www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 15(3), Sep-Dec 2024, 126-144. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2024.v15i2.463

Bayesian Expert System for Suggesting Personalized Training Plans with Exercises and Routines with a Model Driven Architecture (MDA)

Rosa Lizeth Estrada Ortega
Guzmán-Mendoza José Eder
Polytechnic University of Aguascalientes
E-mails: mc220010@alumnos.upa.edu.mx
jose.guzman@upa.edu.mx

Abstract. In the dynamic world of fitness, the quest for tailor-made exercise plans has never been more critical. Expert systems are stepping up to this challenge, offering a promise of personalized routines for both seasoned athletes and newcomers. In this article, we introduce an ingenious fuzzy-Bayesian expert system, armed with fuzzy and Bayesian logic, that tackles the uncertainties lurking within fitness data. It's a system with a dual focus: for fitness veterans, it conducts a meticulous analysis of their training history, past achievements, and current ambitions. Guided by the nuanced power of fuzzy and Bayesian logic, it makes decisions rooted in probabilities. As for beginners, the system builds a strong foundation, considering your present fitness status and your future aspirations. This forward-thinking approach speaks directly to the increasing demand for adaptable fitness solutions. For seasoned athletes, it paves the way to peak performance and injury prevention. Meanwhile, beginners embark on their fitness journey with gradual and secure steps. The Fuzzy-Bayesian Expert System offers a comprehensive answer that embraces data diversity and individual fitness aspirations in both scenarios. Furthermore, the seamless integration of the UPAFuzzySystem (Montes Rivera et al., 2023) library and Twilio elevates the system's functionality and its connection with users, turning it into a personal fitness enhancement tool. Dive into a world where fitness meets technology for a custom-made future of health and well-being.
Keywords: Exercise Routines, Healthy Recommendation System, Expert Systems, Fuzzy Logic, Bayesian Logic, Interactive content design, User-centered design methodology

Article Info
Received May 26, 2024
Accepted Sep 11, 2024

1 Introduction

An expert system is a computer system capable of reasoning and acting at the level of an expert in a specific field or activity. Expert systems are designed to address specific problems by applying Artificial Intelligence (AI) techniques that simulate the human reasoning process. These systems earn their name, "expert systems", due to their ability to emulate the decision making performed by experts in a particular field(Refni Wahyuni & Yuda Irawan, 2019), (Unsworth et al., 2021). Approaching the expert system with the training plans, we realize that there is a close relationship, then, some concepts that will be discussed later: Physical exercise is an essential activity for human beings, as it strengthens muscles, improves blood circulation, oxygenates organs and increases skin nutrition. In addition, it contributes to the development of muscle mass, increases endurance and has positive effects such as weight loss, mood improvement and increased body energy(Refni Wahyuni & Yuda Irawan, 2019). With the growth of the fitness community, there have been technological advances that facilitate activities related to nutrition and exercise. However, there is still an opportunity to improve care for these needs (Unsworth et al., 2021)The following is a summary of the findings.

Exercising, participating in sports activities and maintaining constant physical activity and even vibration of hole body routines offer a valuable opportunity to preserve our health and improve both our physical condition and mental well-being. There is strong evidence to support the positive impact of physical activity on our health, helping to prevent and mitigate various

diseases and health problems. Regardless of our age, we all have the ability to integrate this healthy habit into our daily routine, whether we are just starting or resuming physical activity after a period of inactivity (Unsworth et al., 2021), (Praja & Yudha, 2021).

Among modern technologies, the Internet of Things (IoT) stands out, which enables the connection to the network and the incorporation of computing capabilities to virtually any object of daily use. This makes it possible to record all interactions with these objects and generate a valuable historical database (Sharples, M., Adams, M., Graham, C., McMahon, T., Baxter, G., & Wood, D., s. f.)., within the development of the system we will take up some topics such as fuzzy logic and Bayesian logic and how they were applied in the process until obtaining a result.

Fuzzy Logic offers an inference mechanism that mimics human reasoning in knowledge-based systems. Its theory provides a mathematical framework for modeling uncertainty in human cognitive processes (Raimondi et al., 2021).

Bayesian inference, founded on Bayes' theorem, is a statistical inference strategy with properties that make it especially relevant in scientific research (Grunau & Linn, 2018), (Perbawawati et al., 2019) .

We decided to create the system of training plans with the objective of providing training focused on beginners and athletes taking into consideration that many of those who start or not in the gym tend to perform the exercises incorrectly or without taking into consideration very important aspects such as rest, health aspects, objectives and levels.

Multivariate analysis plays a crucial role in our research by allowing us to simultaneously examine multiple variables and their interactions. As we explore the data collected and evaluate the results of our study, it is essential to consider the joint influence of several variables on the phenomena studied. This technique gives us a deeper understanding of the complex relationships that can exist be-tween different factors and allows us to identify patterns, trends and effects that might otherwise go unnoticed. By applying multivariate analysis in our work, we are equipped to address complex questions and answer our research objectives with greater precision and robustness. This translates into greater reliability and validity in our conclusions, which strengthens the robustness of our research and the quality of the recommendations and training plans we provide in our expert system.

Experimental design is going to play a fundamental role in this research, as it focuses on developing an expert system that suggests personalized training plans. Determining the appropriate design will allow the collection of accurate and representative data to support fitness decision making. To achieve this, careful con-sideration must be given to variables such as the athletes' level of experience, their individual goals, proposed exercise routines, and other factors that influence the effectiveness of training plans. The choice of a sound experimental de-sign will allow meaningful analysis of how decisions based on fuzzy and Bayesian logic influence athletes' progress, providing valuable information for continuous improvement of the expert system.

The challenge lies in a comprehensive model to guide user development, which involves the intersection of different disciplines, such as Model Driven Architecture (MDA), user interface design, specific workflows and the use of assistive technologies, such as SPEM, HAMSTERS and FIGMA.

The problem is complex because it involves adapting to different exercise levels, health goals and preferences. In addition, careful planning and effective collaboration between designers, developers and educators is required to ensure compliance with technology standards.

1.2 Formulas

Forward chaining and backward chaining inference techniques
Forward chaining.

