International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 6 Issue 7, July - 2022, Pages: 1-12

Designing Expert Systems for Diagnosing the Diseases Using the Rule-Based and Neural Networks Methods

Majid Pakdel

Malek Ashtar University of Technology, Tehran, Iran

Abstract*: Currently, the four most common illnesses are allergies, Covid-19, the flu, and the upper respiratory infection (URI), which have more or less similar symptoms. These symptoms can be mild or very severe because the patient feels almost the same symptoms. The proposed expert systems are designed to help sufferers with early analysis of the diseases, perform preventive and therapeutic measures and, if necessary, consult a specialist. Considering the limitations of the expert system with the rule-based method that many rules should be written to diagnose the type of disease with different symptoms, to diagnose the type of disease with different symptoms the machine learning techniques are used and the type of disease (most likely to have common symptoms) are predicted using the neural networks. The language used in programming the proposed systems is Python.*

Keywords*:* Expert systems; Experta library; Python; neural networks; disease diagnosis

1. Introduction

In the midst of a worldwide epidemic and with the flu season approaching, the capacity to differentiate among specific signs and symptoms and right care may be challenging. In a virtual conversation hosted by the University's Department of Continuing and International Education, medical experts at UHealth and the American Heart Association examined differences between common diagnoses and treatment recommendations. Part of the allergy problem is that more than 50 million Americans have the allergic disease. It is that the sixth leading reason behind chronic unhealthiness within the USA and is the leading cause for youngsters [1].

In addition, allergies are often treated or associated with conditions that are treated by other specialists, including pulmonologist and ENT specialists who deal with Covid-19. Examining the symptoms observed in allergies, Covid-19, influenza, and upper respiratory infections (URI), some of the most important distinguishing factors are whether there is fever, dry cough, or loss of taste and smell or not. While the loss of taste and smell does not occur in the flu and is a rare occurrence in allergies and URIs, 50% of patients with Covid-19 actually experience this loss [1, 2].

Similarities between Covid-19 symptoms and allergies make it difficult to distinguish between them. The first two case studies were of a 61-year-old woman with a history of seasonal allergies who had no fever or shortness of breath but experienced nasal congestion, runny nose, and a mild, intermittent cough for several days. The second was a 40-year-old woman who experienced a decrease in smell and taste within a few days but had no fever, cough, shortness of breath, nasal congestion or headache. Most people who know that olfactory and gustatory loss is a major problem will probably say that the Covid-19 test for the second one was positive, but in fact, both patients tested positive for Covid-19. The key point here is that the husband of the first person had been positive for Covid-19. Screening and requesting a records is critical due to the fact something that appears very simple, which include an allergy, can seem as every other disorder, consisting of Covid-19 [1, 3]. In addition to allergies, the flu also poses challenges. While the flu and Covid-19 have many similarities, health experts strongly encourage everyone to get the flu vaccine to help reduce the confusion of symptoms and the possibility of two viruses interacting. Therefore, receiving the flu vaccine seems necessary [1,4].

In terms of the adverse effects of untreated or uncontrolled allergies, there is an increase in patients who have experienced complications due to untreated allergic disorders. Because the epidemic caused several clinics to shut throughout quarantine, fewer patients had access to hypersensitivity reaction treatment, together with hypersensitivity reaction therapy injections. As a result, patients with these poorly managed chronic conditions have experienced its side effects. 23% of individuals hospitalized for Covid-19 have experienced serious vessel complications. While the epidemic has impaired the ability to receive allergy treatment, it has also disrupted general routines, including exercise, leading to a domino effect that contributes to high blood pressure and obesity. There is a relation between allergies and vessel disease. The better the extent of allergy, the greater not unusual place vessel disease. The simplest thanks to perceive this relationship is to know the role of inflammation. Uncontrolled inflammation could be a common feature. An allergic event might not be enough to cause severe aspect effects. But chronic allergic inflammation can affect the cardiovascular system and impair quality of life. Since allergies are still unavoidable for many, CDC guidelines are currently in place to reduce the prevalence of Covid-19 and protect the community from seasonal flu [1, 5].

Table 1 inspects the symptoms of four form of diseases: Allergy, Covid-19, Influenza and Upper Respiratory Infection (URI) Cold. This table is used to design the proposed expert systems using the rule-based and neural networks methods.

