Option #2: Healthcare Expert Systems: A Practical Case Study Approach

Scott Miner

Colorado State University – Global Campus

Abstract

This paper presents the development and application of a rule-based expert system (ES) in healthcare, specifically for heart failure telemonitoring. The ES is created in Pyke, a Python-based knowledge-based system that leverages logic programming. The paper discusses the fundamental components of expert systems and the unique attributes of Pyke that make it a versatile tool for developing ESs. It also provides a detailed analysis of a specific rule from the ES, demonstrating the power of Pyke in creating a rule-based ES. The paper further discusses the potential of incorporating inquiry capabilities into the ES for a more comprehensive and personalized analysis of each patient's health. The study concludes with a discussion on the transformative potential of rule-based expert systems in healthcare, particularly in enhancing patient self-care and improving the clinical management of heart failure.

				BP		
				N	High	Low
Weight	Normal	HR	N	1	2	2
			Ab	2	2	2
	High	HR	N	3	3	5
			Ab	3	3	5
	Low	HR	N	4	4	4
			Ab	4	4	4

Figure 1. Outcome Matrix from BP and HR Measurements. The table shows outcome states (normal 'N' or abnormal 'AB') from various BP and HR combinations. Each matrix cell has a number corresponding to a unique patient alert in Table 2.

Alert Number	Patient Message
1	"Your measurements are fine today"
2	"If you feel worse later today, use the system to take symptoms."
3	"Contact family dr. or go to the Emergency Department if you feel you should."
4	"Contact the Heart Function Clinic or your family doctor."
5	"Have someone drive you to the local Emergency Department or call 911 now."

Figure 2. Generated Patient Instructions. This table outlines the instructions produced by the Expert System based on various measurements.

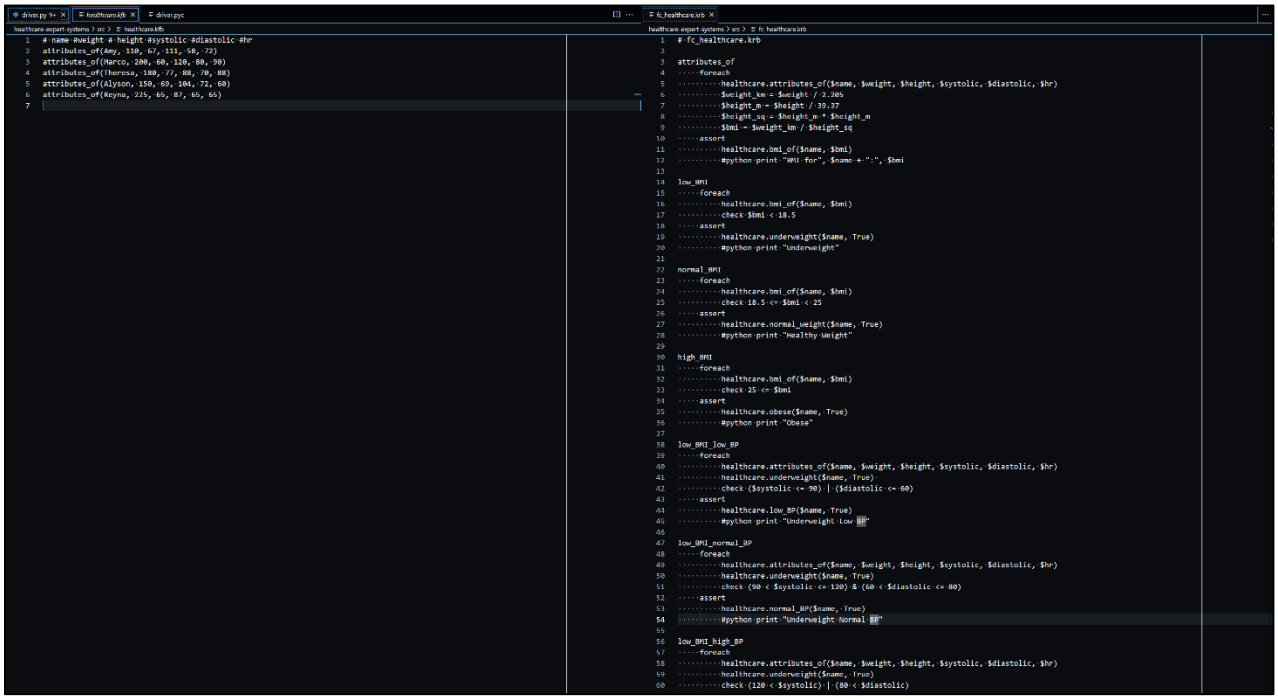

Figure 3. Pyke Fact and Rule Base Files. Display of the 'healthcare.kfb' (fact base) and 'fc_healthcare.krb' (rule base) files used in the Expert System. These files represent the core knowledge and rule sets for healthcare measurements.