Is an inference method used in expert systems to make decisions or reach conclusions based on a set of initial rules and facts, this approach starts with known facts or input data and combines them with inference rules to generate new conclusions or actions.(Perbawawati et al., 2019)

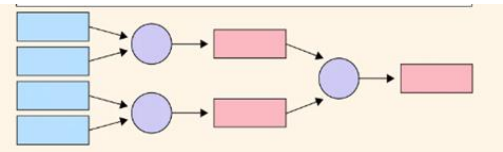


Fig. 1 Forward Chaining

Backward chaining

A method of inference used in expert systems to reach a conclusion or decide by working backward from the desired goal to the available facts or data. Instead of starting with the initial facts and applying rules to reach a conclusion, back-ward chaining starts from a goal and looks for rules and facts that support or satisfy that goal (Cárdenas López et al., 2016).

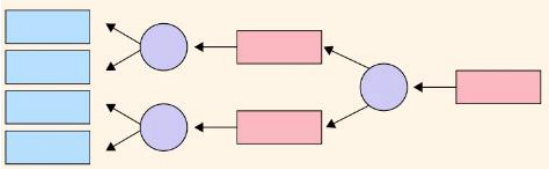


Fig. 2 Backward chaining

Bayes theorem.

Bayes' theorem is used to calculate the probability of an event, having information beforehand about that event. We can calculate the probability of an event A, knowing also that A complies with a certain characteristic that conditions its probability. Bayes' theorem understands probability inversely to the total probability theorem. The total probability theorem makes inference about an event B, from the results of the events A. On the other hand, Bayes calculates the probability of A conditioned to B, as detailed in equation (1) (Penalva Martínez & Posadas García, 2009).

$$P(A \mid B) = \frac{P(B|A)P(A)}{P(B)} \quad (1)$$

Where P(A|B) is the conditional probability of A given B. P(B|A) is the conditional probability of B given A, P(A) and P(B) are the prior probabilities of A and B, respectively.

Fuzzy logic.

The use of fuzzy logic in expert systems has proven to be especially effective in areas where uncertainty and subjectivity are common, such as medical diagnosis, engineering decision making, and route planning in navigation systems. These systems have enabled greater accuracy and adaptability in a variety of practical applications (Botto-Tobar et al., 2020).

Tkinter

Tkinter is a Python library designed for creating and developing desktop ap-plications with a graphical user interface. This library simplifies the creation of graphical user interfaces in Python and is used to interface with the Tcl/Tk (Tk) toolkit. It is the standard Python package that facilitates the construction of graphical user interfaces in desktop applications (D B Beniz & A M Espindola, 2016).

Twilio

Twilio provides an easy entry point into the world of telephony (D B Beniz & A M Espindola, 2016), it is a development platform that enables programmers to create cloud communication applications and web systems. Its application programming interfaces (APIs) make it possible to deliver personalized communication experiences for users through web and mobile Applications (Romeiser et al., 2021).Twilio's APIs are also available in the cloud (Figure 3 Dashboard Twilio).

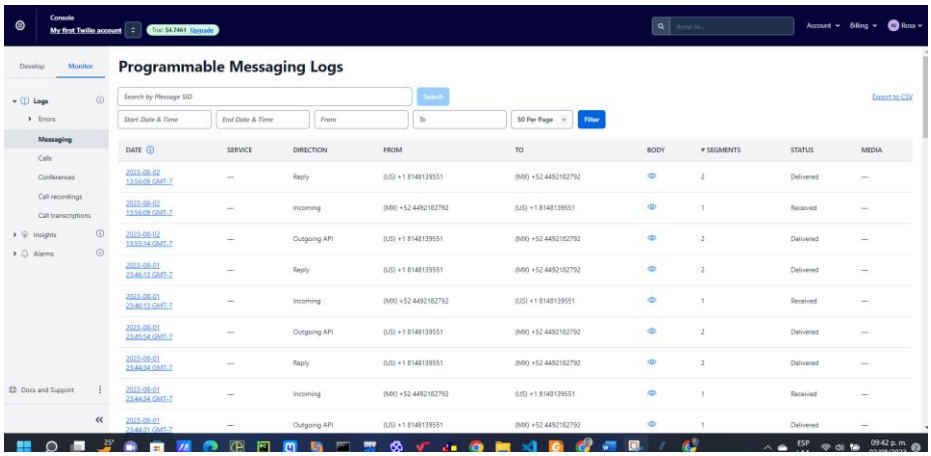


Fig. 3 Dashboard Twilio

Analysis of the User Task

MDA approach in today's digital and technological world, user experience plays a critical role in the effectiveness and success of applications and systems. User task analysis is an essential practice used to understand and optimize how users interact with an interface, application or online process.

User task analysis involves the careful study of how individuals carry out specific activities in a digital environment. This ranges from using a mobile and/or web application or interacting with a complex system. By breaking down and examining the tasks users must perform, designers and developers can identify, improvements and opportunities to create more intuitive and efficient experiences (Bystrenina & Nikitin, 2022).

In addition, when building a software application is to make it easier for end users to achieve their goals and carry out their tasks with the systems in an effective and efficient way. Therefore, in this process it becomes necessary, not only to take into account the system requirements, which would lead us to obtain a good system, robust and complete from the computer science point of view (Martínez et al., 2014) , also in several case studies have a poor level of usability, since they pay more attention to elements related to the interior of the system as its performance or reliability how to perform an interactive design focused on maintaining the attention of the receiver, which adapts to the specific characteristics of each user.

MDA approach

The Model Driven Architecture (MDA) (Pascuas Rengifo et al., 2015) approach places models and their transformations at the center of the software development process. The MDA approach is supported by the Object Management Group (OMG), which is a consortium made up of several well-known companies, such as IBM, Borland, Hewlett-Packard and Boeing, among others. The main purpose of this consortium is to create and maintain a set of specifications that facilitate interoperability between different software applications. This implies that as we move forward in the development process, more and more detailed transformations of these models are required. Model transformations are considered fundamental assets that must be treated rigorously in terms of software engineering. This includes analysis, design, implementation, testing, maintenance, and configuration management of these transformations (Colque C & Valdivia P, 2006). This includes the analysis, design, implementation, testing, maintenance, and configuration management of these transformations.

2 Experimental procedures

In addition, the key steps that make up the implementation process will be dis-cussed, providing a complete overview of the system. Facts are the basic unit of information of experta. They are used by the system to reason about the problem. We start by adding the required classes to define the facts that the expert system requires to identify rules and conclusions in the pro-posed expert system. Those classes include level of exercise, pressure, training, future health, and age, impression, satisfaction, quality, time, target, routine the Code 1 shows the definition of the classes in Python using experta library.

