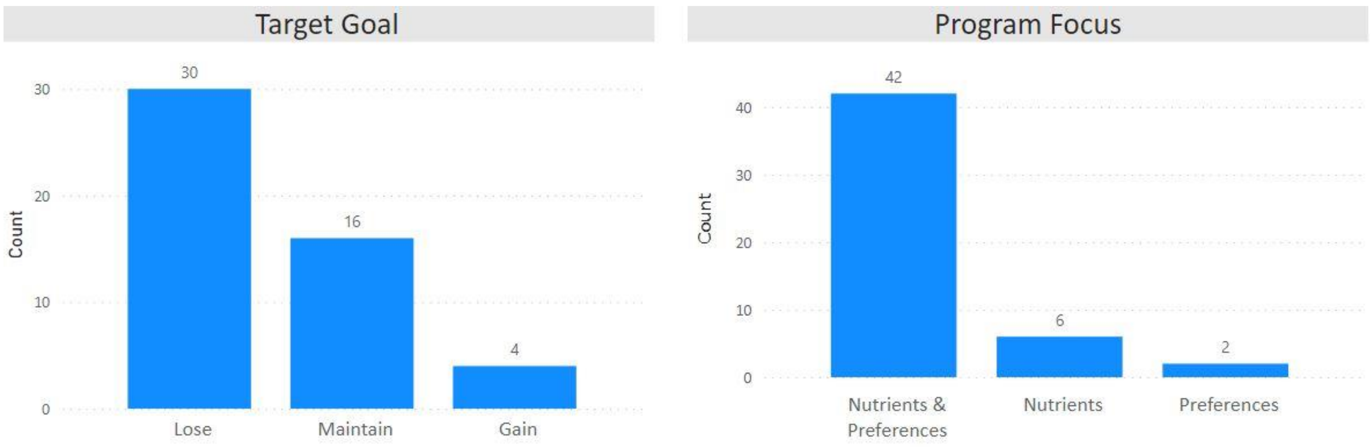


Figure 20: People's fitness goal and where they want their menu to be focused.

As mentioned before user's could choose between three different fitness goals which are to lose weight, maintain weight and gain weight. As seen in Figure 20 60% percent of our volunteers want to lose weight, 32% want to maintain weight and only 4 users want to gain weight. This was expected, as most people that visit a nutritionist (which I am not) want to lose weight and not to maintain or gain weight. The last question was about where their diet program should focus and almost all users choose the same answer. Forty-two out of fifty users want their menu to be focused on both preferences and nutrients, while six users care less about their preferences and they want a very accurate diet program. However, the real question is are all programs accurate in terms of the user's nutrient needs and how did users rate each program? And how many of them do they consider themselves as an athlete? This was an important question because athletes need more carbohydrates and less protein than regular users and they follow a completely different program.

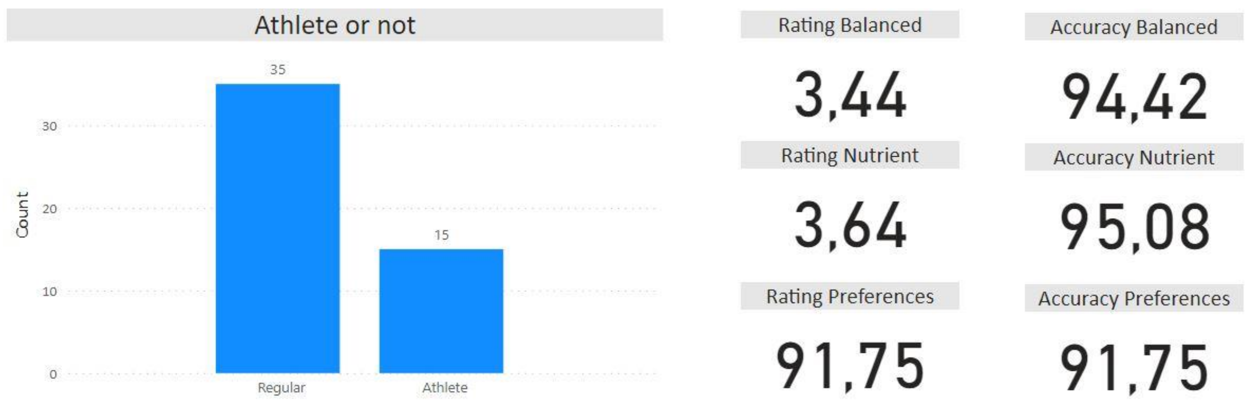


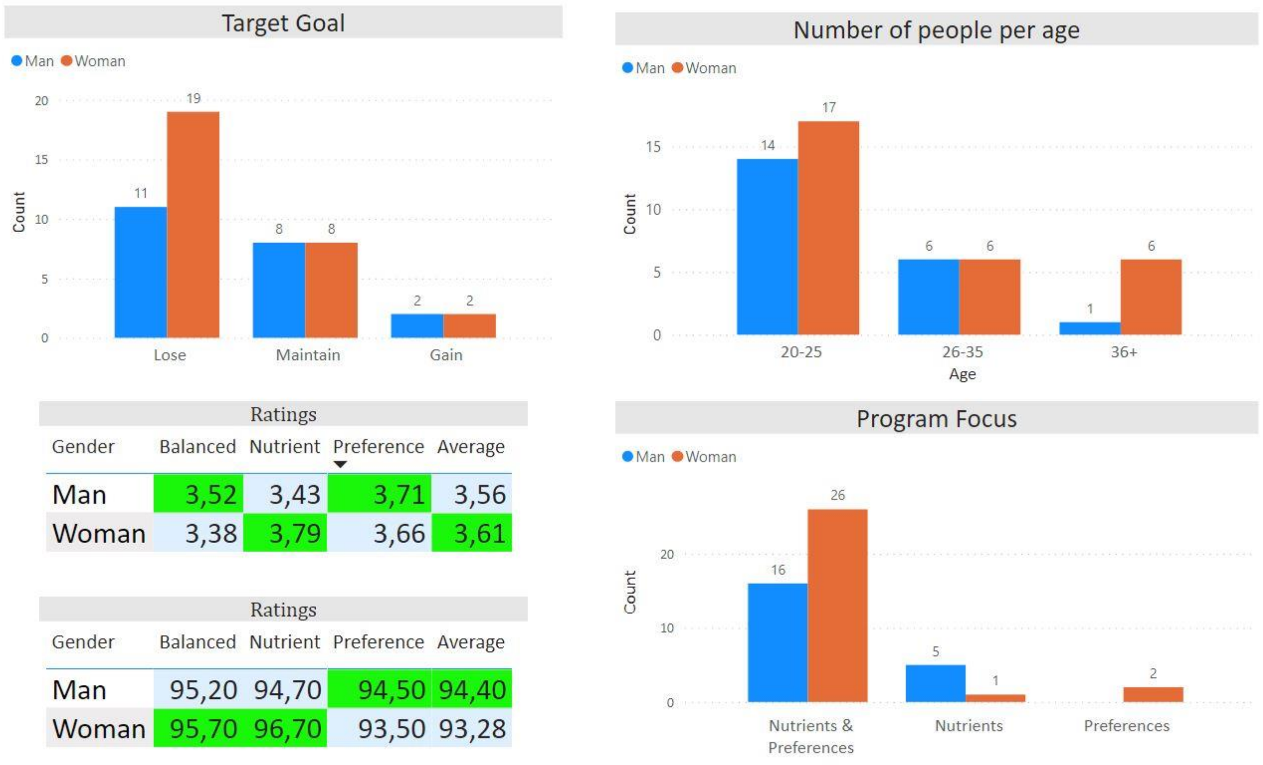
Figure 21: Number of people that consider themselves athletes. Also, average rating value and accuracy of each diet program.

Figure 21 provides us lots of information. At first the bar-plot shows that 35 out of 50 people got a 40-30-30 program that is meant for non-athlete people. As said above from daily calories 40% should be carbohydrate, 30% fat and 30% protein. The rest 15 volunteers consider themselves athletes and their program consist of 51% carbohydrate, 33% fat and 18% protein needs. Using this program, they will not see any negative effect on their endurance. This figure also provides the user's average rating and accuracy in each menu. As seen, the difference between these three average ratings is pretty small. We did not provide users the accuracy of each program in order to rate them based only on food quantities and food and exercise preferences. As expected, the bundle whose focus was on user's preferences was highest rated (3.68). In these menu's users saw their favorite meals and exercises and it was more than enough for a positive rating. However, the bundle that focused on food nutrient value was only 0.04 behind with 3.64 and its 3.33% more accurate. This could mean that either users were able to understand this bundle benefits, which is not probable because in most occasions all three menus had lots of similarities, or that these bundles are also accurate in terms of users' preferences. While we add more weight in food nutrients, fitness function still is being affected by user preferences. As a result, all three menus take into consideration both preferences and nutrients, but with different weights. Lastly while a balanced bundle is accurate and it takes into serious consideration user's preferences, it has the lowest average rating. We expected this bundle to have a similar rating with "preferences bundle" and larger from "nutrient bundle" due to its weight distribution. However only one question comes into our mind when we look at these results. So, which is the best bundle? We will try to answer this question by analyzing a bit more of these results.

5.2.2 Gender

Firstly, let's examine these results for each gender separately. As seen in Figure 22 the same amount of men and women want to maintain or gain weight. The rest want to lose weight and the difference between women and men is also the difference in this category. So, we have a balanced dataset of people in terms of target setting. In terms of age, 6 out of the 7 people that are above 36 years old are women. So, if we had analyzed data by age, this category would mostly be affected by women preferences and ratings. While in the other 2 age groups there is a similar number of women and men. In addition, while most men and women wanted their program to focus on both nutrients and preferences 23% (5 out of 21) of men and one woman wanted their program to focus only on nutrients. There were also 2 women that think it was very important to have a meal that they enjoy and focus on their preferences. Could this mean that it's important for some women to follow a diet plan that they enjoy even if it is less accurate in terms of needed nutrients?

Until now there is not a significant difference between genders in terms of goal setting, program focus preference or even age. So we could expect similar results in terms of rating. However, when we look at the rating table in Figure 22 there are noticeable differences.



Ratings

Gender	Balanced	Nutrient	Preference	Average
Man	3,52	3,43	3,71	3,56
Woman	3,38	3,79	3,66	3,61

Ratings

Gender	Balanced	Nutrient	Preference	Average
Man	95,20	94,70	94,50	94,40
Woman	95,70	96,70	93,50	93,28

Figure 22: Target goal, number of people per age category, program focus preference and bundles accuracy and ratings for each gender.