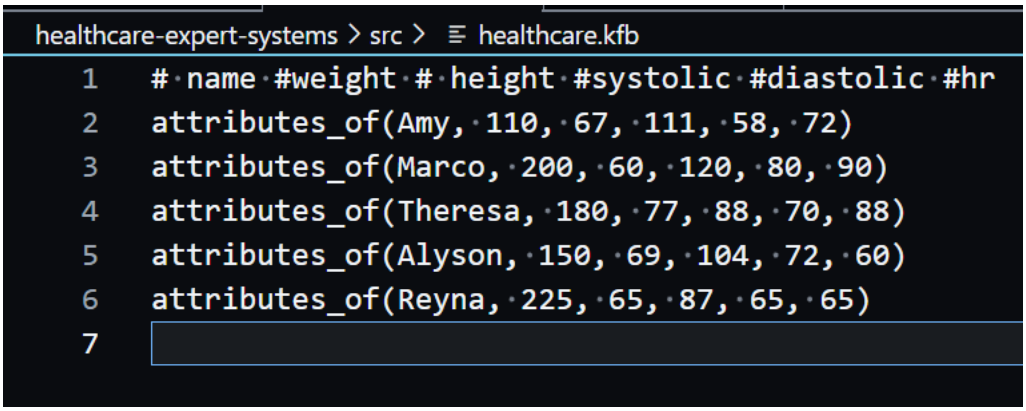

Figure 4. Screenshot of the Knowledge Fact Base (.kfb) file. This file contains the facts that the expert system uses to match with the conditions specified in the rules. These facts are typically straightforward statements about the patients.

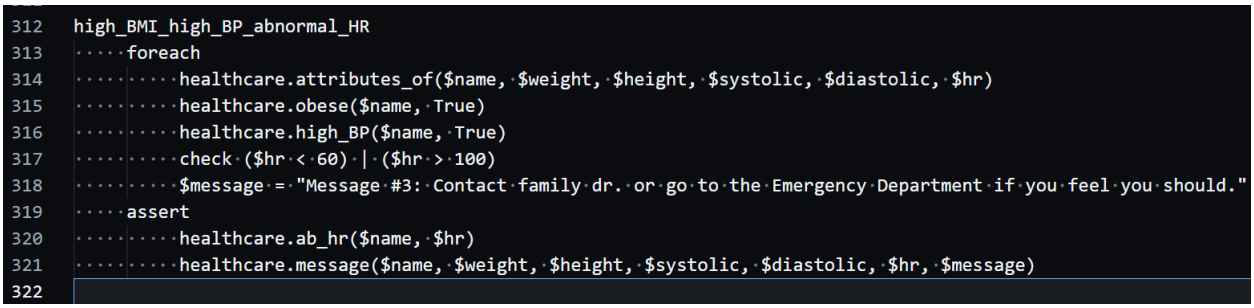

Figure 5. Screenshot of the Knowledge Rule Base (.krb) file. This file contains the rules used by the expert system to make inferences. Each rule is written in a specific syntax that allows the system to process it effectively.