Code 1 Rule-based expert system

```
class Level(Fact):

    pass
class Pressure(Fact):

    pass
class Training (Fact):

    pass
class HealthFuture (Fact):

    pass
class Age(Fact):

    pass
class Printed Fact):

    pass
class Satisfaction (Fact):

    pass
class Quality (Fact):

    pass
class Time (Fact):

    pass
class Objective (Fact):

    pass
class Routine (Fact):

    pass
```

Most of the time expert systems needs a set of facts to be present for the system to work. This is the purpose of the DefFacts decorator. You can subclass Fact to express different kinds of data or extend it with your custom functionality. Those sub-facts could help to work with the functions de-fined in each rule of the expert system. In Code 2 there is an example defining the sub-class facts required when the expert system is initialized.

Code 2 Definition of sub-facts for the initialization of the expert system.

```
    @DefFacts()
    def initialize_facts (self):
        yield Level ()
        yield Pression()
        yield Quality ()
        yield Time()
        yield Satisfaction ()
```

A more efficient and effective approach to inferring causes and taking actions based on accumulated knowledge, backward chaining is a crucial technique for decision making, problem solving and development of intelligent systems. Within question_objective1, the ask info function is used to ask the user about his or her goal ("What is your goal (to lose weight/gain muscle mass/fitness)?"). Here, the goal "lose weight" is set as a possibility. The following Code 3 shows one of the forward chaining rules in the Exercise class:

Code 3 The rules that are of type backward chaining

```
@Rule(Training(training ="yes"))
    def question_objective1 (self):
        self.declare(Objetive(objetive=ask_info("¿What is your goal
```

```
(to lose weight/gain muscle mass/fitness)?)")))
    @Rule(Objetive(objetive=" lose weight "), NOT(Level(level=W())))
    def level_question11(self):
        self.declare(Level(level=ask_info("What is your experience
level? (1/2/3)")))
```

Backward chaining is used to make knowledge-based decisions. Its capability is to work backward from a target. First, rules are established containing conditions (premises) and actions (con-sequences). The rules are activated if the condition of having a "lose weight" goal and a specific level ("1" or "2") is met. When one of these conditions is met, the action corresponding to the rule is executed. For example, if the objective is "lose weight" and the level is "1", the rule recommendation_route11 is executed.

Code 4 The rules that are of type forward chaining

```
@Rule(Objetive(objetive=" lose weight "), Level(level="1"))
    def recommendation_route11 (self):
        show_info("We recommend low intensity cardio and strength
training routines.")
        self.declare(Rutine(rutine="1"))
    @Rule(Objetive(objetive=" lose weight "), Level(level="2"))
    def recommendation_routine21 (self):
        show_info("We recommend doing cardio and strength training
routines with moderate intensity.")
        self.declare(Rutina(rutina="2"))
```

A rule based on Bayes' theorem is mentioned that involves a person's age and the probability of experiencing difficulties in his body; {LS = 1.17, LN= 0.37}: These numerical values are the probability factors associated with the above condition. In this case, two values are provided: LS and LN. the rule is shown below in Code 5. Examples and possible results: "What is your goal (lose weight /gain muscle mass /fitness)?"

Objective	Nivel	Result
lose weight	"1"	We recommend low intensity cardio and strength training routines.
lose weight	"2"	We recommend doing cardio routines and strength exercises with moderate intensity.
lose weight	"3"	We recommend functional training routines and high intensity exercises.
gain muscle mass	"1"	We recommend strength routines with moderate weight and high number of repetitions.
gain muscle mass	"2"	We recommend strength routines with high load and low repetitions.
gain muscle mass	"3"	We recommend strength routines with high load and low repetitions.

fitness	"1"	We recommend basic exercise routines to improve your physical condition.
lose weight	"2"	We recommend doing combined exercise routines to improve your fitness.
lose weight	"3"	We recommend advanced exercise routines to maintain a high level of physical fitness.

Code 5 Bayesian rules IF Age is Old {LS = 1.17 LN= 0.37} THEN difficulties in their body {0.38}

```
@Rule(Age(state="old") | Age(state="young"))
    def EdadFit(self):
        LS=1.17
        LN=0.37
        OS=0.38
        OS=OS/(1-OS)
        if self.today_edad['state'] == 'old':
            OS_T = OS*LS
            OS = OS_T/(1+OS_T)
        if self.today_edad['state'] == 'young':
            OS_N = OS*LN
            OS = OS_N/(1+OS_N)
        self.future = SaludFuturo(OSr=OS)
        self.declare(self.future)
        print("Old, probability problems in body %f"
%self.future['OSr'])
```

This rule is based on the condition of a person's age and uses probability factors based on Bayes' theorem to calculate the probability of needing training. If the age is classified as "young", the posterior probability of needing training is 0.61, according to the probability factors provided. Show Code 6.

Code 6 IF Age is young {LS = 2.67 LN= 0.85} THEN training {0.61}

```
@Rule(Age(state="young") | Age(state="old"))
    def EdadFit1(self):
        LS=2.67
        LN=0.85
        OS=0.61
        OS=OS/(1-OS)
        if self.today_edad['state'] == 'young':
            OS_T = OS*LS
            OS = OS_T/(1+OS_T)
        if self.today_edad['state'] == 'old':
```

```
            OS_N = OS*LN
            OS = OS_N/(1+OS_N)
        self.future = SaludFuturo(OSr=OS)
        self.declare(self.future)
        print("Young, probability able to training %f"
%self.future['OSr'])
```

Rule-based expert system with fuzzy rules.

The proposed expert system has 2 rules, which use at least two fuzzy universes in the input to calculate the acceptance of each of the rules. For knowledge generation, rules supported on a dataset were used Rules.