Table 1: Comparison of symptoms of four types of diseases: Allergy, Covid-19, Influenza and Upper Respiratory Infection

SYMPTOMS	ALLERGIES	COVID-19	INFLUENZA	URI-COLD
Fever	None	30% None	Yes	Rare
Smell/Taste Loss	Rare	50% Yes	No	Rare
Cough	Asthma Only	Dry	Wet or Dry	Mild
Short of Breath	No	Yes, Not Always	Rare	No
Fatigue	No	Mild to Moderate	Moderate	Rare
Sneezing	Yes	Rare	Rare	Yes
Runny Nose	Yes	Yes or Stuffy	Rare	Yes
Itchy	Eyes, Nose	Rare Rash	No	No
Headache	Rare	Yes	Rare	No
Aches and Pains	No	Yes	Yes	Possibly
Sore Throat	Tickle	Yes, Most	Yes	Yes
Diarrhea	Possibly	12% Yes	Possibly	No

(URI) Cold [1]

2. Related works

Artificial intelligence systems are now widely used in economics, medicine, engineering and social media and are also embedded in many software applications and computer strategy games such as computer chess and computer video games. An expert system can process huge quantities of regarded facts and use reasoning skills to attract conclusions. An expert system is a system that uses human knowledge gathered in an automated system to solve problems that usually require human expertise.

Expert systems are computer programs derived from artificial intelligence (AI). The scientific goal of artificial intelligence is to understand intelligence and gain insight by building computer programs that behave intelligently [6]. This includes the concepts and methods of symbolic inference or reasoning by the computer and the mechanism of using knowledge for inference within the machine. The development of traditional expert systems was done by developing the Lisp and Prolog symbolic processing languages [7]. To prevent rework, expert system shells were created with specialized features to build large expert systems. Recent advances in artificial intelligence have led to the emergence of expert medical systems [8]. One of the most successful expert systems was MYCIN, an early expert system designed to identify bacteria that cause severe infections such as bacteremia and meningitis and to recommend antibiotic treatment [9]. Creating a reliable medical diagnostic system is a complex task. Recently, several expert systems based on fuzzy logic have been developed to diagnose diseases specific to coronary artery disease [10] and chronic kidney disease [11].

In this paper, we aim to develop a medical expert system based on general knowledge. This medical expert system has a range of screening and diagnosis based on some of the symptoms and features of major and common diseases. Experta is a Python library for building expert systems that is heavily inspired by CLIPS [12]. For the reasons mentioned above, the proposed expert system is designed to help diagnose the disease early, perform preventive and therapeutic measures and, if necessary, recommend to see a specialized doctor. Expert System (ES) is a computer program that, like a skilled human expert, has decision-making intelligence in solving problems and uses the skills and specialized information provided by some experts as a computer consulting service. In this article, due to the limitations of the expert system with the rule-based method that many rules must be written to diagnose the type of diseases with different symptoms, machine learning techniques are used to diagnose the type of disease with some different symptoms and the type of disease with the most likely common symptoms is predicted using neural networks.

3. An expert frame for diagnosing the diseases

An expert system consists of two main parts, a knowledge base and an inference engine. The inference engine argues for a knowledge base like a human to run an expert system. An interface is used to communicate between users and the system that has the ability to communicate with users. Our specialist medical system is built with the usual components of an expert system. Fig. 1 shows the high-level design of an expert system based on knowledge-based disease diagnosis. This system has all the essential components that an expert system should have. The main components are:

• Knowledge Base - A declaration of expertise can be based on IF-THEN rules or knowledge-based representations such as semantic networks and frame-based representations.

• Inference engine - is the core of the system that extracts recommendations from the knowledge base and data related to working memory problems.

• Working memory - Used to store user input and for more matching queries, which helps avoid duplicate questions.

• User Interface - The interface that controls the interaction between the user and the main expert system.

The basic components of an expert system are shown in Fig. 1.

3.1. Knowledge structure

Knowledge construction was based on a concept of rules that were implemented in a hierarchical tree structure. For this part, rule-based knowledge with a hierarchical tree structure is used to demonstrate knowledge about different diseases and their symptoms. This system formats its internal rules with header information and all spaces are connected with AND/OR conditions.