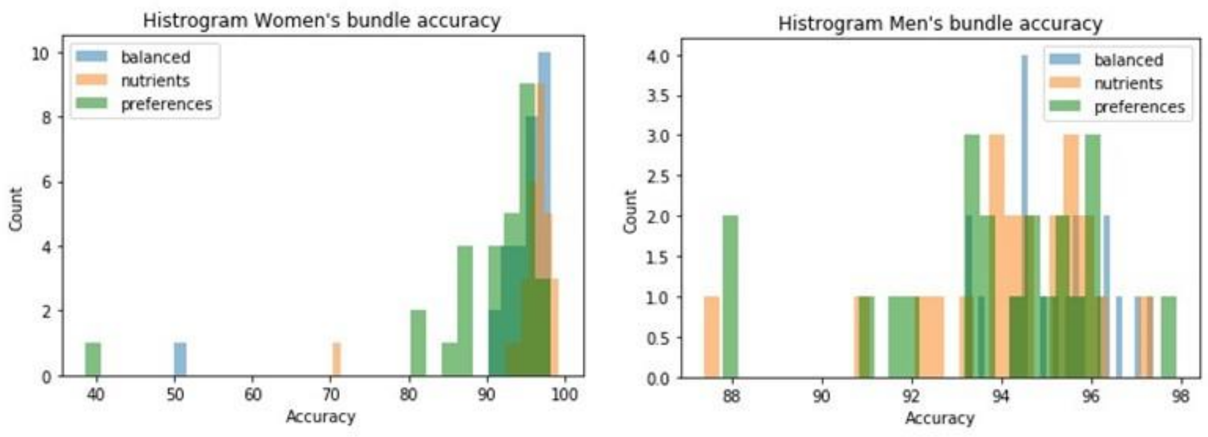
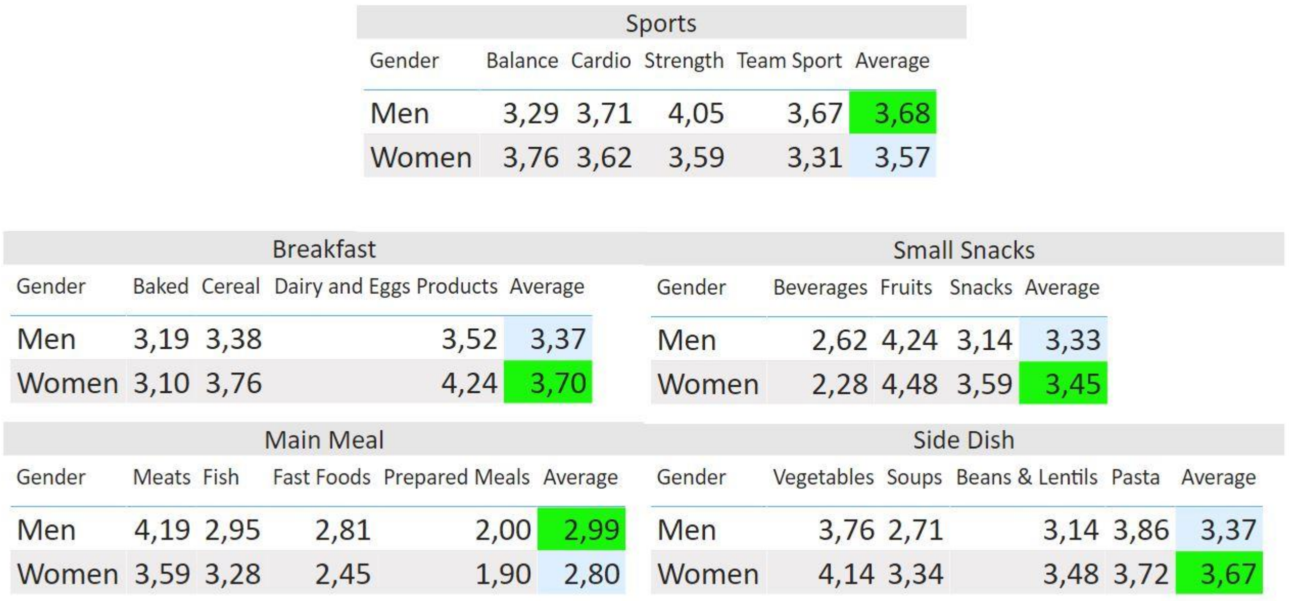


Figure 23: Histograms of women's and men's bundle accuracy per bundle category.

While women tend to rate a bit higher their rating is larger only in the nutrient category which is also their highest rating category. This category has also the highest accuracy for women and is 2% larger than the average accuracy of all 3 programs. So if we look at their program focus preference and their ratings this was not an expected result. As said above there was only one woman that wanted her diet plan to focus only on nutrients in order to reach her goal in the fastest and best way possible. However as said above users did not know the accuracy of each program, so they had to rate them based on their food and exercise preferences. It seems that for women, providing them a bundle that focuses on their nutrients not only covers their nutrient needs but it is also the one they will probably like more. So a bundle that focuses on nutrient needs can also be the one that users prefer as it also takes into consideration their preferences. On the other hand, men give lower average ratings but provide higher ratings on balanced and preference bundles. While preference bundles have the highest rating, they also have the lowest accuracy. There were no men that wanted a program that focused on preferences; however, they rated this bundle higher. This was an expected result when users rate a program based only on their preferences. However, it is not safe to assume that providing them a preference bundle will please them. Ratings and their preference on program's focus do not match. While the difference from the average rating for both genders is 0.15 or more, it is still not safe to assume that we reached a conclusion about the best bundle. It is also important to notice that looking only at the accuracy table it may seem that men's bundles tend to be more accurate than women. However, this is not entirely true, this result is due to a suggestion on a female volunteer that rated several food categories with 1 and added a restriction. As a result, our program did not have many options in order to form a correct meal which led to suggestions with low accuracy on food nutrients. We will refer to this incident later on this report.



Sports					
Gender	Balance	Cardio	Strength	Team Sport	Average
Men	3,29	3,71	4,05	3,67	3,68
Women	3,76	3,62	3,59	3,31	3,57

Breakfast				
Gender	Baked	Cereal	Dairy and Eggs Products	Average
Men	3,19	3,38	3,52	3,37
Women	3,10	3,76	4,24	3,70

Small Snacks				
Gender	Beverages	Fruits	Snacks	Average
Men	2,62	4,24	3,14	3,33
Women	2,28	4,48	3,59	3,45

Main Meal					
Gender	Meats	Fish	Fast Foods	Prepared Meals	Average
Men	4,19	2,95	2,81	2,00	2,99
Women	3,59	3,28	2,45	1,90	2,80

Side Dish					
Gender	Vegetables	Soups	Beans & Lentils	Pasta	Average
Men	3,76	2,71	3,14	3,86	3,37
Women	4,14	3,34	3,48	3,72	3,67

Figure 24: Average preferences of men and women in each food or sports category.

Figure 23 provides us two bundle's accuracy histograms, one for each gender, per bundle's category. Green color represents bundles that focus on users' preferences, orange on nutrients and blue represents the balanced version of the other two. As said above women's histogram have three bundles (one from each category) that are below 80% accuracy. These three bundles formed as a suggestion to one specific user that we will refer to later on. There are also 8 preference bundles below 90% accuracy. This histogram tells the same story as the average accuracy values in Figure 22. That preference bundle provides the least nutrients accuracy on bundles suggested to women. On the other hand, nutrient bundles are all (except one) above 92% and all balanced bundles above 89%. Similarly in men's histogram all except one nutrient bundle are above 90%. In addition, all preference bundles are above 88% and all balanced bundles above 92%. So, except one occasion all nutrient and balanced bundles can be considered accurate for both genders. Preference bundles provide less accuracy in some occasions especially for women.

However, the bundle's accuracy is correlated with users' preferences. If a user rates highly a lot of food categories then our program has a lot of options from which it can choose in order to build a plant that fits better each user's needs. Figure 24 shows the average ratings of each food and exercise category per gender. As mentioned above each category belongs to one class from where we choose in order to formulate a meal. For example, in order to propose a lunch or dinner meal we choose one food from the Main Meal category and one from Side Dish. From the sports table we can observe that men's ratings are higher in all exercise categories except balance compared with women's ratings. So, most men will probably rate positively diet plans that contain strength exercises and negatively plans that provide them balance exercises like Yoga. On the other hand, most women prefer balance exercises and do not like to be involved in team sports like football or basketball. However, both genders rate in general all 4 exercise categories above 3 which could mean that they are willing to try all kinds of exercises. Female participants rated on average higher three out of four classes than men. But let’s talk about each class separately. Breakfast contains three categories: Baked foods, Cereal and Dairy & Egg Products. In this category women rated positively Cereal and Dairy & Egg Products with ratings above 3.5 and baked foods with 3.10. So when our program will suggest foods from breakfast class it will probably not choose from the baked foods category due to its lower rating. Men on the hand rated similarly all food categories providing more options for a breakfast suggestion. The second class is called Small Snacks and it contains beverages, fruits and snacks. We choose food items from this class for the user's breakfast (1 out of 2 foods items), midmorning and afternoon meals. So it’s a class that impacts more than 30% of the final results because three out of nine food items belong to this class. Almost all users rated similarly here by providing beverages a small rating (below 3), snacks a medium rating and fruits their higher rating. So, on most occasions a fruit or a juice was being suggested. However, we added a restriction where if a user has rated with more than 3 two or more categories then food items from the Small Snack class that are suggested on breakfast and mid-morning should not belong to the same category. Similarly, food items from the Small Snack class that are suggested on mid-morning and afternoon meals should not belong to the same category. So, users will not get bored by eating similar foods all day. For example, there could be a 1-day meal in which users had to drink an orange juice for breakfast, watermelon for mid-morning break and eat apples in the afternoon. This suggestion contains 2 dangers. At first it probably would not provide enough protein and fat to users and in addition users would definitely not want to eat another fruit the next day. So, adding this restriction we are making sure that users will not burn out by eating lots of similar foods in a day. In terms of lunch and dinner men provided an average rating above 4 for meats and below 3 for the other three categories in Main Meal class. So, for most men it was probable that the main ingredient of both lunch and dinner contained meat. The restriction for both genders was that lunch and dinner could not have the same main meal. For example, if lunch contains pork with potatoes, then dinner could be any other meat or Main Meal except pork. It could however have the same side dish which, in this occasion, is potatoes. Women rated similarly with men, but the difference in average ratings between fish and meat is only 0.31 while in men it is 1.14. So, our model has more options when choosing a Main Meal for women than men due to smaller rating differences. Both genders provided the lowest rating at fast food and prepared meals (precooked) that, in general, provided less useful nutrients. Lastly in Side Dish class women on average rated above 3 all four categories while men pasta, vegetables and Beans & Lentils above 3 and Soups below 3. There was no restriction for this class and while women tend to rate higher the Side Dish options were similar for both genders. To sum up, while using the user's gender in order to categorize them provides us much information, there is certainly not enough in order to choose which bundle fits best for each user. The differences between male and female users are not enough in order to reach a safe conclusion about the best bundle category.