1. Classify user satisfaction on the basis of two variables: quality and time.

Fuzzy sets

Quality_Universe: Represents the fuzzy universe for training quality, with the fuzzy sets "bad", "fair" and "good".
Universe_Time: Represents the fuzzy universe for training outcome time, with the fuzzy sets "low", "medium" and "high".
Satisfaction_Universe: Represents the fuzzy universe for customer satisfaction status, with the fuzzy sets "bad", "fair" and "good".
quality_universe = np.arange(1, 101, 1)
tiempoRes_universe = np.arange(1, 7, 1) [Days]
satisfaction_universe = np.arange(0,100.1,0.1)

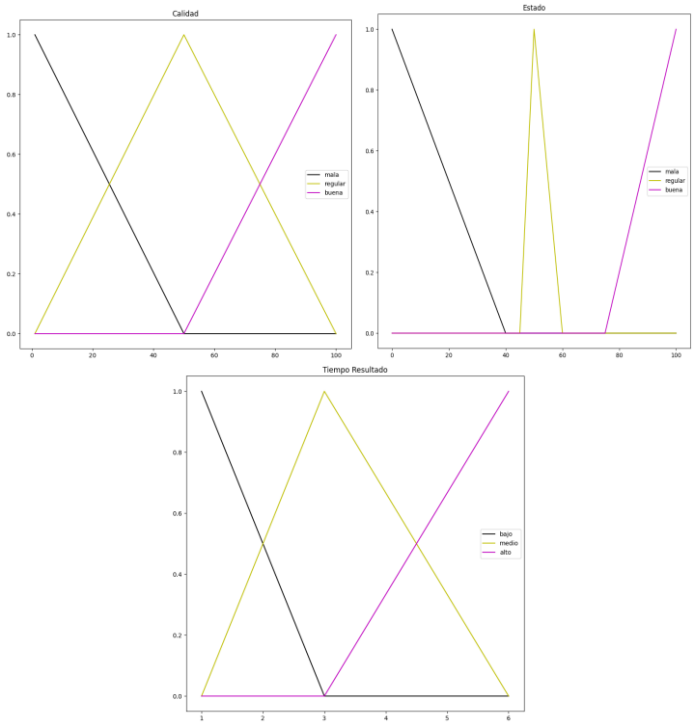


Fig. 4 Universe's quality, time, satisfaction

The following Code 7defines a rule-based inference system using the UPAFuzzySystems library. The inference system is called "Result Quality_Time Inference" and is used to evaluate satisfaction as a function of result quality and time.

Assumptions are added to the inference system, which include three universes: Quality_Universe, TimeRes_Universe (Result Time) and Satisfaction_Universe (Satisfaction).

Rules are added to the inference system that establish relationships between quality, time-result and satisfaction status. For example, if the quality is "bad" and the result time is "low", the satisfaction state is "bad".

Multiple rules are defined covering different combinations of quality and out-come time, and how they affect the satisfaction state. The inference system is set up using the "Mamdani" method.

Code 7 Definition of the Rule 1

```
Satisfaction_Inference   =UPAfs.inference_system('Result   Quality_Time
Inference')
Satisfaction_Inference.add_premise(Quality_Universe)
Satisfaction_Inference.add_premise(TimeRes_Universe)
Satisfaction_Inference.add_premise(Satisfaction_Universe)
Satisfaction_Inference.add_rule([['Quality','bad'],['Time
Result','low']],['and'],[['State','bad']])
Satisfaction_Inference.add_rule([['Quality','bad'],['Time
Result','medium']],['and'],[['State','regular']])
Satisfaction_Inference.add_rule([['Quality','bad'],['Time
Result','high']],['and'],[['State','good']])
Satisfaction_Inference.add_rule([['Quality','regular'],['Time
Result','low']],['and'],[['State','bad']])
Satisfaction_Inference.add_rule([['Quality','regular'],['Time
Result','medium']],['and'],[['State','regular']])
Satisfaction_Inference.add_rule([['Quality','regular'],['Time
Result','high']],['and'],[['State','regular']])
Satisfaction_Inference.add_rule([['Quality','good'],['Time
Result','low']],['and'],[['State','good']])
Satisfaction_Inference.add_rule([['Quality','good'],['Time
Result','medium']],['and'],[['State','good']])
Satisfaction_Inference.add_rule([['Quality','good'],['Time
Result','high']],['and'],[['State','good']])
Satisfaction_Inference.configure('Mamdani')
Satisfaction_Inference.build()
```

2. Calculate blood pressure status and level of training

Fuzzy sets

Universe_Level: Represents the fuzzy universe for the training level, containing three fuzzy sets: "low", "medium" and "high".

Arterial_pressure_Universe: Represents the fuzzy universe for blood pressure, containing three fuzzy sets: "Low", "Normal" and "High".

nivel_universe = np.arange(1,4,1)

presion_arterial_universe = np.arange(60, 161,0 1)

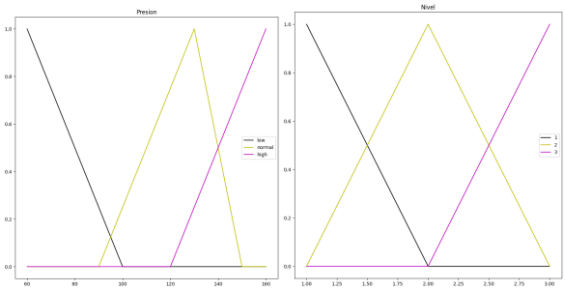


Fig. 5 . Level and Pression Universes

We add premises to the inference system, **Error! Reference source not found.** which include two universes: Level_Universe and Pressure_Universe.
A consequence is established in the inference system, which is the Estate_Universe.

Rules are added to the inference system that establish relationships between level and pressure, and how they affect the state. For example, if the level is "1" and the pressure is "low", then the state is "Low".

Multiple rules are defined that cover different combinations of level and pressure, and how they affect the state.

Code 8 Definition of the Rule 2

```
Estate_Inference =UPAfs.inference_system('Results pression_level Inference')
Estate_Inference.add_premise(Level_Universe)
Estate_Inference.add_premise(Pression_Universe)
Estate_Inference.add_consequence(Estate_Universe)
Estate_Inference.add_rule([['Level','1'],['Presion','low']],['and'],[['Estate','Low']])
Estate_Inference.add_rule([['Level','1'],['Presion','normal']],['and'],[['Estate','Low']])
Estate_Inference.add_rule([['Level','1'],['Presion','high']],['and'],[['Estate','Low']])
Estate_Inference.add_rule([['Level','2'],['Presion','low']],['and'],[['Estate','Low']])
Estate_Inference.add_rule([['Level','2'],['Presion','normal']],['and'],[['Estate','Middle']])
Estate_Inference.add_rule([['Level','2'],['Presion','high']],['and'],[['Estate','High']])
Estate_Inference.add_rule([['Level','3'],['Presion','low']],['and'],[['Estate','High']])
Estate_Inference.add_rule([['Level','3'],['Presion','normal']],['and'],[['Estate','High']])
Estate_Inference.add_rule([['Level','3'],['Presion','high']],['and'],[['Estate','High']])
Estate_Inference.configure('Mamdani')
Estate_Inference.build()
```

Python Expert Systems.