driver.py 9+ driver.py\ fc_test

```python
from __future__ import with_statement
import sys
from pyke import knowledge_engine
from pyke import krb_traceback
from pyke import goal

#Compile and load .krb file in same directory that I'm in (recursively).
engine = knowledge_engine.engine(__file__)

#Compiling the health goal
fc_goal = goal.compile('healthcare.message($name, $weight, $height, $systolic, $diastolic, $hr, $message)')


def fc_test():
    #Allows for running tests multiple times
    engine.reset()
    try:
        #Activating the engine - fires all FC rules
        engine.activate('fc_healthcare')
        with fc_goal.prove(engine) as gen:
            print '*************************************************************************'
            print '*                                                                       *'
            print '*   This is a test output for a Healthcare Expert System (ES) using     *'
            print '*   patient fact information from the healthcare.kfb fact base to fire  *'
            print '*   rules in the fc_healthcare.krb rule base using forward-chaining.    *'
            print '*                                                                       *'
            print '*   It generates output to the terminal that supports decision-makers   *'
            print '*   in their decision-making process.                                   *'
            print '*                                                                       *'
            print '*   This program is compatible with Python 2.6.                         *'
            print '*   You can run it by typing the following command in your terminal:    *'
            print '*                                                                       *'
            print '*   $ python driver.py                                                  *'
            print '*                                                                       *'
            print '*************************************************************************'
            for vars, plan in gen:
                #Calculating BMI
                weight_km = vars['weight'] / 2.205
                height_m = vars['height'] / 39.37
                bmi = weight_km / (height_m ** 2)
                #Obtaining Blood Pressure (BP)
                bp = str(vars['systolic']) + '/' + str(vars['diastolic'])
                #Outputting Results
                print "\n   Patient Name: %s" % (vars['name'])
                print "-----------------------------------"
                print "    Weight (lbs): %s" % (vars['weight'])
                print "    Height (inch): %s" % (vars['height'])
                print "    Body Mass Index (BMI): %.2f" % bmi
                print "    Blood Pressure (BP): %s" % (bp)
                print "    Heart Rate (HR): %s" % (vars['hr'])
                print "    %s" % (vars['message'])
                print "-----------------------------------\n"
    except:
        krb_traceback.print_exc()
        sys.exit(1)


if __name__ == "__main__":
    fc_test()
```

Figure 6. Driver Script in Python. The ‘driver.py’ file, which serves as the main script to execute the Expert System, integrating the fact and rule bases to generate patient instructions based on healthcare measurements.

```
$ python driver.py
writing [compiled_krb]/bc_healthcare_bc.py
writing [compiled_krb]/fc_healthcare_fc.py
writing [compiled_krb]/healthcare.fbc
writing [compiled_krb]/questions.qbc
writing [compiled_krb]/compiled_pyke_files.py
************************************************************************
*                                                                      *
*   This is a test output for a Healthcare Expert System (ES) using    *
*   patient fact information from the healthcare.kfb fact base to fire *
*   rules in the fc_healthcare.krb rule base using forward-chaining.   *
*                                                                      *
*   It generates output to the terminal that supports decision-makers  *
*   in their decision-making process.                                  *
*                                                                      *
*   This program is compatible with Python 2.6.                        *
*   You can run it by typing the following command in your terminal:   *
*                                                                      *
*   $ python driver.py                                                 *
*                                                                      *
************************************************************************

   Patient Name: Amy
-----------------------------------
    Weight (lbs): 110
    Height (inch): 67
    Body Mass Index (BMI): 17.23
    Blood Pressure (BP): 111/58
    Heart Rate (HR): 72
    Message #4: Contact the Heart Function Clinic or your family doctor.
-----------------------------------


   Patient Name: Theresa
-----------------------------------
    Weight (lbs): 180
    Height (inch): 77
    Body Mass Index (BMI): 21.34
    Blood Pressure (BP): 88/70
    Heart Rate (HR): 88
    Message #2: If you feel worse later today, use the system to take symptoms.
-----------------------------------


   Patient Name: Alyson
-----------------------------------
    Weight (lbs): 150
    Height (inch): 69
    Body Mass Index (BMI): 22.15
    Blood Pressure (BP): 104/72
    Heart Rate (HR): 60
    Message #1: Your measurements are fine today.
-----------------------------------


   Patient Name: Reyna
-----------------------------------
    Weight (lbs): 225
    Height (inch): 65
    Body Mass Index (BMI): 37.44
    Blood Pressure (BP): 87/65
    Heart Rate (HR): 65
    Message #5: Have someone drive you to the local Emergency Department or call 911 now.
-----------------------------------


   Patient Name: Marco
-----------------------------------
    Weight (lbs): 200
    Height (inch): 60
    Body Mass Index (BMI): 39.05
    Blood Pressure (BP): 120/80
    Heart Rate (HR): 90
    Message #3: Contact family dr. or go to the Emergency Department if you feel you should.
-----------------------------------
```

Figure 7. Successful Program Execution Output. This figure displays the output generated by the successful execution of the program. Please be aware that Python 2.6 is required for the program to run properly.

Table of Contents

Option #2: Healthcare Expert Systems: A Practical Case Study Approach

This paper explores the structure and application of expert systems (ES), with a particular focus on healthcare. Expert systems, as defined by Sharda *et al.* (2020), are advanced computing systems designed to mimic human expert decision-making. These systems aim to empower non-specialists to solve problems that typically require specialized knowledge. Since their inception in the 1980s, ESs have found applications across various sectors, from finance to manufacturing, and from human resources to marketing.