5.3 Target & Athletic condition

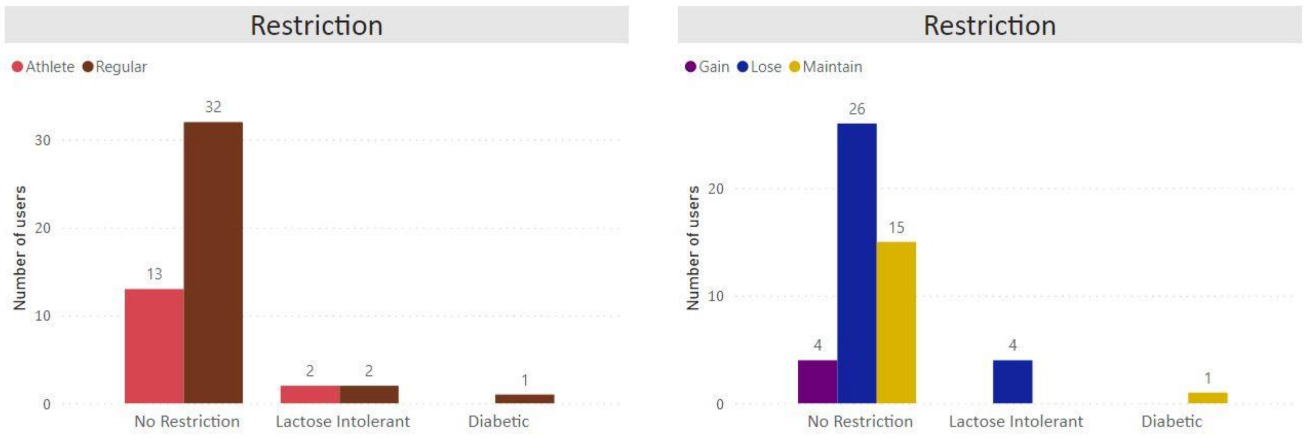
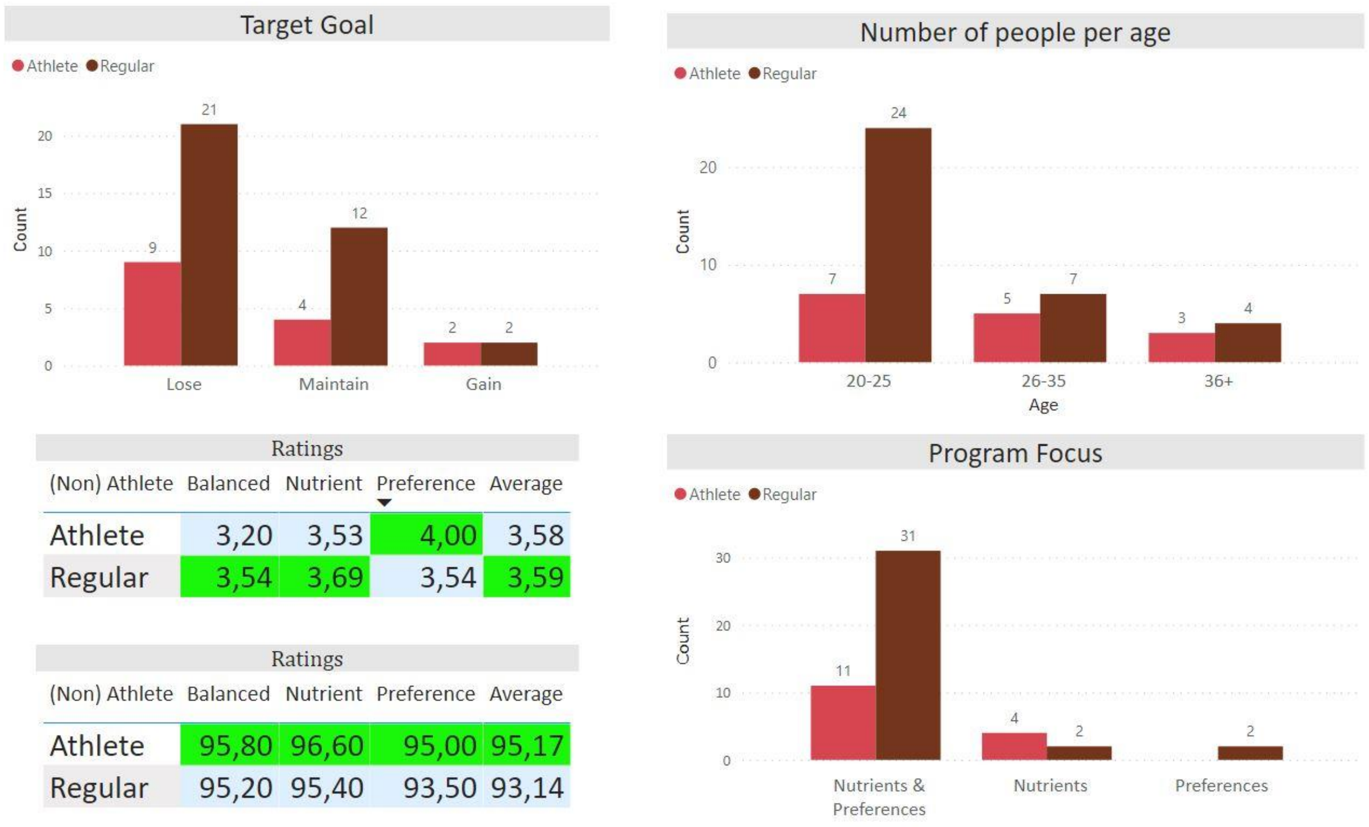


Figure 25: Number of people in each restriction category. On the left picture users are categorized by their athletic behavior and on the right by weight goal.

While gender provides us information about each bundle, we do not think is a correct way to separate users. We certainly have to add more parameters in order to better categorize them. One of the most important parameters is the user's goal. As seen in Figure 20 thirty users would like to lose weight, sixteen to maintain their weight and only four to gain weight. Another important parameter is restriction however we can not make a good analysis based on this factor. As Figure 25 shows us we do not have enough users that are lactose intolerant or have diabetes. It is not safe to make assumptions about a category by using only one user's opinion. On the other hand 15 out of 50 users consider themselves athletes and were provided with different programs than non-athlete users (Regulars). It is safe to start this analysis by comparing athletes with non athletes then users with different target goals and lastly combining these two categories. After these steps we will be confident enough to reach safer conclusions than the ones expressed in the previous section.

5.3.1 Athlete vs Regular

Figure 26 provide us the main information about users' classification based on their athletic abilities. With red are depicted the people that consider themselves as athletes and with brown the non-athletes. As Figure 26 shows us this dataset contains 35 regular and 15 athlete users. Figure 26 shows that 60% of athletes and 60% of regular users want to lose weight. Also 4 out of 25 athletes and 12 out of 35 regular users want to maintain their weight and lastly 2 people from each category want to gain weight. The interesting thing here is that we have a similar percentage of users in each category so we can conduct an informative analysis. Similarly, we have athletes and non-athletes in all age categories. As seen 50% of athletes are below 26 years old, 5 are between 26 and 35 and the rest are above 35 years old. However, for regular users 75% of them are below 26 years old, 22% between 26 and 35, and the rest 12.5% is above 35 years old. While in this case athletes and non-athletes are not similarly separated in each age category, there are enough in order to reach conclusions. The same Figure also shows in which parameter they wanted the program to be focused on based on their athletic behavior. Before moving to results from Figure 26 we should discuss users' preferences.



Ratings

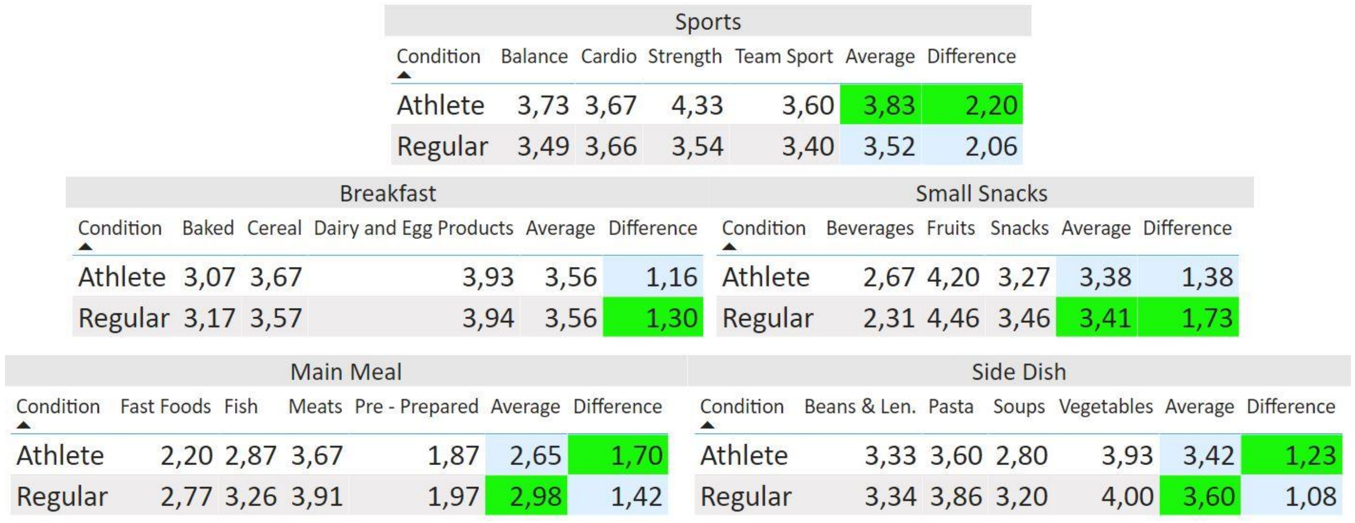
(Non) Athlete	Balanced	Nutrient	Preference	Average
Athlete	3,20	3,53	4,00	3,58
Regular	3,54	3,69	3,54	3,59

Ratings

(Non) Athlete	Balanced	Nutrient	Preference	Average
Athlete	95,80	96,60	95,00	95,17
Regular	95,20	95,40	93,50	93,14

Figure 26: At top left is depicted the weight goal for each user. At top right we classify users by age. Bottom left are two arrays that show each program average accuracy and users rating. Bottom right depicts the user's willingness on where their program should focus.

As seen in Figure 27 athletes have rated all sport categories higher than regular users. In the strength category this difference is 0,7 from the corresponding value of regulars and 0,5 from the average value. So, these athletes prefer exercises from the strength category and it is highly probable that at least 1 out of 3 suggested bundles contain a strength exercise. However, in order to provide lots of options to our program, it is important to rate a lot of categories similarly and not some categories high and some others low. The highest difference between these ratings is the lower options our program has. So regular users provide more options in terms of sports. All breakfast and small snacks categories were rated similarly by both types of users. Average breakfast value of athletic users is equal to the corresponding breakfast value of non-athletic users and the difference from the small snacks category is only 0,03. In both the difference in athletes rating is smaller than the corresponding of regular users by 0,14 and 0,25 respectively. So, this time for athletes, in general, there are more breakfast and small snacks food items with similar probability to be chosen. Furthermore, for main meal and side dish categories we have similar rating distribution for both user types. Regular users' average ratings are larger in both classes while having more food items with similar probability due to lower difference values. Adding all the average differences together we get 7,67 for athletes and 7,59 for Regular users. While this difference is insignificant, the difference in breakfast and small snacks may be important due to the small number of food items that belong to these categories.



Sports						
Condition	Balance	Cardio	Strength	Team Sport	Average	Difference
Athlete	3,73	3,67	4,33	3,60	3,83	2,20
Regular	3,49	3,66	3,54	3,40	3,52	2,06

Breakfast					
Condition	Baked	Cereal	Dairy and Egg Products	Average	Difference
Athlete	3,07	3,67	3,93	3,56	1,16
Regular	3,17	3,57	3,94	3,56	1,30

Small Snacks					
Condition	Beverages	Fruits	Snacks	Average	Difference
Athlete	2,67	4,20	3,27	3,38	1,38
Regular	2,31	4,46	3,46	3,41	1,73

Main Meal						
Condition	Fast Foods	Fish	Meats	Pre - Prepared	Average	Difference
Athlete	2,20	2,87	3,67	1,87	2,65	1,70
Regular	2,77	3,26	3,91	1,97	2,98	1,42

Side Dish						
Condition	Beans & Len.	Pasta	Soups	Vegetables	Average	Difference
Athlete	3,33	3,60	2,80	3,93	3,42	1,23
Regular	3,34	3,86	3,20	4,00	3,60	1,08

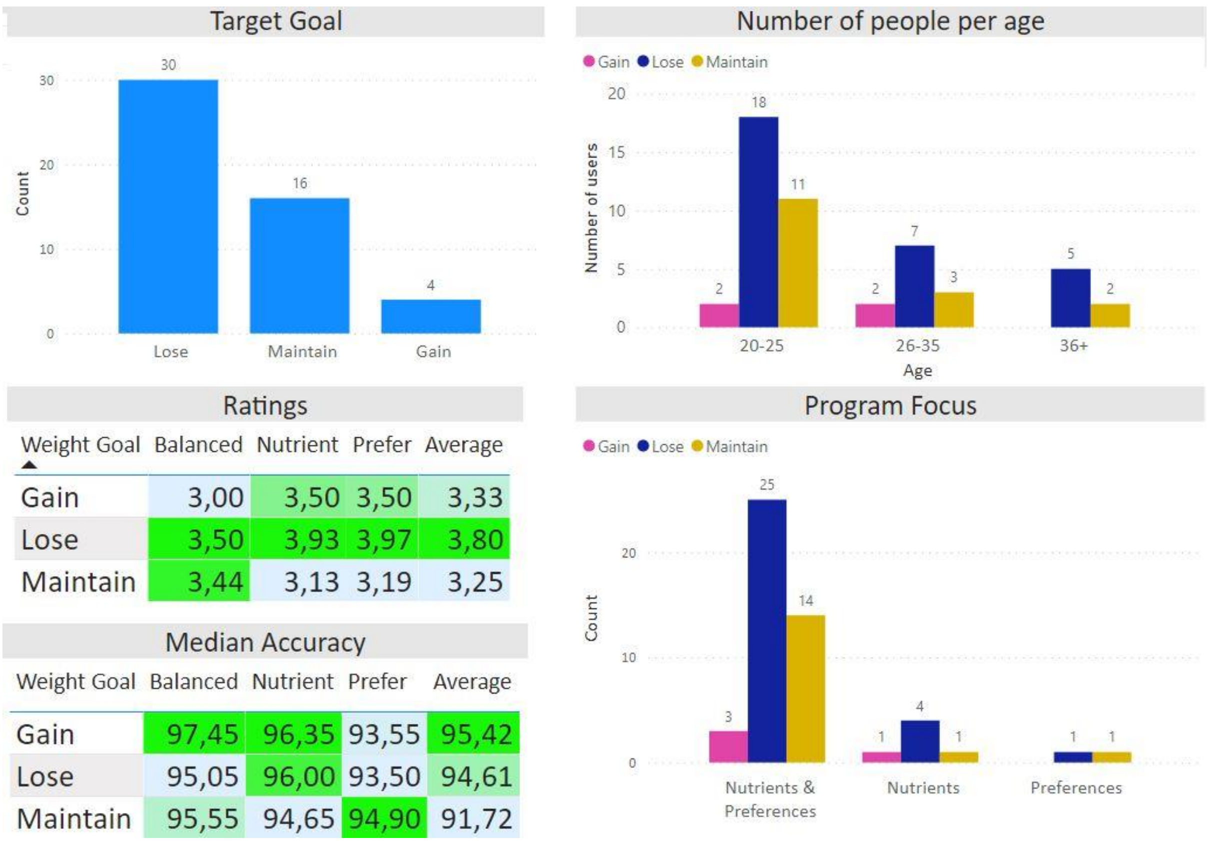
Figure 27: Average preference values of athlete and non-athlete (regular) users.