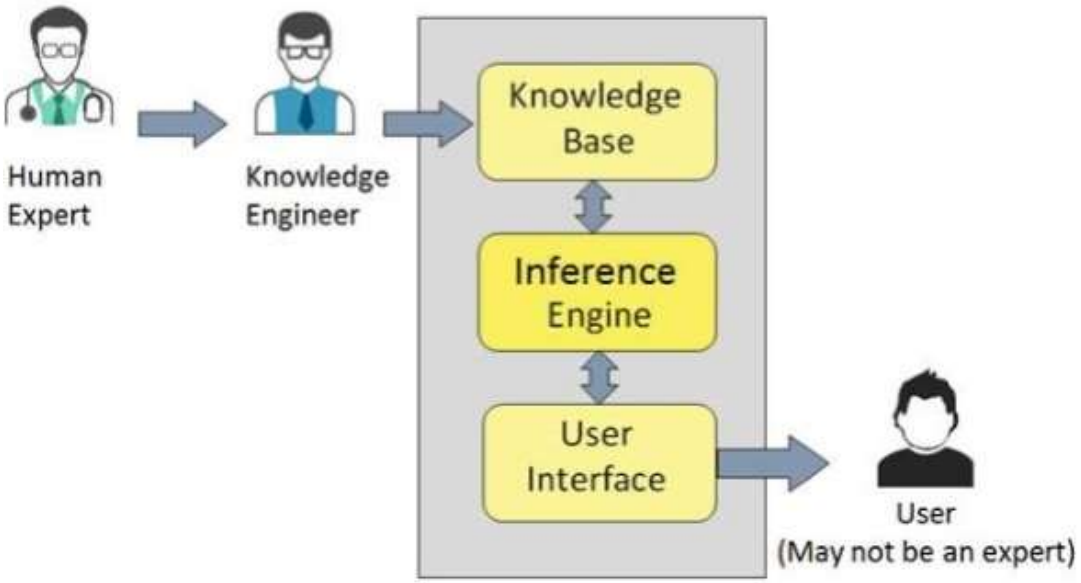

Fig. 1. Basic components of an expert system [13]

This frame has a section to display the coefficient of certainty of the rules. This confidence factor is used by the inference engine to decide on a possible solution. A rule can be nested, meaning that a rule can have nested rules and different facts.

3.2. Inference engine

An inference engine is used to extract answers from the knowledge base. A finite state machine in the inference engine can be described by a cycle of three modes of operation as matching rules, selection rules, and execution rules. The rules are designed to be a hierarchical tree-like structure to demonstrate knowledge about different diseases and their symptoms. With the first state of matching rules, the inference engine searches for all the rules that are met by the current content of the target data. In the latter case, the inference engine implements some selection strategies to determine which rules actually apply. Finally, the inference engine

executes the selected rules with the prototype data items as a parameter. Using this set of rules, the inference engine of this system can argue the knowledge of diseases with multiple symptoms. The main behaviors of the inference engine are:

- Combine the confidence factors as mentioned earlier.
- Preserve working memory that is updated with new evidence.
- When a feature is requested, find all the specific information and save that information in the working memory.

3.3. Working memory

Activated memory can be considered as brain processes used for temporary storage and manipulation of information. It operates in a short window of time, usually a few seconds, helping users focus their attention, resist distractions, and guide decisions. Working memory simply contains the known facts about attribute-value pairs. Known facts are stored in the database for further use. If a current fact and its definite fact is already stored in working memory, the system will not ask users any more questions about it and will use the stored data to calculate the definite fact.

3.4. User interface

One of the important contributions to the expert system is the user interface to facilitate the visualization of the decision process. Traditionally, most program outputs are texts that make it easier for users to understand the solution provided by expert systems.