Expert Systems in Healthcare

The healthcare sector presents a particularly interesting and important context for ES due to several reasons. Firstly, ES can play a crucial role in managing chronic conditions and pandemics. For instance, during the COVID-19 pandemic, an international expert panel used ES to formulate recommendations for caring for patients with chronic pain (Shanthanna *et al.*, 2020). Secondly, the rising cost of new medicines is a challenge for healthcare systems worldwide. Expert systems can help optimize the utilization of new medicines, making healthcare systems more sustainable (Godman *et al.*, 2015).

Thirdly, with the rise of the Internet of Medical Things (IoM), patient information is increasingly being collected and analyzed remotely. This raises concerns about the security and privacy of medical data. Expert systems can provide solutions for privacy preservation in smart e-healthcare systems (Deebak *et al.*, 2019). Lastly, automated telephone communication systems, a type of expert system, can be used for preventive healthcare and management of long-term conditions (Posadzki *et al.*, 2016). Despite the emergence of newer systems in the 2010s, understanding the foundational principles of ESs remains crucial as they form the basis for many of today’s advanced systems.

Core Components of Expert Systems

Sharda *et al.* (2020) outline the essential elements of an ES, which include (a) the process of knowledge acquisition, (b) the knowledge base, (c) the means of knowledge representation, (d) the inference engine, and (e) the user interface.

Knowledge Acquisition

Knowledge acquisition is the process of gathering expert knowledge to be used in the ES. Tecuci (1992) views this as a process of incremental extension, updating, and improvement of an incomplete and possibly partially incorrect knowledge base of an expert system. The knowledge base is an approximate representation of objects and inference processes in the expertise domain. Its gradual development is guided by the general goal of improving this representation to consistently integrate new input information received from the human expert (Tecuci, 1992).

Knowledge Base

The knowledge base of an ES contains specific information and rules about a particular field of knowledge. This information is usually provided by a human expert in the field. For instance, Akanbi and Masinde (2018) developed a rule-based drought early warning expert system that uses local indigenous knowledge obtained from domain experts. The system generates inference by using a rule set and provides drought advisory information based on the user's input (Akanbi & Masinde, 2018).

Knowledge Representation

Knowledge representation is how the knowledge in the ES is structured or organized. Morgenstern (1987) discusses the importance of knowledge representation in the context of multilevel database systems and their relevance to knowledge-based systems. The representation

of the relevant semantics of the application is crucial for the safety and security of a system (Morgenstern, 1987).

Inference Engine

The inference engine is the part of the ES that applies the rules to the data to reach conclusions. Sharda *et al.* (2020) often refer to this as the ES's "brain," as it applies system rules through either forward- or backward-chaining methods. According to these authors, traditional ESs employ a sequence of "IF-THEN" rules to facilitate reasoning.

User Interface

The user interface is the part of the ES that allows users to interact with the system. Hwang (1995) discusses the importance of a user-friendly interface in the development of fuzzy expert systems. A graphic user interface is provided to facilitate the adjustment of membership functions and the display of outputs (Hwang, 1995).

Development and Consultation Environments

Sharda *et al.* (2020) explain that the development of an ES involves two distinct environments: the development environment and the consultation environment. In the development environment, the ES builder constructs the necessary components and populates the knowledge base with expert knowledge. The consultation environment is where non-experts interact with the system to obtain advice and solve problems.

Evolutionary Challenges of Expert Systems

Sharda *et al.* (2020) highlight that despite their advantages, such as operational cost reduction and quicker problem-solving capabilities, classical ESs face challenges. These include maintenance complexities and less sophisticated reasoning capabilities compared to

contemporary technologies. Modern ESs, for instance, leverage machine learning algorithms to draw inferences, surpassing the effectiveness of traditional rule-based systems.

Jacobs *et al.* (2021) provide an example of this in the healthcare sector, where machine learning models are used in decision support systems to improve the standard of care for major depressive disorder. However, the study also cautions that incorrect machine learning recommendations can adversely impact clinician treatment selections, suggesting that the integration of machine learning in ESs needs to be carefully managed.

Healthcare Expert Systems: A Heart Failure Telemonitoring Case Study

Seto *et al.'s* (2012) research offers a compelling illustration of how expert systems can be effectively applied in the healthcare sector. The study centered on the creation and evaluation of a rule-based expert system, specifically designed for a heart failure telemonitoring system that operates via mobile phones. The primary aim of this system was to generate automated alerts and provide patient guidelines based on the data collected through telemonitoring. This innovative approach aimed to enhance patient self-care and improve the clinical management of heart failure. The study involved 100 patients from a heart function clinic, half of whom were assigned to the telemonitoring group and the other half to the control group.