In terms of accuracy between respective bundles of athletes and non-athlete users, figure 26 shows that bundles suggested to athletes are more accurate in all categories. The minimum difference in accuracy between these bundles is 1,7 in Nutrient and maximum 2,56 in Preferences. In addition, bundles that focus on nutrients provide, in general, suggestions with higher accuracy for both types of users while bundles focused on preferences have the lowest accuracy. Moreover, we notice that the average accuracy of balanced bundles for athletes, which is between nutrient and preference bundles accuracy, is larger than average accuracy of each bundle of regular users. Why are athlete's bundles more accurate than the ones of regular users? Could it be because these two categories have different nutrient distribution? The answer is yes this could be a possibility. As stated in 3.2.2 athletes' calories are distributed as 51% carbohydrates, 32% fat and 17% protein while non athletes 40% carbohydrate, 30% fat and 30% protein. So, there is a significant difference between the quantities of carbohydrate and protein that each program provides. Another reason could be the rating of breakfast and small snacks classes, especially for bundles that focus on user's preferences. As said above athletes provide more options to our program by providing similar ratings to these categories. When two ratings of different food categories are equal, then they equally affect fitness so it is like eliminating their significance on the final result. As a result, the bundle with higher nutrient value is selected.

However, bundles with higher accuracy does not always mean that they are more likeable to users. Average ratings of all bundles are almost equal for both types of users with 3,58 for athletes and 3,59 for regular users. Furthermore, regular users provide, on average, higher ratings to balanced and nutrient bundles with 3,54 and 3,69 than athletes with 3,20 and 3,53 respectively. However, athletes rated with 4,0 the preference bundles while regular users with 3,54. The 15 athletes that took part in our survey definitely prefer bundles that focus on their preferences as the difference between nutrient categories is 0,47. Even so none of them wanted their program to focus on their preferences. Providing each bundle accuracy could change the rating of at least 4 people that want a diet program with high nutrient value. On the other hand, taking into consideration that for athletes all programs have average nutrient accuracy above 93% could make them keep their ratings the same despite knowing each bundle accuracy. Regular users rated all bundles similarly with balanced and preference bundles having the same average rating and nutrient bundles being larger by 0,15. So suggesting regular users their nutrient bundle, we provide a diet plan that they prefer and also fits their nutrient needs. In general balanced bundles again seem to get the lowest rating for both types of users, meaning that this function provides the least favorable bundles so far.

5.3.2 Target Goal

Another important factor that we should take into consideration is the weight goal of each user. There are 3 different weight goals: Gain, Lose and Maintain their body weight. After finding the user's BMR and multiplying this number by a value depending on exercise intensity, then we either keep this value (maintain) or subtract/add (lose/gain) 500 calories. So is there a difference in accuracy or rating depending on the user's goal?



Ratings

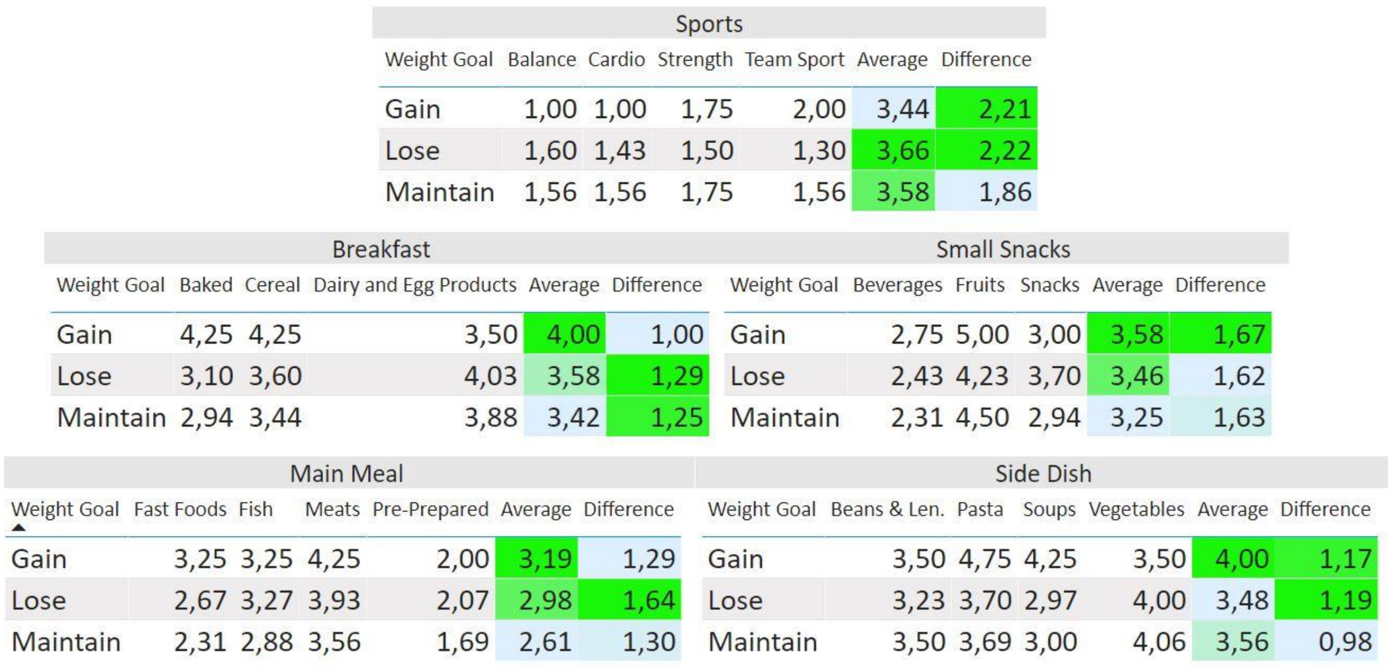
Weight Goal	Balanced	Nutrient	Prefer	Average
Gain	3,00	3,50	3,50	3,33
Lose	3,50	3,93	3,97	3,80
Maintain	3,44	3,13	3,19	3,25

Median Accuracy

Weight Goal	Balanced	Nutrient	Prefer	Average
Gain	97,45	96,35	93,55	95,42
Lose	95,05	96,00	93,50	94,61
Maintain	95,55	94,65	94,90	91,72

Figure 28: Similar diagrams as Figure 26. Instead of categorizing athletic conditions, we categorize users by their weight goal.

As previously mentioned, 30 users want to lose weight, 16 to maintain and 4 to gain weight in a healthy way. Lose and maintain weight users belong with similar proportions in all age clusters as Figure 28 shows. Two users that want to gain weight are between 20 and 25 years old and two are between 26 and 35 years old. So except for the users that want to gain weight, users that belong to one of the other two categories belong to a similar proportion in all age clusters. Moreover, no matter the target, users will probably want their program to focus on both nutrients and preferences as seen at the bottom right column chart of Figure 28. Also, in this section before we discuss each bundle accuracy and rating, we should see the average rating of each user based on their weight goal.



Sports

Weight Goal	Balance	Cardio	Strength	Team Sport	Average	Difference
Gain	1,00	1,00	1,75	2,00	3,44	2,21
Lose	1,60	1,43	1,50	1,30	3,66	2,22
Maintain	1,56	1,56	1,75	1,56	3,58	1,86

Breakfast

Weight Goal	Baked	Cereal	Dairy and Egg Products	Average	Difference
Gain	4,25	4,25	3,50	4,00	1,00
Lose	3,10	3,60	4,03	3,58	1,29
Maintain	2,94	3,44	3,88	3,42	1,25

Small Snacks

Weight Goal	Beverages	Fruits	Snacks	Average	Difference
Gain	2,75	5,00	3,00	3,58	1,67
Lose	2,43	4,23	3,70	3,46	1,62
Maintain	2,31	4,50	2,94	3,25	1,63

Main Meal

Weight Goal	Fast Foods	Fish	Meats	Pre-Prepared	Average	Difference
Gain	3,25	3,25	4,25	2,00	3,19	1,29
Lose	2,67	3,27	3,93	2,07	2,98	1,64
Maintain	2,31	2,88	3,56	1,69	2,61	1,30

Side Dish

Weight Goal	Beans & Len.	Pasta	Soups	Vegetables	Average	Difference
Gain	3,50	4,75	4,25	3,50	4,00	1,17
Lose	3,23	3,70	2,97	4,00	3,48	1,19
Maintain	3,50	3,69	3,00	4,06	3,56	0,98

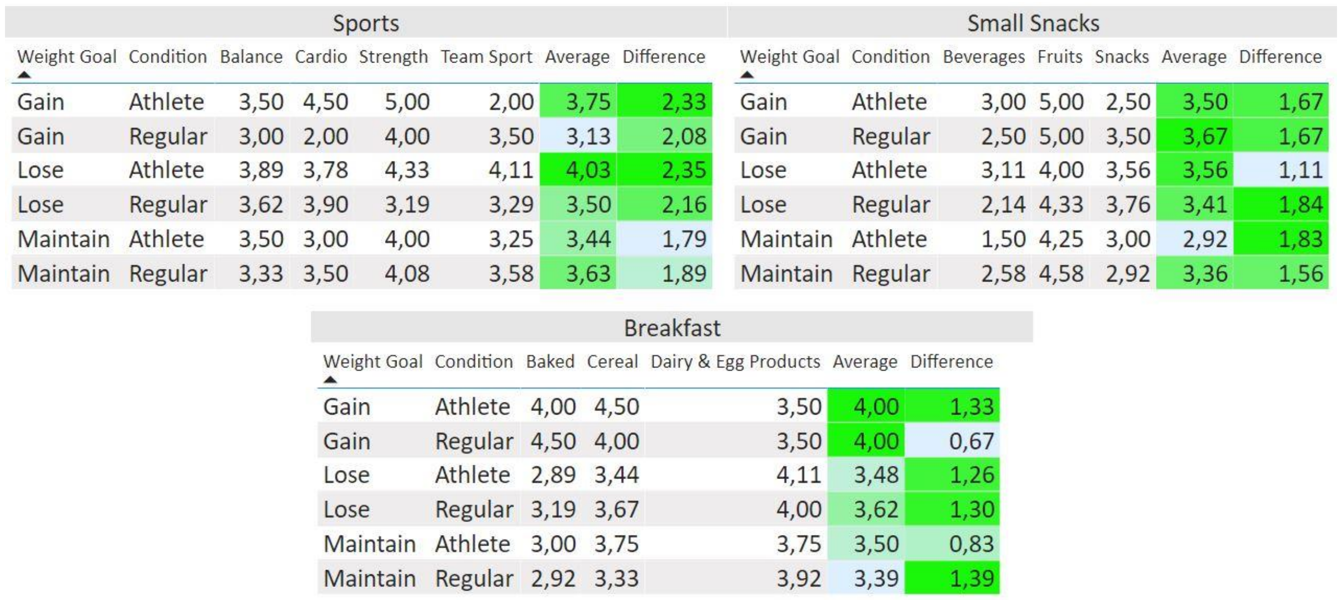
Figure 29: Average ratings on certain categories based on users' weight goal.