Two modules related to the Python graphical interface are imported using the tkinter library:
From tkinter import *: This line imports all the elements and functions available in the tkinter module. This includes classes and functions to create windows, botons, labels and other GUI elements.

From tkinter import ttk: In addition to tkinter, this line imports the ttk submod-ule of tkinter, which provides widgets. Code 9

Code 9 Tkinter in python

```
from tkinter import *
from tkinter import ttk
```

These modules allow you to create graphical user interfaces (GUI) in a simple way Code 10,

sendInfo(): This function is used to collect the information entered by the user through a text input window and then print that information to the console. It then destroys the main window.

showInfo(): This function is used to close the main window without collecting additional in-formation.
ask_info(question): This function creates a new dialog box to ask the user a question. The question is passed as an argument in the variable question.

Code 10 Interfaces

```
def sendInfo():
    global information, root, info
    info = information.get()
    print(info)
    root.destroy()
def showInfo():
    global root
    root.destroy()
def ask_info(question):
    global information,info,root
    root =Tk()
    root.title("Ask information")
    root.geometry("700x300+300+300")
    Ask_information = StringVar()
    Ask_information.set(question)
    Ask_information_label = ttk.Label(root,
textvariable=Ask_information)
    Ask_information_label.pack(pady=5)
    information = StringVar()
    information_entry = ttk.Entry(root, width=30,
textvariable=information)
    information_entry.pack(pady=5)
    Send_information_button =
ttk.Button(root,text="Accept",command=sendInfo)
    Send_information_button.pack()
    root.mainloop()
    return info
```

These functions are designed to obtain information entered by the user, store it in a variable called info, print that information on the console and then close the main window where the application or graphical interface is located.

The GUI function developed in Python using the tkinter library, Code 11. The function is called show_info(info) and is used to display information in a dialog box.

This function creates a pop-up window to request information from the user, obtains the information entered by the user and returns it as a result.

Code 11 Windows

```
def show_info(info):
    global information,root
    root = Tk()
    root.title("Ask information")
    root.geometry("700x300+300+300")
    show_information = StringVar()
    show_information.set(info)
    show_information_label =
ttk.Label(root,textvariable=show_information)
    show_information_label.pack(pady=5)
    Send_information_button = ttk.Button(root, text="Accept",
command=showInfo)
    Send_information_button.pack()
    root.mainloop()
```

Twilio

This Code 12imports and uses the twilio, flask, and twilio.twiml libraries to integrate the Twilio platform into a Python application from twilio.rest import Client: Imports the Client class from the twilio.rest library, which allows communication with the Twilio API.

From flask import Flask, request, jsonify: Imports the Flask, request, and jsoni-fy classes from the flask library, which are used to create a web application and handle HTTP requests.

Fromtwilio.twiml.messaging_response import MessagingResponse:Importsthe MessagingResponse class from the twilio.twiml library, which is used to generate responses to incoming SMS messages.

account_sid and auth_token: These variables store the authentication credentials of your Twilio account.

client = Client (account_sid, auth_token): Creates a client object using the provided credentials. This object is used to interact with the Twilio API and send SMS messages.

Code 12 Twilio SMS

```
from twilio.rest import Client
from flask import Flask, request, jsonify
from twilio.twiml.messaging_response import MessagingResponse
import time
account_sid = 'AC1ea0fd99cb15e7a138a1035e14dd4904'
auth_token = 'df9e074b1e9d0dcd86f7e15919aa93a0'
client = Client(account_sid, auth_token)
```

Sends an SMS message with the content provided in the parameter ask from the phone number +18148139551 to the phone number +524492182792, and then prints the SID of the sent message. Code 13

message = client.messages.create(...): This line uses the client object (which was previously initialized with your Twilio credentials) to create a new SMS message. It specifies the sender's number (from_), the message content (body), and the recipient's number (to).

print(message.sid): Once the message is sent, the identifier (SID) of the message is printed.

Code 13 Parameters

```
def send_sms(ask):
    print('test)
    message = client.messages.create(
        from_= '+18148139551',
        body=ask,
        to=  '+524492182792'
    )
    print(message.sid)
```

The function returns the contents of the last message received as a string. If no message has been received, the function will return None.

The while loop will run as long as test is equal to 0, which means that it will keep looking for messages until one is received. Code 14

If there are no messages in the list, this code block will be executed.

Previous variables are used in a while loop to control the reception of messages.

while test == 0::: This loop will be executed as long as test is equal to 0, that is, until a message is received.
messages_list = list(messages): Converts the list of SMS messages to a Python list for easy manipulation.
if messages_list::: Checks if there are messages in the list.
last_message = messages_list[0]: If there are messages in the list, gets the last message, which is the first item in the list.
respose = last_message.body: Gets the content of the message and stores it in the response variable.

Code 14 Content

```
def recive_sms():
    print('prueba salida')
    time.sleep(5)
    messages = client.messages.list(from_= '+524492182792' , limit=5)
    test = 0
    while test == 0 :
        messages_list = list(messages)
        if messages_list:
            ultimo_mensaje = messages_list[0
            respose = ultimo_mensaje.body
            test = 1
        #return str(respose)
        else:
            #respose = "No se encontraron mensajes."
            test = 0
            time.sleep(5)
            messages = client.messages.list(from_= '+524492182792' ,
limit=5)
            #print=(respose)
    return str(respose)
```

3 Results

Possible answers that the user can enter. Example when a user enters this information: Information is being collected on whether the user wishes to train Fig. 6.

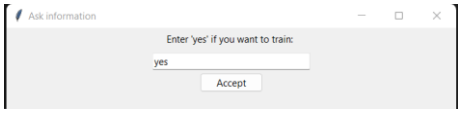


Fig. 6 Ask the user if he/she is ready to train.