4. Knowledge demonstration by the rule-based method

The proposed expert system asks the consumer to reply questions on the signs of the sickness and in the end diagnoses the type of disease, personal care methods and how to deal with the disease and recommendations for it. The main sources of knowledge in this expert system are the symptoms of the four common diseases shown in Table 1, and this knowledge has been used to construct facts and rules using the Python programming language and the Experta library used to build expert systems. The proposed expert system, using the knowledge obtained from a specialized website [1] and Table 1, asks the user to answer the questions, through which the proposed expert system diagnoses the type of disease and provides recommendations to the user. The proposed expert system for diagnosing 4 types of diseases includes 20 rules. The rules and code of the expert system related to diagnosing 4 types of diseases are presented below.

```
from experta import *

class Loads(KnowledgeEngine):
    @DefFacts()
    def _initial_action(self):

        yield Fact(Problem="desease")
    print(" An Expert System for Diagnosing ".center(80, "*"))
    print(" Desease ".center(80, "*"))
    print(" Programmed by Majid Pakdel ".center(80, "*"))
    print("".center(80, "*"))
    print("".center(80, "*"))

    # Rule 1
    @Rule(Fact(Problem='desease'), NOT(Fact(Q1=W())))
    def ask_1(self):
        self.declare(Fact(Q1=input("Q1: Do you have fever? Please type
(none/30%none/yes/rare): ")))

    # Rule 2
    @Rule(Fact(Problem='desease'), NOT(Fact(Q2=W())))
    def ask_2(self):
        self.declare(Fact(Q2=input("Q2: Do you have taste loss? type (rare/50%yes/no):
")))

    # Rule 3
```

```
    @Rule(Fact(Problem='desease'), NOT(Fact(Q3=W())))
    def ask_3(self):
        self.declare(Fact(Q3=input("Q3: Do you have cough? Please type
(asthma_only/dry/wet_or_dry/mild): ")))

    # Rule 4
    @Rule(Fact(Problem='desease'), NOT(Fact(Q4=W())))
    def ask_4(self):
       self.declare(Fact(Q4=input("Q4: Do you have short of breath? Please type
(no/yes_not_always/rare): ")))

    # Rule 5
    @Rule(Fact(Problem='desease'), NOT(Fact(Q5=W())))
    def ask_5(self):
        self.declare(Fact(Q5=input("Q5: Do you have fatigue? Please type
(no/mild_to_moderate/moderate/rare): ")))

    # Rule 6
    @Rule(Fact(Problem='desease'), NOT(Fact(Q6=W())))
    def ask_6(self):
        self.declare(Fact(Q6=input("Q6: Do you have sneezing? type (yes/rare): ")))

    # Rule 7
    @Rule(Fact(Problem='desease'), NOT(Fact(Q7=W())))
    def ask_7(self):
        self.declare(Fact(Q7=input("Q7: Do you have runny nose? Please type
(yes/yes_or_stuffy/rare): ")))

    # Rule 8
    @Rule(Fact(Problem='desease'), NOT(Fact(Q8=W())))
    def ask_8(self):
       self.declare(Fact(Q8=input("Q8: Do you have itchy? Please type
(eyes_nose/rare_rash/no): ")))

    # Rule 9
    @Rule(Fact(Problem='desease'), NOT(Fact(Q9=W())))
    def ask_9(self):
        self.declare(Fact(Q9=input("Q9: Do you have headache? Please type
(rare/yes/no): ")))

    # Rule 10
    @Rule(Fact(Problem='desease'), NOT(Fact(Q10=W())))
    def ask_10(self):
        self.declare(Fact(Q10=input("Q10: Do you have aches and pains? type
(no/yes/possibly): ")))

    # Rule 11
    @Rule(Fact(Problem='desease'), NOT(Fact(Q11=W())))
    def ask_11(self):
        self.declare(Fact(Q11=input("Q11: Do you have sore throat? Please type
(tickle/yes_most/yes): ")))

    # Rule 12
    @Rule(Fact(Problem='desease'), NOT(Fact(Q12=W())))
    def ask_12(self):
       self.declare(Fact(Q12=input("Q12: Do you have diarrhea? Please type
```

```
(possibly/12%yes/no): ")))

    # Rule 13
    @Rule(Fact(Problem='desease'),(AND(Fact(Q1='none'), Fact(Q2='rare'),
Fact(Q3='asthma_only'), Fact(Q4='no'),
                                      Fact(Q5='no'), Fact(Q6='yes'), Fact(Q7='yes'),
Fact(Q8='eyes_nose'),
                                      Fact(Q9='rare'), Fact(Q10='no'),
Fact(Q11='tickle'), Fact(Q12='possibly'))))
    def diagnosis_1(self):
        print("".center(80, "*"))
        print("Allergies!".center(20, "*"))
        print("".center(80, "*"))
        print("SELF CARE".center(20, "*"))
        print("For more information, please talk to your doctor. If you think the
problem is serious,",
              "call your doctor right away.")

    # Rule 14
    @Rule(Fact(Problem='desease'),(AND(Fact(Q1='30%none'), Fact(Q2='50%yes'),
Fact(Q3='dry'), Fact(Q4='yes_not_always'),
                                      Fact(Q5='mild_to_moderate'), Fact(Q6='rare'),