As depicted in Figure 29 users rated each sport category differently. Users that want to increase their body mass, also want to increase their strength by rating with 4.5 strength exercises. Users that want to decrease their weight rated all exercise categories similarly while maintain weight users prefer exercises that focus either on strength or are a Team Sport. While average difference is equal in both gain and lose weight users lose weight users provide more exercise options to the program as all categories are rated similarly. In addition, maintain weight users have the lowest difference but on average these bundles will contain more Strength and team sport exercises. In breakfast class, gain weight users enjoy food items that belong to either baked foods or breakfast & cereal. Lose weight users rated dairy and egg products highly and then breakfast and cereal with 0,4 rating difference from the first. From all other classes one category stands out for each weight goal class. For example, in the main meal class gain weight users rated with 4.25 meats and all the other categories above 3.5 and similarly did lose weight users for vegetables in side dish class. However, which users provide, in general, more options to our program. Maintain weight users have the lowest difference value in 4 out of 5 classes and then come gain weight users. Does providing more options lead to programs with higher accuracy?

The answer is not clear. Figure 28 shows that balanced bundles for gain weight and maintain weight users present higher accuracy from nutrient bundles. Nutrient and balanced bundles are highly accurate user types while preference bundles present the lowest accuracy for gain and lose weight users. Maintain weight provides the highest rating, on average, to balance bundles which is also the category with the highest accuracy. This is where also most maintain weight users wanted their program to focus. Moving on, lose weight users provide similar ratings to nutrient and preference bundles that have the highest and lowest accuracy respectively. Lastly gain weight users give an average of 3,5 to both nutrient and preference bundles with nutrient bundles to be very accurate. So, in general bundles from nutrient and accurate categories present high nutrient accuracy for all users (median above 94%). Combining nutrient accuracy and bundles rating we choose balance bundles for maintain weight users and Nutrient bundles for lose and gain weight users. However, can we categorise each user more in order to give a suggestion that best fits users' needs and preferences?

5.3.3 Combination

Top left bar plot at Figure 26 shows the number of regular and athletes for each weight goal. In this section we categorize users by their athletic condition and also by their weight goal. So, this time we have 6 different categories instead of 3. In the lose weight category we have 21 regular users and 9 athletes, in maintain category 12 regular users and 4 athletes and lastly, we have 2 athletes and 2 regular users that want to gain weight. We have already discussed athletes and about each weight goal program so we will discuss users' ratings on certain categories.



Sports

Weight Goal	Condition	Balance	Cardio	Strength	Team Sport	Average	Difference
Gain	Athlete	3,50	4,50	5,00	2,00	3,75	2,33
Gain	Regular	3,00	2,00	4,00	3,50	3,13	2,08
Lose	Athlete	3,89	3,78	4,33	4,11	4,03	2,35
Lose	Regular	3,62	3,90	3,19	3,29	3,50	2,16
Maintain	Athlete	3,50	3,00	4,00	3,25	3,44	1,79
Maintain	Regular	3,33	3,50	4,08	3,58	3,63	1,89

Small Snacks

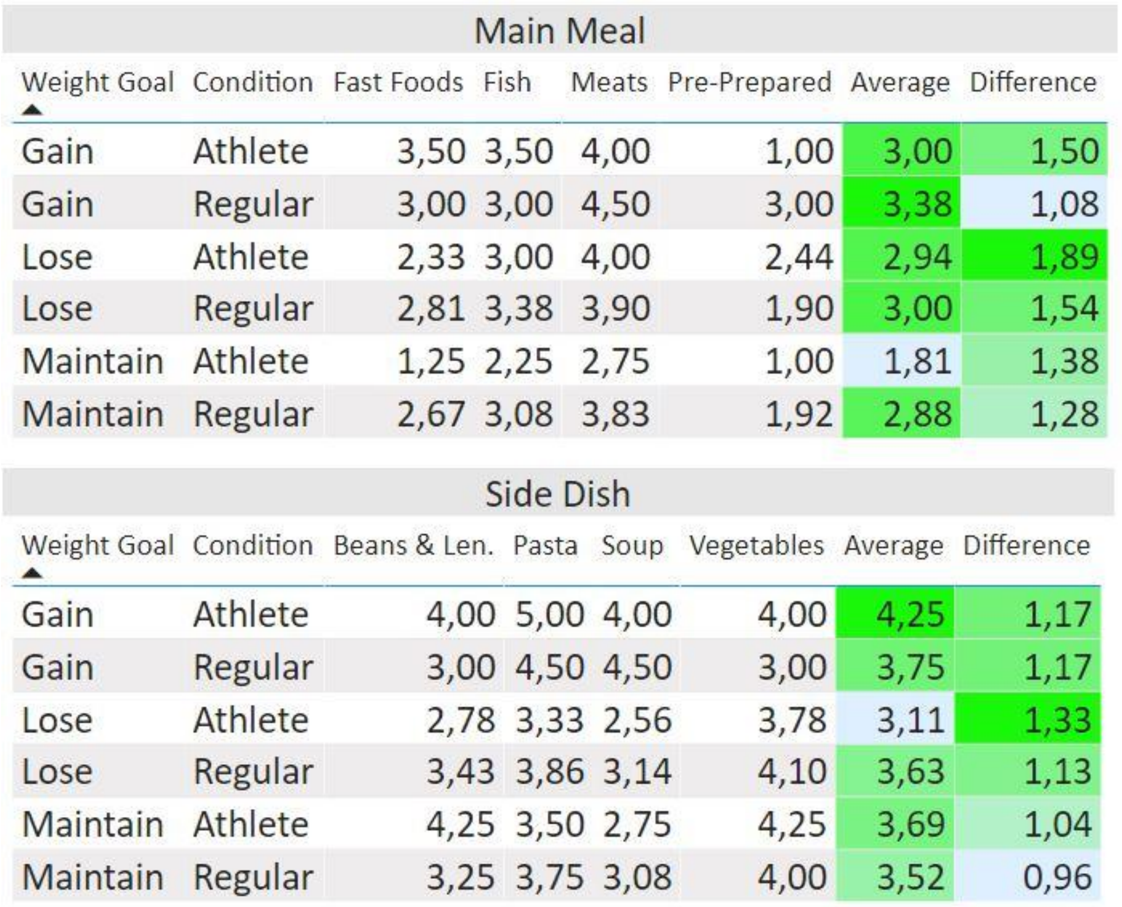
Weight Goal	Condition	Beverages	Fruits	Snacks	Average	Difference
Gain	Athlete	3,00	5,00	2,50	3,50	1,67
Gain	Regular	2,50	5,00	3,50	3,67	1,67
Lose	Athlete	3,11	4,00	3,56	3,56	1,11
Lose	Regular	2,14	4,33	3,76	3,41	1,84
Maintain	Athlete	1,50	4,25	3,00	2,92	1,83
Maintain	Regular	2,58	4,58	2,92	3,36	1,56

Breakfast

Weight Goal	Condition	Baked	Cereal	Dairy & Egg Products	Average	Difference
Gain	Athlete	4,00	4,50	3,50	4,00	1,33
Gain	Regular	4,50	4,00	3,50	4,00	0,67
Lose	Athlete	2,89	3,44	4,11	3,48	1,26
Lose	Regular	3,19	3,67	4,00	3,62	1,30
Maintain	Athlete	3,00	3,75	3,75	3,50	0,83
Maintain	Regular	2,92	3,33	3,92	3,39	1,39

Figure 30: Average ratings on sports, small snacks and breakfast classes and their categories based on users weight goal and athletic condition..

Athletes that want to gain weight or maintain weight have chosen 2 sport categories to stand out from the rest (Figure 30). At first athletes that want to gain or maintain weight rated two categories higher than the rest. Athletes that want to lose weight rated all sport categories similarly having also the highest average rating in this class. Gain and maintain weight regular users have chosen strength exercises as the ones they enjoy the most while lose weight regular users rated positively cardio and balance exercises. In the small snacks category, Fruits is the top rated category in all categories with 4 as the smallest average rating for athletes that want to lose weight. Except for athletes that want to gain weight snacks was the second rated small snack category having however a significant difference from the first. Athletes that want to lose weight seem to provide the most options to our model by having the smallest rating difference between small snacks categories. In the breakfast category gain weight users rated positively all categories while maintain and lose weight users gave higher average ratings to dairy & egg products. Maintain weight athletes and gain weight regular users have the lowest average difference with 0,83 and 0,67 respectively.



Main Meal

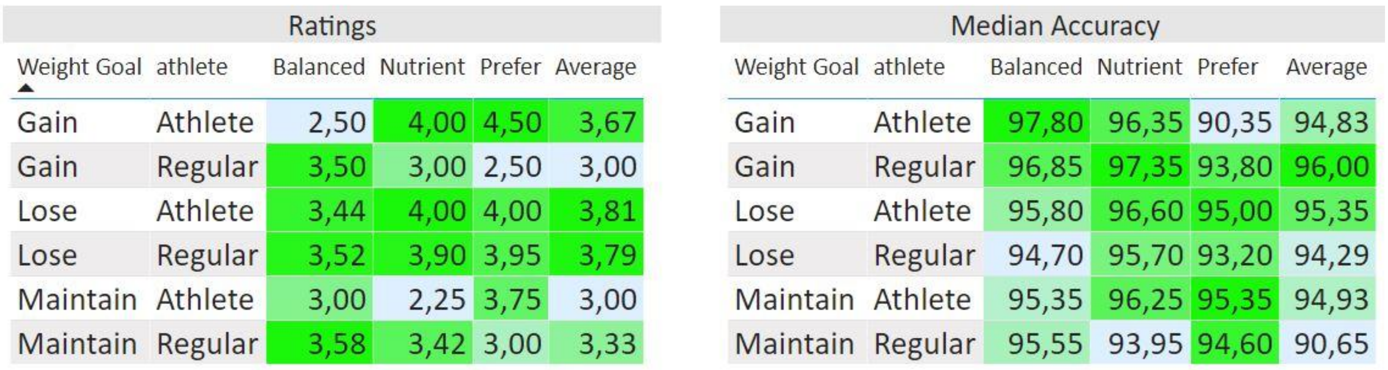
Weight Goal	Condition	Fast Foods	Fish	Meats	Pre-Prepared	Average	Difference
Gain	Athlete	3,50	3,50	4,00	1,00	3,00	1,50
Gain	Regular	3,00	3,00	4,50	3,00	3,38	1,08
Lose	Athlete	2,33	3,00	4,00	2,44	2,94	1,89
Lose	Regular	2,81	3,38	3,90	1,90	3,00	1,54
Maintain	Athlete	1,25	2,25	2,75	1,00	1,81	1,38
Maintain	Regular	2,67	3,08	3,83	1,92	2,88	1,28

Side Dish

Weight Goal	Condition	Beans & Len.	Pasta	Soup	Vegetables	Average	Difference
Gain	Athlete	4,00	5,00	4,00	4,00	4,25	1,17
Gain	Regular	3,00	4,50	4,50	3,00	3,75	1,17
Lose	Athlete	2,78	3,33	2,56	3,78	3,11	1,33
Lose	Regular	3,43	3,86	3,14	4,10	3,63	1,13
Maintain	Athlete	4,25	3,50	2,75	4,25	3,69	1,04
Maintain	Regular	3,25	3,75	3,08	4,00	3,52	0,96

Figure 31: Average ratings on main meal, side dish and their categories based on users' weight goal and athletic condition

In class main meal, meat is the top-rated category for all user types while 1 out of the two athletes that want to maintain their weight rated all categories with 1 so they have average rating 1,81. Prepared meals is the lowest rated category in all user types except athletes that want to lose weight. The same athletes also have the highest average difference between their ratings in this class meaning that their ratings affect significantly each bundle fitness. While average ratings on mail meal are close to 3, in side dish all average ratings are above 3,5 except lose weight athletes. This class contains foods that almost all are likeable to users providing lots of large search space to our model. All user types except lose weight users present average difference below 1,2 which means that a bundle can contain foods from certain categories and also have high accuracy.