The user's goal (e.g., to lose weight, gain muscle mass or improve fitness) is being collected and incorporated as part of the expert system's knowledge for further processing and recommendations. Fig. 7

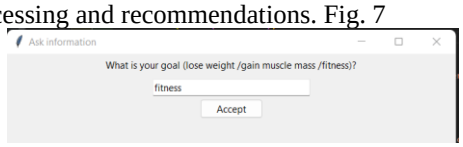


Fig. 7 User's objective: Lose weight, Gain muscle mass Fitness

Determine a user's level of expertise based on his or her training objective Fig. 8

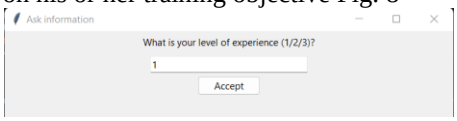


Fig. 8 Experience level 1 beginner,2 Intermediate 3 Expert

Differentiating between beginners, intermediates and experts allows training programs to be tailored effectively. Each group has specific needs and goals, and this classification serves as a starting point for designing customized fitness plans.

By categorizing individuals into these three levels, it is easier to measure and communicate progress. Athletes can clearly see their progress as they move from one level to the next, which can be a source of motivation.

Beginners often have different needs than experts in terms of exercise, intensity and volume. This classification helps ensure that newcomers are not faced with routines that are too intense and potentially dangerous.

This rule Fig. 9provides a specific exercise routine recommendation for a user who has the goal of improving their fitness and is a beginner (level 1). The recommendation is to follow basic exercise routines to achieve that goal.

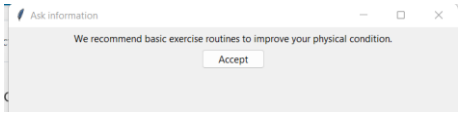


Fig. 9 Results for user

This rule is used to ask the user about the weekly training time in days (in a range from 1 to 6) if the system has no previous information about this data. Fig. 10

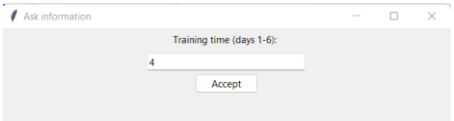


Fig. 10 Real time for training

To ask the user about the quality of the training on a scale from 1 to 100.Fig. 11 The user's answer is converted into a decimal value and stored in an instance of the "Quality" entity. This allows the expert system to gain knowledge about the quality of the training as perceived by the user.

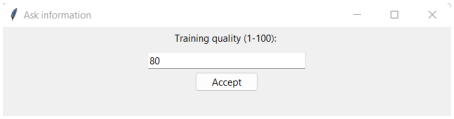


Fig. 11 Percent for training quality

Final customer satisfaction is determined as the minimum value between the quality and time memberships. This means that satisfaction will be limited by whichever aspect has the lowest membership. Fig. 12

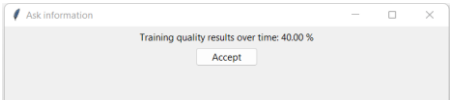


Fig. 12 Result for user

It is used to collect information about the user's age if the system has no previous information about it. The user's response is stored for further processing in the expert system. Fig. 13

Fig. 13 Range for user old or young

Assess the likelihood of problems in the body of a young person with high or low blood pressure. A message is displayed to the user informing about the probability of problems in the body of a young person with high blood pressure. The probability value is obtained from the "FutureHealth" instance. **Error! Reference source not found.**

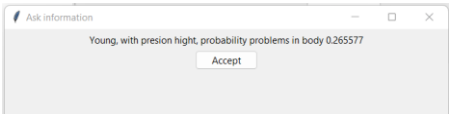


Fig. 14 Results for user

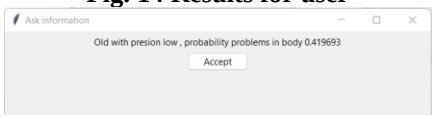


Fig. 15 Probability problem pressure

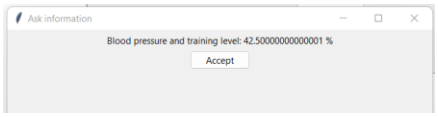


Fig. 16 Results of the blood pressure

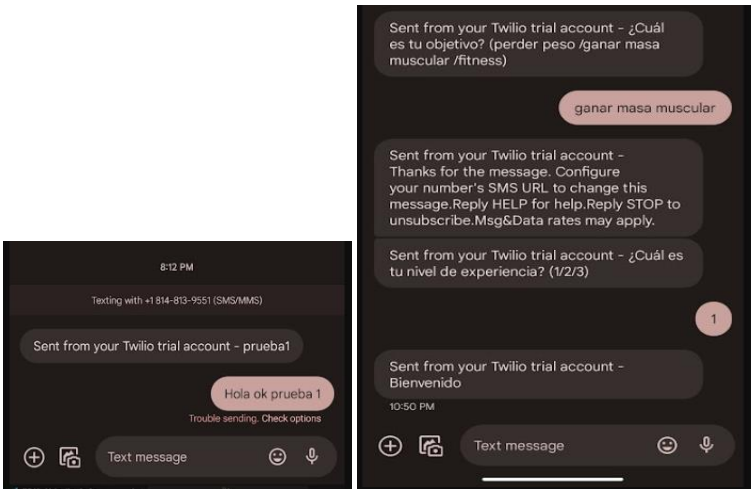
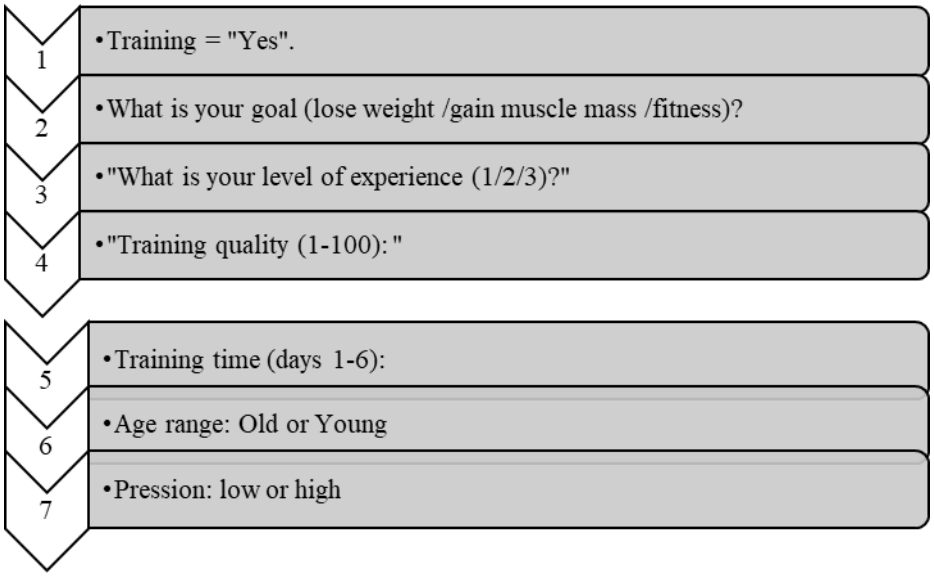