Fact(Q7='yes_or_stuffy'),
                                      Fact(Q8='rare_rash'), Fact(Q9='yes'),
Fact(Q10='yes'), Fact(Q11='yes_most'),
                                      Fact(Q12='12%yes'))))
    def diagnosis_2(self):
        print("".center(80, "*"))
        print("Covid-19!".center(20, "*"))
        print("".center(80, "*"))
        print("SELF CARE".center(20, "*"))
        print("For more information, please talk to your doctor. If you think the
problem is serious,",
              "call your doctor right away.")

    # Rule 15
    @Rule(Fact(Problem='desease'),(AND(Fact(Q1='yes'), Fact(Q2='no'),
Fact(Q3='wet_or_dry'), Fact(Q4='rare'),
                                      Fact(Q5='moderate'), Fact(Q6='rare'),
Fact(Q7='rare'), Fact(Q8='no'),
                                      Fact(Q9='rare'), Fact(Q10='yes'),
Fact(Q11='yes'), Fact(Q12='possibly'))))
    def diagnosis_3(self):
        print("".center(80, "*"))
        print("Influenza!".center(20, "*"))
        print("".center(80, "*"))
        print("SELF CARE".center(20, "*"))
        print("For more information, please talk to your doctor. If you think the
problem is serious,",
              "call your doctor right away.")

    # Rule 16
    @Rule(Fact(Problem='desease'),(AND(Fact(Q1='rare'), Fact(Q2='rare'),
Fact(Q3='mild'), Fact(Q4='no'),
                                      Fact(Q5='rare'), Fact(Q6='yes'), Fact(Q7='yes'),
Fact(Q8='no'),
```

```
                                        Fact(Q9='no'), Fact(Q10='possibly'),
Fact(Q11='yes'), Fact(Q12='no'))))
    def diagnosis_4(self):
        print("".center(80, "*"))
        print("URI-Cold!".center(20, "*"))
        print("".center(80, "*"))
        print("SELF CARE".center(20, "*"))
        print("For more information, please talk to your doctor. If you think the
problem is serious,",
              "call your doctor right away.")

    # Rule 17
    @Rule(Fact(Problem='desease'),(AND(Fact(Q1='none'), Fact(Q6='yes'), Fact(Q7='yes'),
Fact(Q8='eyes_nose'))),
          (OR(Fact(Q10='yes'), Fact(Q2='no'), Fact(Q3='wet_or_dry'), Fact(Q4='rare'),
Fact(Q5='rare'), Fact(Q9='no'),
              Fact(Q11='yes'), Fact(Q12='no'))))
    def diagnosis_5(self):
        print("".center(80, "*"))
        print("80% Allergies!".center(20, "*"))
        print("".center(80, "*"))
        print("SELF CARE".center(20, "*"))
        print("For more information, please talk to your doctor. If you think the
problem is serious,",
              "call your doctor right away.")

    # Rule 18
    @Rule(Fact(Problem='desease'),(AND(Fact(Q2='50%yes'), Fact(Q3='dry'),
Fact(Q4='yes_not_always'), Fact(Q9='yes'),
          Fact(Q10='yes'), Fact(Q11='yes_most'))), (OR(Fact(Q1='none'),
Fact(Q5='moderate'), Fact(Q6='yes'),
                                                    Fact(Q7='yes'), Fact(Q8='no'),
Fact(Q12='possibly'))))
    def diagnosis_6(self):
        print("".center(80, "*"))
        print("80% Covid-19!".center(20, "*"))
        print("".center(80, "*"))
        print("SELF CARE".center(20, "*"))
        print("For more information, please talk to your doctor. If you think the
problem is serious,",
              "call your doctor right away.")

    # Rule 19
    @Rule(Fact(Problem='desease'),(AND(Fact(Q1='yes'), Fact(Q2='no'),
Fact(Q3='wet_or_dry'), Fact(Q10='yes'),
                                        Fact(Q11='yes'), Fact(Q8='no'))),
(OR(Fact(Q4='no'), Fact(Q5='rare'),

Fact(Q6='yes'), Fact(Q7='yes'),

Fact(Q9='no'), Fact(Q12='no'))))
    def diagnosis_7(self):
        print("".center(80, "*"))
        print("80% Influenza!".center(20, "*"))
        print("".center(80, "*"))
        print("SELF CARE".center(20, "*"))
```

```
        print("For more information, please talk to your doctor. If you think the
problem is serious,",
              "call your doctor right away.")

    # Rule 20
    @Rule(Fact(Problem='desease'),(AND(Fact(Q3='mild'), Fact(Q6='yes'), Fact(Q7='yes'),
Fact(Q10='possibly'),
                                       Fact(Q11='yes'), Fact(Q8='no'))),
(OR(Fact(Q1='yes'), Fact(Q2='no'),

Fact(Q4='rare'), Fact(Q5='no'),

Fact(Q9='rare'), Fact(Q12='possibly'))))
    def diagnosis_8(self):
        print("".center(80, "*"))
        print("80% URI-Cold!".center(20, "*"))
        print("".center(80, "*"))
        print("SELF CARE".center(20, "*"))
        print("For more information, please talk to your doctor. If you think the
problem is serious,",
              "call your doctor right away.")

engine = Loads()
engine.reset()
engine.run()
input('press enter to exit ')
```