Ratings

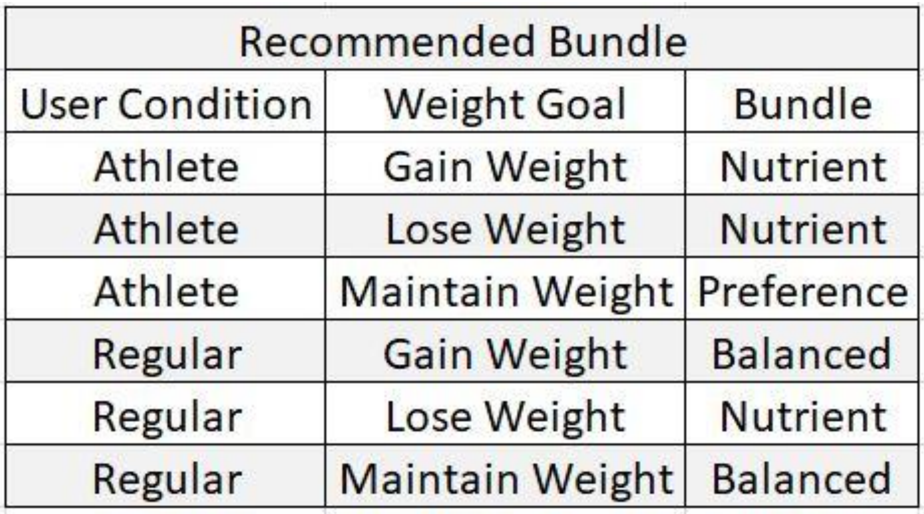
Weight Goal	athlete	Balanced	Nutrient	Prefer	Average
Gain	Athlete	2,50	4,00	4,50	3,67
Gain	Regular	3,50	3,00	2,50	3,00
Lose	Athlete	3,44	4,00	4,00	3,81
Lose	Regular	3,52	3,90	3,95	3,79
Maintain	Athlete	3,00	2,25	3,75	3,00
Maintain	Regular	3,58	3,42	3,00	3,33

Median Accuracy

Weight Goal	athlete	Balanced	Nutrient	Prefer	Average
Gain	Athlete	97,80	96,35	90,35	94,83
Gain	Regular	96,85	97,35	93,80	96,00
Lose	Athlete	95,80	96,60	95,00	95,35
Lose	Regular	94,70	95,70	93,20	94,29
Maintain	Athlete	95,35	96,25	95,35	94,93
Maintain	Regular	95,55	93,95	94,60	90,65

Figure 32: Average bundle rating and median accuracy of each user type.

Figure 32 provide us information about each bundle category average rating and accuracy. Also, these stats are categorized per user athletic condition and weight goal. Firstly, for all categories median nutrient accuracy is above 90%. So, we could say that all suggestions provide a pretty good result in terms of nutrients to the user. As could also be expected, preference bundles have the lowest median accuracy in almost all categories except regular users that want to maintain their weight. In this category that contains 4 users, nutrient bundles have the lowest median accuracy. This could happen due to the fact that for our calculations we used normalized values while we measured accuracy using actual nutrient values. Furthermore, weight athletes have the highest average accuracy followed by lose weight athletes. Balanced bundles present the highest accuracy in gain weight athletes followed by gain weight regular users. The latter type of users also has the highest nutrient bundle accuracy only 0,7 higher from lose weight athletes. Preference bundles are more than 93% accurate in all categories except for weight athletes that are 90,35% accurate.



Recommended Bundle		
User Condition	Weight Goal	Bundle
Athlete	Gain Weight	Nutrient
Athlete	Lose Weight	Nutrient
Athlete	Maintain Weight	Preference
Regular	Gain Weight	Balanced
Regular	Lose Weight	Nutrient
Regular	Maintain Weight	Balanced

Figure 33: Recommended bundle for each user depending on their athletic condition and weight goal.

However, in this section bundles with high accuracy do not always translate to bundles with positive ratings. Firstly, there was no more than one bundle in each user type rated below 3. That means that most menus were appreciated by users and they would enjoy following these diet plans. Athletes that want to increase their weight rated, in average, with 4,5 the preference bundles and with 4,0 the nutrient bundles. Due to the fact that these preference bundles have the lowest accuracy, nutrient bundles should be suggested to users that combine both high accuracy and users' approval. Gain weight regular users rated lower in average than the latter however they rated with 3,5 balanced bundles which is their highest average rating. Lose weight athletes and regular users rated, on average, positively both nutrient and preference bundles. While both bundles present high nutrient accuracy in this category, nutrient bundles will almost always provide a more accurate outcome. Athletes that want to maintain weight gave an average rating of 3,75 to preference bundles and of 3 to balanced bundles. Lastly regular users with the same goal rated with 3,58 and 3,42 balanced and nutrient bundles respectively.

6. CONCLUSIONS

In this thesis we focus on building an accurate recommender system that provides users a healthy diet plan. This diet plan could be for one or more days and contain meals and exercises, so users could reach their target weight. This system takes into consideration users' restrictions and preferences in order to provide a personalized recommendation that fits best each user. The idea on [3] provides us a lot of information about the usage of bundles on recommender systems. We introduce a fitness function that takes into consideration users' preferences and we also add some restrictions in order to keep users excited about their diet plan. For example, we avoid recommending the same main meal for two consecutive days so users taste different flavors every day. We conducted a number of experiments in order to better understand our system properties.

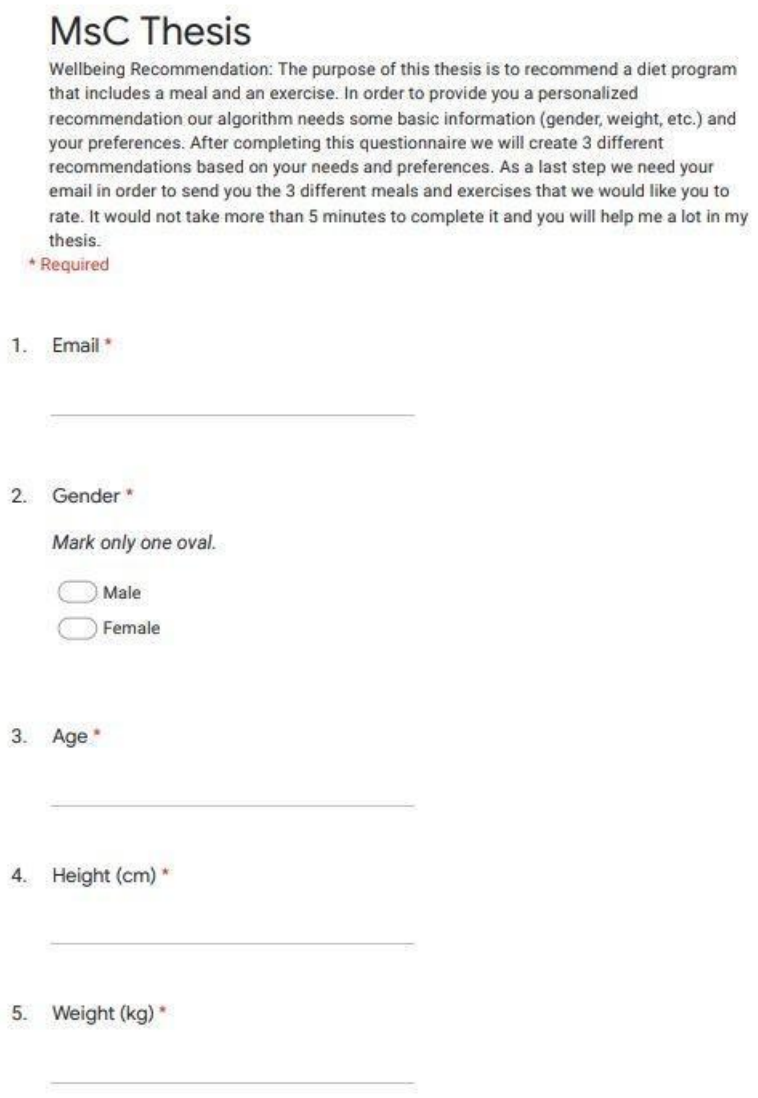
At first, we saw our fitness function lies on an exponential decay through the evolution process. We also noticed that for 10,000 bundles with more than 14,000 different food items we need a bit more than 140 generations in order to provide an (almost) optimal recommendation. We also checked the dependence between different crossover numbers and fitness values. When we repeated the crossover operator more times then the fitness decay rate was getting larger through the evolution process. For larger crossover numbers there was a possibility to create a population with such a low diversity that the algorithm would stick to a local minimum and would not be able to find a solution close to optimal one. However, we saw that even for larger crossover numbers we still find solutions close to optimal and also in fewer generations. In addition, users' preferences do not affect fitness distribution through the evolution process but it just moves the fitness function horizontally (fitness vs distribution diagram). Moreover, we examine a different way to stop the evolution process apart from setting a maximum generation number. We set a maximum fitness difference of the best bundles between two generations and repeat the experiment 30 times. We noticed that when an evolution process stops before reaching 180 generations, its recommendation has a larger fitness value than the best bundle for all iterations. However, its fitness value is close to the average fitness of the best bundles for all iterations which means that while the answer was not close to optimal, it was not also very different from the average best bundle.

We then conducted a survey with 50 volunteers that shared with us some basic information (weight, age, sex etc.), their preferences and weight goal. We recommended to each one of them 3 different personalized diet plans where each one of them focuses on different things. The first one focuses on their preferences, the second one on food nutrients and their target goal and the last one focuses on both preferences and food nutrients equally. All three diet plans were on average at least 90% accurate in terms of food nutrients. So even the plan that focuses on preferences provides a very accurate recommendation that contains the necessary amount of food nutrients that target users have to consume. We used different nutrient distribution for athletes and regular users. We wanted to be sure that our diet plan will not affect the athletic condition of users that participate in sports that need high endurance. We noticed that the athletic condition and weight goal are the most important factors to consider when choosing one out of the three diet plans. Also, most users at first stated that they would prefer a diet plan that focuses on food nutrients or on both food nutrients and preferences. While athletes that want to lose or gain weight gave the highest average rating on the diet plan that focuses on nutrients, the athletes that want to maintain their weight rated higher the diet plan that focuses on their preferences. Almost all users find at least one diet plan that fits their preferences and are happy to follow.

As for future work two better datasets would certainly provide better recommendations. Due to the fact that our food dataset was from Health Canada most of the foods were unfamiliar to Greek users. As a result, we used only 1,000 out of the 14,000 food items that the initial file contained. While these food items provided a wide search space, a dataset with food items closer to the ones we eat in Greece would probably provide better recommendations. We also did not find a dataset that contains specific physical exercises with their intensity and average calories burned. The intensity information and the physical exercises that this dataset contains are very general and not something that we expect to find in a recommender system that is used in a production environment. However, our recommender system still provides accurate diet plans and satisfied almost all of our volunteers. We would love to see in the future a similar algorithm to be established in industrial standard, and provide health related recommendations in order to have more people that follow a healthy lifestyle.

A. Questionnaire

The questionnaire that the 50 volunteers had to answer.



MsC Thesis

Wellbeing Recommendation: The purpose of this thesis is to recommend a diet program that includes a meal and an exercise. In order to provide you a personalized recommendation our algorithm needs some basic information (gender, weight, etc.) and your preferences. After completing this questionnaire we will create 3 different recommendations based on your needs and preferences. As a last step we need your email in order to send you the 3 different meals and exercises that we would like you to rate. It would not take more than 5 minutes to complete it and you will help me a lot in my thesis.