Fig. 17 Results with Twilio SMS

Note: The situation in which the system fails to recognize the text entered by the user becomes a critical point that can lead to the deterioration of the user experience and negatively affect the effectiveness of the system as a whole. When the system is unable to correctly interpret user requests, questions or commands, a number of unintended consequences are generated. This can include irrelevant or incorrect responses, redundant requests for clarification, and even the inability to complete tasks or provide crucial information. This lack of proper acknowledgment can also result in wasted user time and increased frustration, ultimately destroying the reliability and usefulness of the system. Therefore, effectively addressing the improvement of text recognition capability is essential to ensure a smooth and successful interaction between the user and the artificial intelligence application or system.

The conceptual diagram, presented below, is a visual map that identifies the key components of our research, the interactions between them and how they contribute to the achievement of our objectives.



As we move forward in exploring our research, the conceptual diagram will help us illustrate the relationship between variables, processes, influencing factors, and expected results. It will also serve as a guide for readers, allowing them to immediately understand the general structure of our study before delving into the details.

4 Conclusions

In this work we defining an expert system that uses fuzzy and Bayesian rules to help expert athletes or beginners to find the right routines for their level of goal they are looking for. The proposed system has successfully achieved the goal of providing personalized recommendations for both experienced athletes and beginners.

The use of fuzzy and Bayesian logic has effectively addressed the inherent uncertainty in the data and optimized fitness decision making. Since we provide rules that take in consideration common and imprecise language by using fuzzy logic. Moreover, in terms of the conclusions associated to health there are rules that can only lead with probabilities, in this type of rules we use Bayesian reasoning. Experienced athletes can maximize their performance and prevent injuries, while beginners can embark on their fitness journey gradually and safely.

The integration of the UPAFuzzySystems library and Twilio has further enhanced the functionality of the system, providing effective communication with users and making it more accessible and valuable as a personalized fitness improvement tool.
We found that our expert system offers the possibility to save and send information on the user's cell phone thanks to twilio library.

Future work:
In future work related to "Fuzzy-Bayesian Expert System for Suggesting Personalized Training Plans with Exercises and Routines", the following areas of improvement and expansion can be considered:

- Incorporation of Real-Time Data: currently, the system is based on user-provided information at the time. You can explore the possibility of integrating real-time data, such as physical activity tracking, energy levels, or biometric data. This would allow you to further fine-tune the recommendations.
- Machine Learning: You can investigate the integration of machine learning algorithms to improve the accuracy of recommendations over time. The system could learn from user responses and results to refine its suggestions.
- Advanced Personalization: Enhance personalization capabilities by considering additional factors, such as personal preferences, physical limitations, or even location preferences (e.g., whether a user prefers to work out at home or at a gym).
- Wearable Device Integration: Consider integrating wearable devices, such as smartwatches or fitness trackers, to collect real-time physical activity and health data. This can help adjust recommendations according to the user's actual performance.
- More Exercise Options: expand the database of exercises and routines to encompass a wider range of fitness preferences and goals. This could include niche or sport-specific exercises.
- Long-Term Progress Monitoring: Develop a feature that allows users to track their progress over the long term and set future goals. This could motivate users to stay committed to their training plans.
- Nutritional Tips: Add a section for personalized dietary recommendations to complement training plans. Nutrition plays an important role in fitness, and this addition could improve the effectiveness of the system.
- Data Quality Analysis: Perform an in-depth analysis of the quality of the input data. Ensure that the system can handle incomplete or inconsistent data effectively.

The following is a proposed model for a more interactive development.

Proposal of a Model for Developing Interactive

It is essential to provide effective exercise experiences that are adapted to the user's needs. To address this demand, a comprehensive model has been developed that unites the MDA (Model-Driven Architecture) approach with interactive models, workflows and supporting technologies such as SPEM, HASMSTERS and FIGMA.

Within this model, we will explore interrelated elements from the outset to accomplish specific tasks:

- Main Workflow: Represents the fundamental structure of the development process, defining the key phases.
- Specific Workflow: Details how specific models will be applied to design activities taken from the main flow. The content is closely related to user experiences.
- Prototyping and Supporting Technology: We will use tools and technologies such as SPEM (Software & Systems Process Engineering Metamodel), HAMSTERS (Holistic Approach to Modeling Software and Services), and FIGMA (an online collaborative design platform) to plan and design effectively.

- MDA Approach: CIM, PIM and PSM. We adopt a holistic process that encompasses three essential levels of abstraction: the CIM (Conceptualization), the PIM (Specification) and the PSM (Realization). Each of these levels plays a fundamental role: CIM (Conceptualization): In this initial phase, we project an overall vision focused on objectives and general requirements. We establish the conceptual foundations that will help us to achieve our goals. PIM (Specification): Next, we dive into the specification level, where we trans-late the CIM concepts into more detailed and specific models. Here we define the activities and apply specific models to design the interaction between the user and the environment, focusing on how to achieve the objectives (training level) defined in the CIM and on the creation of interactive experiences for the user. PSM (Realization): The realization phase is where we materialize and translate the PIM models into concrete implementations, which includes the creation of interactive content. Here, the models and specifications are turned into reality by leveraging supporting technologies such as SPEM, HAMSTERS and FIGMA to achieve effective implementation (see Fig. 18).