5. Knowledge demonstration through neural networks method

Multilayer perceptron (MLP) is a supervised learning algorithm that learns the function $f(.): R^m \rightarrow R^o$ by training on a data set, where m is the number of dimensions for input and o is the number of dimensions for the output. Given a set of features $X = x_1, x_2, \ldots, x_m$ and object y, it can learn a nonlinear function approximation for classification or regression. This isn't the same as logistic regression, due to the fact there may be one or greater nonlinear layers among the enter and output layers, known as hidden layers. Fig. 2 suggests an MLP with a hidden layer with a scalar output.

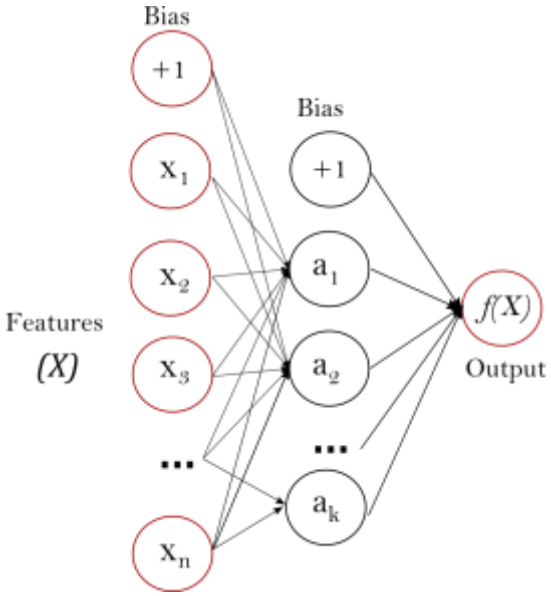

Fig. 2. MLP with a hidden layer

The leftmost layer, known as the input layer, is composed of a set of neurons $\{x_i|x_1, x_2, \dots, x_m\}$ that represent the input features. Each neuron with inside the hidden layer converts the values of the preceding layer to a weighted linear sum $w_1x_1 + w_2x_2 + \cdots + w_mx_m$ followed by a nonlinear activation function $g(.):R \to R$ (such as the hyperbolic tangent function) changes. The output layer receives the values of the last hidden layer and converts them to output values. The MLP classifier implements a multilayer perceptron (MLP) algorithm that trains using the back propagation method.

The benefits of multilayer perceptron include:

• Ability to learn nonlinear models

• Ability to learn models in real time (online training)

Disadvantages of multilayer perceptron (MLP) include:

• MLP with hidden layers has a non-convex cost function where there is more than one local minimum. Thus different initial values of random weight can lead to different validation accuracy.