* Required

1. Email *

2. Gender *

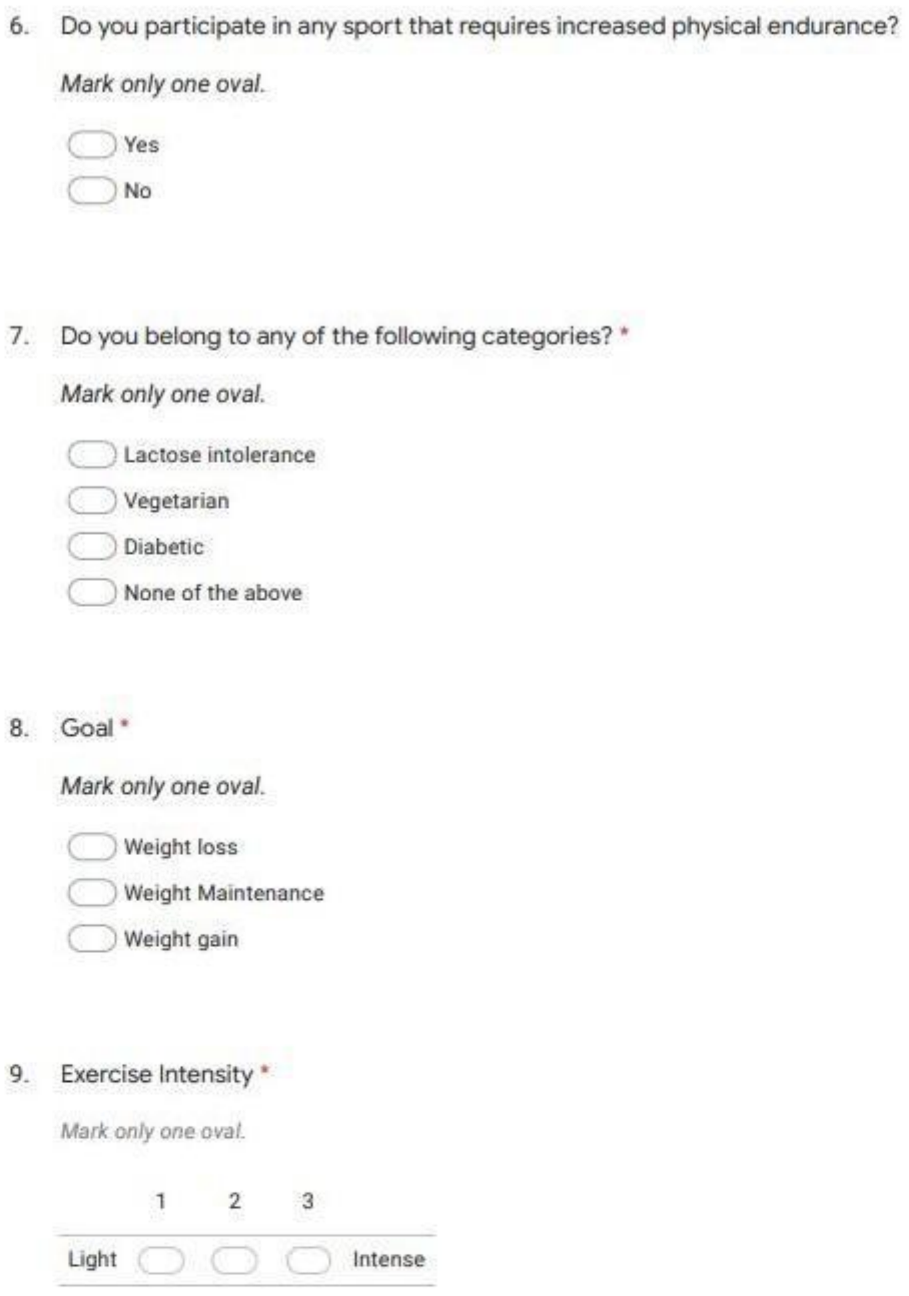
   Mark only one oval.

   ◯ Male
   ◯ Female

3. Age *

4. Height (cm) *

5. Weight (kg) *



6. Do you participate in any sport that requires increased physical endurance?

Mark only one oval.

- ◯ Yes
- ◯ No

7. Do you belong to any of the following categories? *

Mark only one oval.

- ◯ Lactose intolerance
- ◯ Vegetarian
- ◯ Diabetic
- ◯ None of the above

8. Goal *

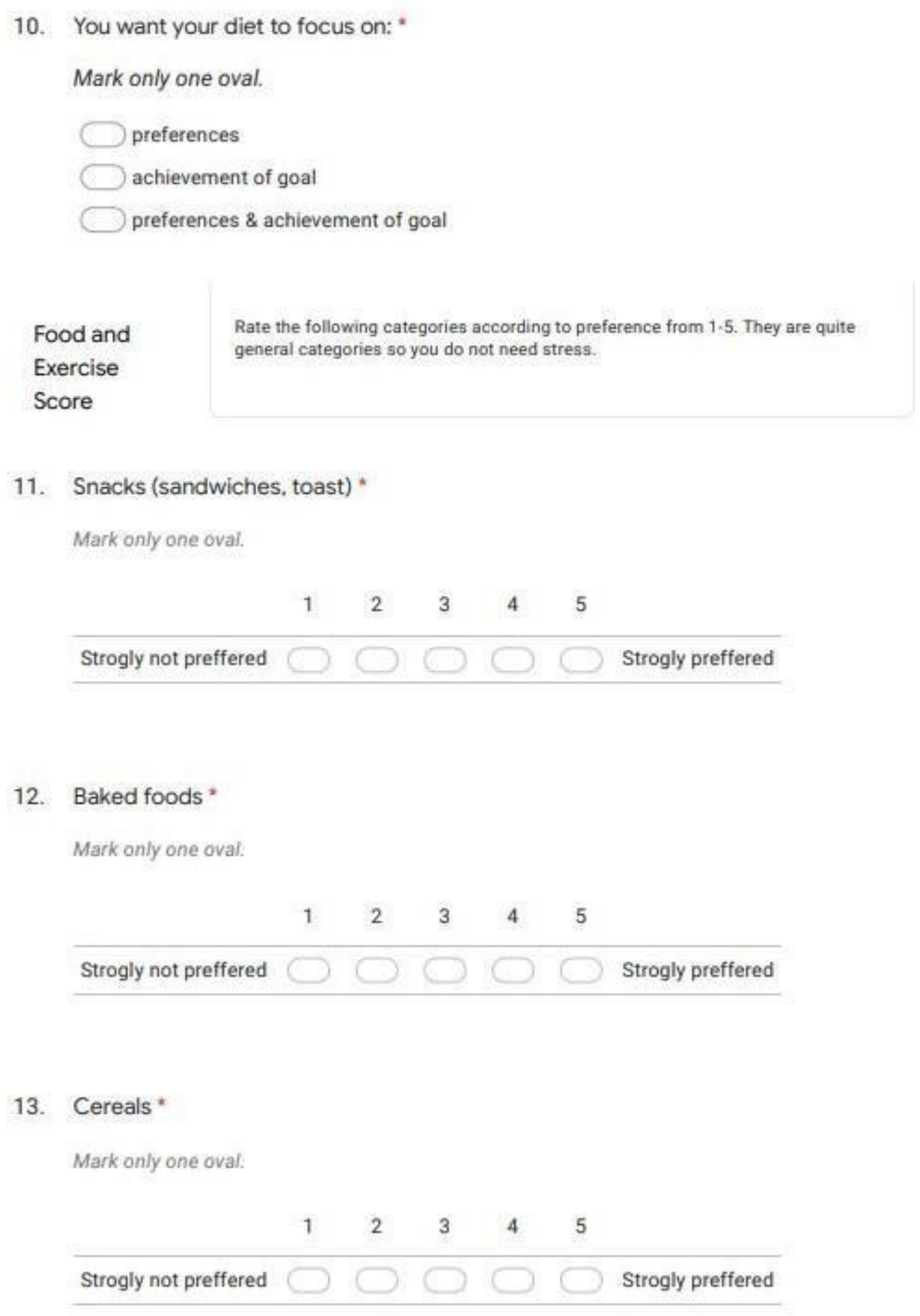
Mark only one oval.

- ◯ Weight loss
- ◯ Weight Maintenance
- ◯ Weight gain

9. Exercise Intensity *

Mark only one oval.

	1	2	3	
Light	◯	◯	◯	Intense



10. You want your diet to focus on: *

Mark only one oval.

- ◯ preferences
- ◯ achievement of goal
- ◯ preferences & achievement of goal

Food and Exercise Score

Rate the following categories according to preference from 1-5. They are quite general categories so you do not need stress.

11. Snacks (sandwiches, toast) *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered

12. Baked foods *

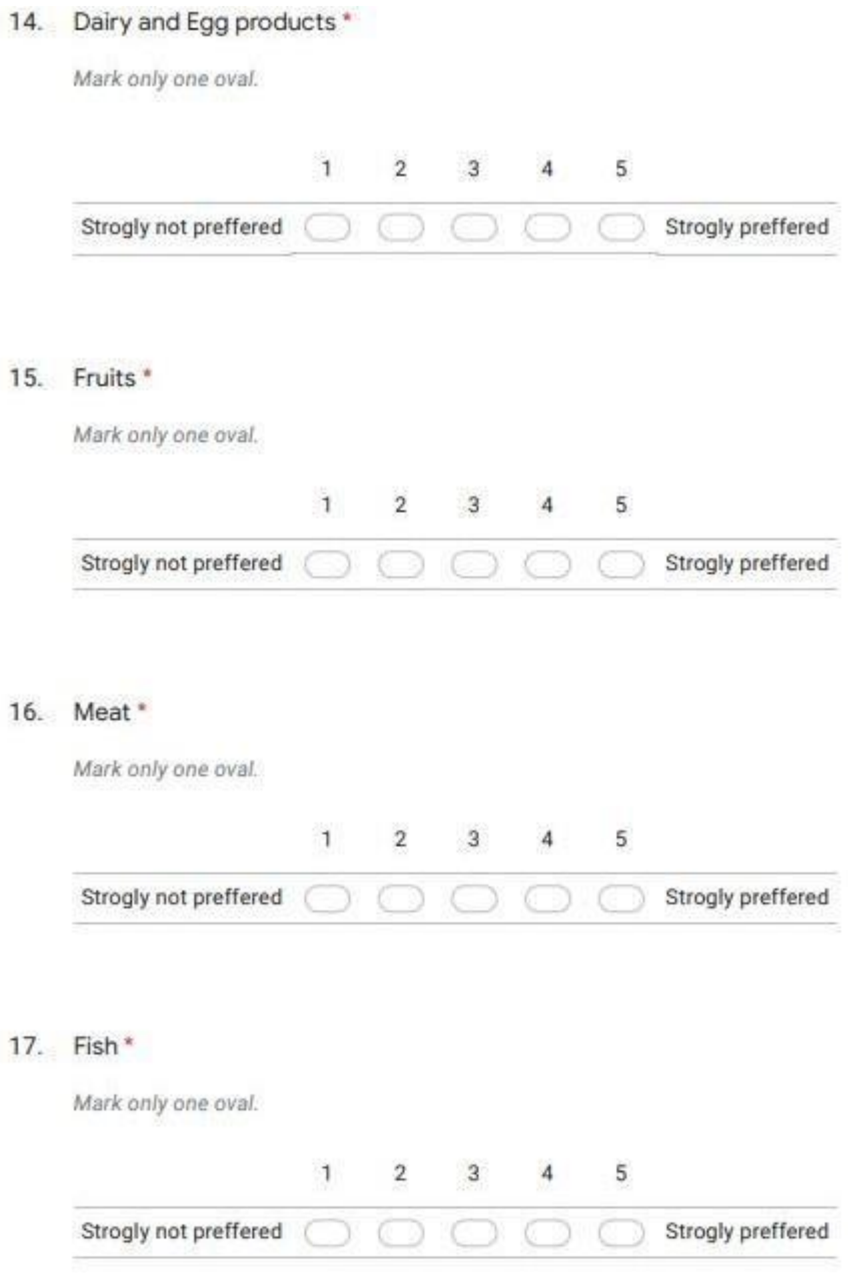
Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered

13. Cereals *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered



14. Dairy and Egg products *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	○	○	○	○	○	Strogly preffered

15. Fruits *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	○	○	○	○	○	Strogly preffered

16. Meat *

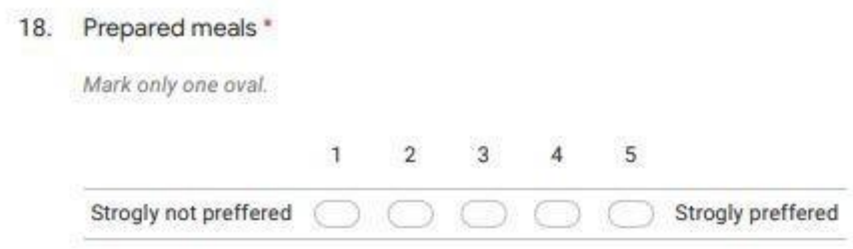
Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	○	○	○	○	○	Strogly preffered

17. Fish *

Mark only one oval.