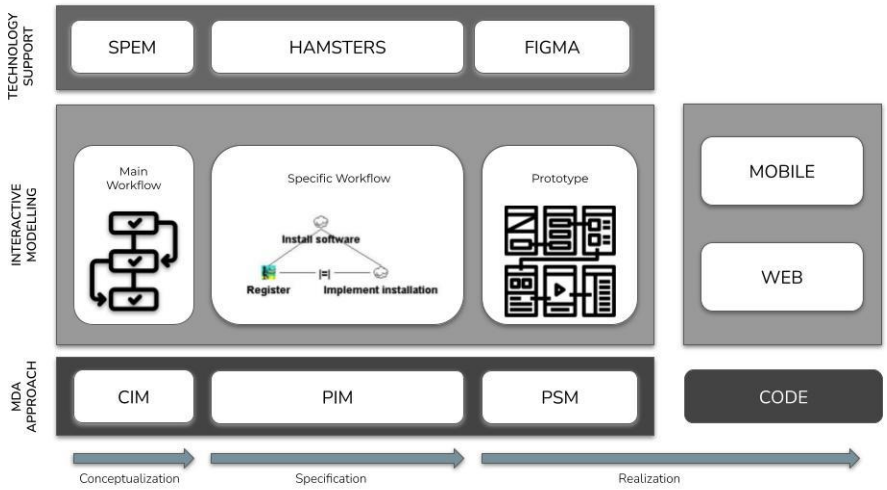


Fig. 18 Technology Development Methodology

The proposed model for the development of exercise environments represents a robust, user-driven approach to improving the health and training process in specific contexts. This model, based on Model Driven Architecture (MDA), focuses on creating effective and adaptive experiences.

References

Botto-Tobar, M., Zambrano Vizuete, M., Torres-Carrión, P., Montes León, S., Pizarro Vásquez, G., & Durakovic, B. (Eds.). (2020). *Applied Technologies: First International Conference, ICAT 2019, Quito, Ecuador, December 3–5, 2019, Proceedings, Part II* (Vol. 1194). Springer International Publishing. https://doi.org/10.1007/978-3-030-42520-3.

Bystrenina, I., & Nikitin, P. (2022). Adaptive Knowledge Control in Digital Learning as a Factor in Improving the Quality of Education. *Education Sciences*, *12*(10). https://doi.org/10.3390/educsci12100638.

Cárdenas López, G., Sánchez, B., & Castillo, E. (2016). Desarrollo y evaluación de simuladores virtuales para la enseñanza de competencias en el campo de la salud. *Assensus*, *1*(1), 59-73. https://doi.org/10.21897/assensus.1284

Colque C, D., & Valdivia P, R. (2006). INTEGRACIÓN DE TECNOLOGÍAS EN UNA PLATAFORMA J2EE DIRIGIDA POR MODELOS. *Ingeniare. Revista Chilena de Ingeniería*, *14*(3). https://doi.org/10.4067/S0718-33052006000200010

D B Beniz & A M Espindola. (2016). *Using Tkinter of Python to create Graphical User Interfaces (GUI) for scripts in LNLS Architecture Overview Source Code Excerpts Tkinter Solution of DXAS*. https://doi.org/10.13140/RG.2.2.14230.86084

Grunau, G., & Linn, S. (2018). Detection and Diagnostic Overall Accuracy Measures of Medical Tests. *Rambam Maimonides Medical Journal*, *9*(4), e0027. https://doi.org/10.5041/RMMJ.10351

Martínez, Y., Cachero, C., & Meliá, S. (2014). Empirical study on the maintainability of Web applications: Model-driven Engineering vs Code-centric. *Empirical Software Engineering*, *19*(6), 1887-1920. https://doi.org/10.1007/s10664-013-9269-5

Montes Rivera, M., Olvera-Gonzalez, E., & Escalante-Garcia, N. (2023). UPAFuzzySystems: A Python Library for Control and Simulation with Fuzzy Inference Systems. *Machines*, *11*(5), 572. https://doi.org/10.3390/machines11050572

Pascuas Rengifo, Y. S., Mendoza Suarez, J. A., & Córdoba Correa, E. D. (2015). Desarrollo Dirigido por Modelo s (MDD) en el Contexto Educativo. *Scientia et technica*, *20*(2), 172. https://doi.org/10.22517/23447214.9321

Penalva Martínez, M. C., & Posadas García, J. A. (2009). El planteamiento de problemas y la construcción del Teorema de Bayes. *Enseñanza de las Ciencias. Revista de investigación y experiencias didácticas*, *27*(3), 331-342. https://doi.org/10.5565/rev/ensciencias.3645

Perbawawati, A. A., Sugiharti, E., & Muslim, M. A. (2019). Bayes Theorem and Forward Chaining Method On Expert System for Determine Hypercholesterolemia Drugs. *Scientific Journal of Informatics*, *6*(1), 116-124. https://doi.org/10.15294/sji.v6i1.14149

Praja, H. N., & Yudha, R. P. (2021). Sports Education Learning Program Evaluation In Senior High School. *JUARA : Jurnal Olahraga*, *6*(2), 222-238. https://doi.org/10.33222/juara.v6i2.1215

Raimondi, G., Martynenko, A., Barsi, L., & Maliarova, L. (2021). Heart rate variability series analyzing by fuzzy logic approach. *The Journal of V. N. Karazin Kharkiv National University, Series «Medicine»*, *43*, 5-12. https://doi.org/10.26565/2313-6693-2021-43-01

Refni Wahyuni & Yuda Irawan. (2019). Web-Based Heart Disease Diagnosis System With Forward Chaining Method (Case Study Of Ibnu Sina Islamic Hospital). *Journal of Applied Engineering and Technological Science (JAETS)*, *1*(1), 43-50. https://doi.org/10.37385/jaets.v1i1.19

Romeiser, J. L., Cavalcante, J., Richman, D. C., Singh, S. M., Liang, X., Pei, A., Sharma, S., Lazarus, Z., Gan, T. J., & Bennett-Guerrero, E. (2021). Comparing Email, SMS, and Concurrent Mixed Modes Approaches to Capture Quality of Recovery in the Perioperative Period: Retrospective Longitudinal Cohort Study. *JMIR Formative Research*, *5*(11), e25209. https://doi.org/10.2196/25209

Sharples, M., Adams, M., Graham, C., McMahon, T., Baxter, G., & Wood, D. (s. f.). *Investigating the use of mobile and wearable devices to support learning in museums and galleries*. Mobile Learning Association.

Unsworth, H., Dillon, B., Collinson, L., Powell, H., Salmon, M., Oladapo, T., Ayiku, L., Shield, G., Holden, J., Patel, N., Campbell, M., Greaves, F., Joshi, I., Powell, J., & Tonnel, A. (2021). The NICE Evidence Standards Framework for digital health and care technologies – Developing and maintaining an innovative evidence framework with global impact. *DIGITAL HEALTH*, *7*, 205520762110186. https://doi.org/10.1177/20552076211018617