• MLP requires the adjustment of a number of hyper parameters such as the number of hidden neurons, layers and repeats.

• MLP is sensitive to feature scaling.

Below we present the expert system code related to the diagnosis of 4 types of diseases by displaying knowledge through neural networks.

```
from sklearn.neural_network import MLPClassifier

X = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [2, 2,
2, 2, 2, 1, 2, 2, 0, 1, 2, 0],
     [3, 0, 3, 0, 3, 0, 0, 2, 2, 2, 2, 2]]
y = [0, 1, 2, 3]
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(4,),
random_state=1)
clf.fit(X, y)

Q1 = input("Q1: Do you have fever? Please type (none/30%none/yes/rare): ")
Q2 = input("Q2: Do you have taste loss? type (rare/50%yes/no): ")
Q3 = input("Q3: Do you have cough? Please type (asthma_only/dry/wet_or_dry/mild): ")
Q4 = input("Q4: Do you have short of breath? Please type (no/yes_not_always/rare): ")
Q5 = input("Q5: Do you have fatigue? Please type (no/mild_to_moderate/moderate/rare):
")
Q6 = input("Q6: Do you have sneezing? type (yes/rare): ")
Q7 = input("Q7: Do you have runny nose? Please type (yes/yes_or_stuffy/rare): ")
Q8 = input("Q8: Do you have itchy? Please type (eyes_nose/rare_rash/no): ")
Q9 = input("Q9: Do you have headache? Please type (rare/yes/no): ")
Q10 = input("Q10: Do you have aches and pains? type (no/yes/possibly): ")
Q11 = input("Q11: Do you have sore throat? Please type (tickle/yes_most/yes): ")
Q12 = input("Q12: Do you have diarrhea? Please type (possibly/12%yes/no): ")

if Q1 == 'none':
    fe = 0
elif Q1 == '30%none':
    fe = 1
elif Q1 == 'yes':
    fe = 2
else:
    fe = 3

if Q2 == 'rare':
    tl = 0
elif Q2 == '50%yes':
    tl = 1
else:
    tl = 2
```

```
if Q3 == 'asthma_only':
    co = 0
elif Q3 == 'dry':
    co = 1
elif Q3 == 'wet_or_dry':
    co = 2
else:
    co = 3

if Q4 == 'no':
    sb = 0
elif Q4 == 'yes_not_always':
    sb = 1
else:
    sb = 2

if Q5 == 'no':
    fa = 0
elif Q5 == 'mild_to_moderate':
    fa = 1
elif Q5 == 'moderate':
    fa = 2
else:
    fa = 3

if Q6 == 'yes':
    sn = 0
else:
    sn = 1

if Q7 == 'yes':
    rn = 0
elif Q7 == 'yes_or_stuffy':
    rn = 1
else:
    rn = 2

if Q8 == 'eyes_nose':
    it = 0
elif Q8 == 'rare_rash':
    it = 1
else:
    it = 2

if Q9 == 'rare':
    he = 0
elif Q9 == 'yes':
    he = 1
else:
    he = 2

if Q10 == 'no':
    ap = 0
elif Q10 == 'yes':
    ap = 1
```

```
else:
    ap = 2

if Q11 == 'tickle':
    st = 0
elif Q11 == 'yes_most':
    st = 1
else:
    st = 2

if Q12 == 'possibly':
    di = 0
elif Q12 == '12%yes':
    di = 1
else:
    di = 2

pred = clf.predict([[fe, tl, co, sb, fa, sn, rn, it, he, ap, st, di]])

if pred == 0:
     print("".center(80, "*"))
     print("Allergies!".center(20, "*"))
     print("".center(80, "*"))
     print("SELF CARE".center(20, "*"))
     print("For more information, please talk to your doctor. If you think the problem
is serious,",
           "call your doctor right away.")
elif pred == 1:
     print("".center(80, "*"))
     print("Covid-19!".center(20, "*"))
     print("".center(80, "*"))
     print("SELF CARE".center(20, "*"))
     print("For more information, please talk to your doctor. If you think the problem
is serious,",
           "call your doctor right away.")
elif pred == 2:
     print("".center(80, "*"))
     print("Influenza!".center(20, "*"))
     print("".center(80, "*"))
     print("SELF CARE".center(20, "*"))
     print("For more information, please talk to your doctor. If you think the problem
is serious,",
           "call your doctor right away.")
else:
     print("".center(80, "*"))
     print("URI-Cold!".center(20, "*"))
     print("".center(80, "*"))
     print("SELF CARE".center(20, "*"))
     print("For more information, please talk to your doctor. If you think the problem
is serious,",
           "call your doctor right away.")
```