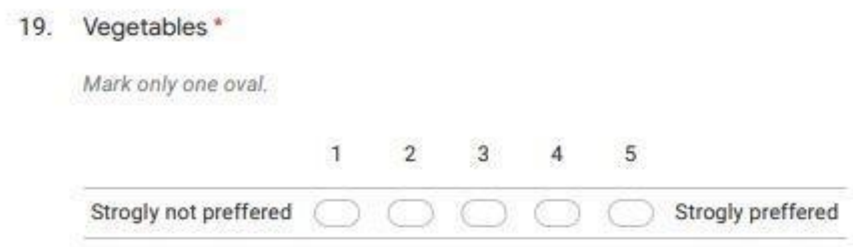
	1	2	3	4	5	
Strogly not preffered	○	○	○	○	○	Strogly preffered



18. Prepared meals *

Mark only one oval.

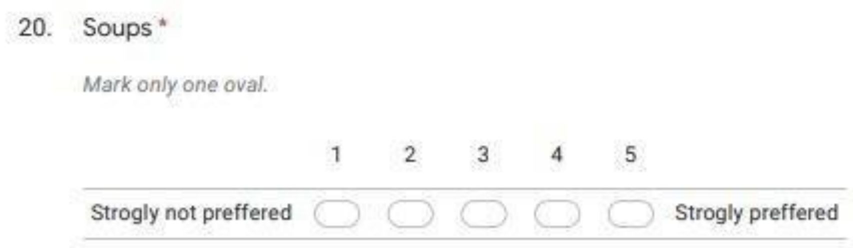
	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered



19. Vegetables *

Mark only one oval.

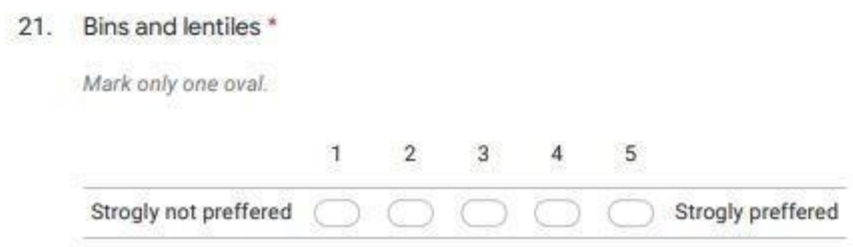
	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered



20. Soups *

Mark only one oval.

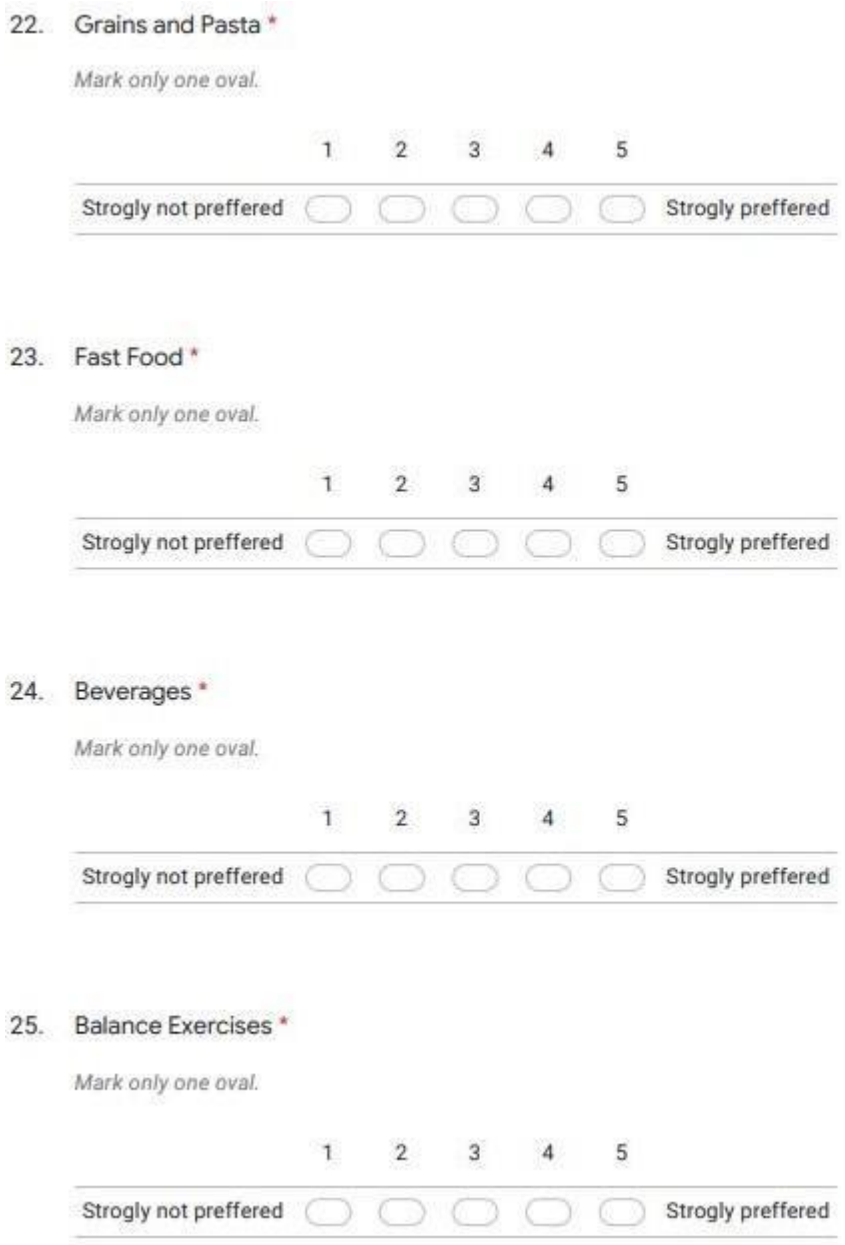
	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered



21. Bins and lentiles *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered



22. Grains and Pasta *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered

23. Fast Food *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered

24. Beverages *

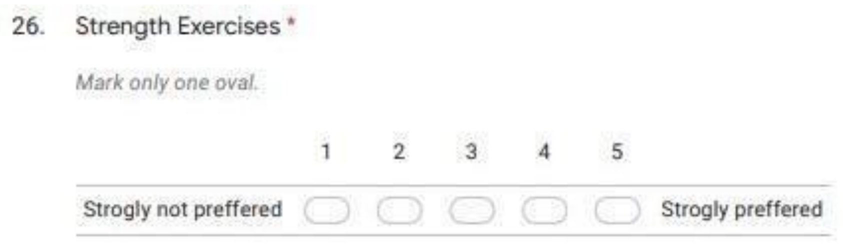
Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered

25. Balance Exercises *

Mark only one oval.

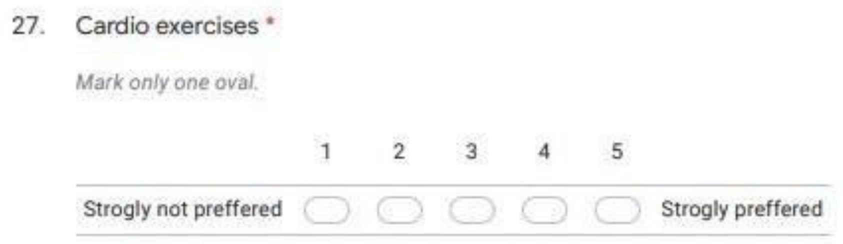
	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered



26. Strength Exercises *

Mark only one oval.

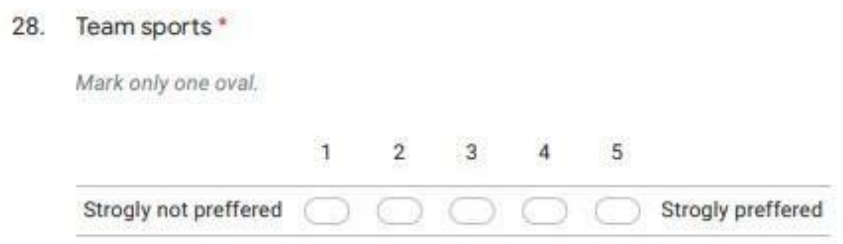
	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered



27. Cardio exercises *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered



28. Team sports *

Mark only one oval.

	1	2	3	4	5	
Strogly not preffered	◯	◯	◯	◯	◯	Strogly preffered

REFERENCES

[1] Tutorials point. https://www.tutorialspoint.com/genetic_algorithms/genetic_ algorithms_fitness_function.htm.

[2] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Com- pany, Incorporated, 1st edition, 2016.

[3] Hugo Alcaraz-Herrera and Iva´n Palomares. Evolutionary approach for ’healthy bundle’ wellbeing recommendations. *HealthRecSys*, 2019.

[4] David A. Coley. *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific Publishing Co., Inc., USA, 1998.

[5] Wikipedia contributors. Genetic algorithms. https://en.wikipedia.org/w/index.php? title=Genetic_algorithm&oldid=1000280478.

[6] Malia Frey. Daily diet composition charts for carbs, protein, and fat. https://www. verywellfit.com/daily-diet-composition-calculator-charts-carbs-protein-fat-3861072, 2020.

[7] Simon Funk. Netflix update: Try this at home. https://sifter.org/~simon/journal/ 20061211.html, 2006.

[8] Adam Hayes. Confidence interval. https://www.investopedia.com.

[9] Wisconsin Department of Health and Familiy Services. Calories burned per hour. https: //www.dhs.wisconsin.gov/library/P-40109.htm, 2005.

[10] The Minister of Health Canada. The canadian nutrient file. https://www.canada. ca/en/health-canada/services/food-nutrition/healthy-eating/nutrient-data/ canadian-nutrient-file-about-us.html, 2015.

[11] Mia A. Papas, Anthony J. Alberg, Reid Ewing, Kathy J. Helzlsouer, Tiffany L. Gary, and Ann C. Klassen. The Built Environment and Obesity. *Epidemiologic Reviews*, 2007.

[12] A.M. Roza and H.M. Shizgal. The harris benedict equation reevaluated. *American Journal of Clinical Nutrition.*, 1984.

[13] Richa Sharma and Rahul Singh. Evolution of recommender systems from ancient times to modern era: A survey. *Indian Journal of Science and Technology*, 9, 2016.

[14] Swearingen Sinha, Kirsten Medhurst, and Rashmi Sinha. Beyond algorithms: An hci perspective on recommender systems. 2001.

[15] Aadhav Vignesh. Calories burned during exercise and activities. https://www.kaggle. com/aadhavvignesh/calories-burned-during-exercise-and-activities, 2020.

[16] Liang Zhang. The definition of novelty in the recommendation system. *Journal of Engineering Science and Technology Review*, 2013.