6. Conclusion

In this paper, two expert systems based on medical knowledge were designed and implemented to diagnose the diseases based on their symptoms. Using a medical expert system is very interesting. The proposed expert system can help professionals and patients

provide support for decision making. This expert system does not require enormous training to use. It is simple to use and the rule-based expert system has been developed using the Python library (Experta). Due to the limitations of the expert system with the rule-based method that many rules must be written to diagnose the type of disease with different symptoms, to diagnose the type of disease with some different symptoms machine learning techniques (neural networks) are used and the type of disease can be predicted with most likely common symptoms. For this reason, an expert system was designed using neural networks with the Python library (scikit-learn). These expert systems are knowledge-based and can be integrated with any rule-based expert system for diagnosing diseases. As a preliminary evaluation, the two expert systems were implemented and received positive feedback from users. For future work, we can increase the capacity of our expert system by enforcing more inference rules to represent knowledge based on semantic networks and conducting real-world studies with patients and medical professionals. Also, the proposed expert systems can be upgraded to have a graphical interface.

Conflict of interest statement

The authors have no conflicts of interest to declare.

References

[1] https://news.miami.edu/life/stories/2020/09/allergies-covid-19-and-the-flu-what-are-the-differences.html.
[2] Goita, Yacouba, and Mohamed Sidibe. "Towards a Comprehensive Expert System for Coronavirus Disease." In *2021 7th International Conference on Information Management (ICIM)*, pp. 18-23. IEEE, 2021.
[3] Almadhoun, Husam R., and Samy S. Abu-Naser. "An Expert System for Diagnosing Coronavirus (COVID-19) Using SL5." (2020).
[4] Hossain, Mohammad Shahadat, Md Saifuddin Khalid, Shamima Akter, and Shati Dey. "A belief rule-based expert system to diagnose influenza." In *2014 9Th international forum on strategic technology (IFOST)*, pp. 113-116. IEEE, 2014.
[5] Zhou, Zhi-Jie, Guan-Yu Hu, Chang-Hua Hu, Cheng-Lin Wen, and Lei-Lei Chang. "A survey of belief rule-base expert system." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, no. 8 (2019): 4944-4958.
[6] G. F. Luger and W. A. Stubblefield, AI Algorithms, Data Structures, and Idioms in Prolog, Lisp, and Java 6th Edition, 2008.
[7] D. Merritt, Building expert systems in Prolog, 2012.
[8] Richens, Jonathan G., Ciarán M. Lee, and Saurabh Johri. "Improving the accuracy of medical diagnosis with causal machine learning." *Nature communications* 11, no. 1 (2020): 1-9.
[9] Watcharachai, Wiriyasuttiwong, and Narkbuakaew Walita. "Medical Knowledge-Based System for Diagnosis from Symptoms and Signs." *international journal of applied biomedical engineering* 2, no. 1 (2009).
[10] Muhammad, L. J., and Ebrahem A. Algehyne. "Fuzzy based expert system for diagnosis of coronary artery disease in Nigeria." *Health and technology* 11, no. 2 (2021): 319-329.
[11] Singla, Jimmy, Balwinder Kaur, Deepak Prashar, Sudan Jha, Gyanendra Prasad Joshi, Kyungyun Park, Usman Tariq, and Changho Seo. "A Novel Fuzzy Logic-Based Medical Expert System for Diagnosis of Chronic Kidney Disease." *Mobile Information Systems* 2020 (2020).
[12] Salman, Fatima M., and Samy S. Abu-Naser. "Expert System for Castor Diseases and Diagnosis." *International Journal of Engineering and Information Systems (IJEAIS)* 3, no. 3 (2019): 1-10.
[13] Almadhoun, Husam R. "Expert System for Diagnosing Urination Problems Using Python." (2020).