Formulating the Rule-Based Expert System

The development of the expert system's rule set was a meticulous process. It began with semi-structured interviews with 10 clinicians specializing in heart failure. These interviews informed the creation of a draft rule set, which outlined alerts and patient instructions. This draft underwent a rigorous validation and refinement process, involving an additional nine interviews with heart failure clinicians. The final rule set, which was thoroughly reviewed and approved by the project's clinical champion, ensured precision and applicability.

Integration and Assessment of the Expert System

The expert system was seamlessly integrated into a heart failure telemonitoring system that operated via mobile phones. Patients in the telemonitoring group used wireless medical devices to record vital signs, including weight, blood pressure, and heart rate. This data was automatically transmitted via Bluetooth to the patients' smartphones and subsequently sent to hospital servers for analysis.

The system generated alerts and instructions based on the patient's data. During the trial, the system generated 1,620 alerts, which led to various clinical actions, including 105 medication changes/instructions.

Significance and Impact of the Study

The results of the trial demonstrated a significant improvement in self-care associated with the rule set. The study by Seto *et al.* (2012) highlights the transformative potential of rule-based expert systems in healthcare. The developed rule set significantly increased self-care and improved the clinical management of heart failure, thereby enhancing patient outcomes and healthcare delivery.

The study found that about 70% of telemonitoring patients completed at least 80% of their possible daily readings, and the change in quality of life from baseline to post-study was significantly greater for the telemonitoring group compared to the control group. However, it's important to note that the trial was underpowered to detect differences in hospitalization, mortality, or emergency department visits. This suggests that while the telemonitoring system showed promising results in improving self-care and clinical management, further research is needed to fully understand its impact on other critical outcomes such as hospitalization and mortality.

Developing a Rule-Based Expert System Using Pyke: A Contemporary Approach

This paper presents the development of a simple expert system (ES), created in Pyke, a Python-based knowledge-based system that leverages logic programming. Influenced by a rule set akin to the one used by Seto *et al.* (2012), Pyke allows an ES that emulates the features of Prolog but is entirely crafted in Python. Prolog, short for Programming in Logic, is a high-level programming language associated with artificial intelligence and computational linguistics. It is known for its capacity to express logic, represented as relations, in a form that is easy to read and understand. This unique attribute of Pyke, which enables the seamless integration of Python statements within the ES rules, makes it a versatile tool for developing expert systems that combine the logical expressiveness of Prolog with the wide-ranging functionality of Python. (*Welcome to Pyke*, n.d.).

Understanding the Fundamental Components of Pyke

In Pyke, a rule is composed of ‘if’ and ‘then’ segments, or what are traditionally referred to as premises and conclusions (*Rules*, n.d.). Pyke programs consist of rule bases, i.e., repositories of rules that are stored in .krb files (*Rule Bases*, n.d.). A single rule base may contain both forward-chaining and backward-chaining rules, providing flexibility in the rule execution process (*Rule Bases*, n.d.). The forward-chaining rules are run automatically when the rule base is activated, asserting new statements of fact, while backward-chaining rules are directly used to determine whether a particular statement is true (*Rule Bases*, n.d.).

Unveiling the Process of Forward-Chaining in Pyke

The method of reasoning used in inference engines, known as forward-chaining, checks if any known facts align with the if clause of a rule (*Forward Chaining*, n.d.). Pyke houses these facts in .kfb files referred to as fact bases (*Fact Bases*, n.d.). These facts are typically

straightforward statements, such as “Bruce is the son of Thomas” or “Ted has a heart rate of 60”, stripped down to their core logic (*Fact Bases*, n.d.). In the event of a match between a fact and a rule’s premise, the rule triggers, advancing to its ‘then’ clause. This consecutive chaining of rules is the rationale behind the term ‘forward-chaining’ (*Forward Chaining*, n.d.).

In Pyke, all forward-chaining rules are processed when a rule base is activated (*Rules*, n.d.). Forward-chaining rules may assert new facts, and activate more specific rule bases (*Rule Bases*, n.d.). Backward-chaining rules are processed when your program asks Pyke to prove a specific goal (*Rules*, n.d.). These rules are designed to answer a question rather than assert new facts or activate more specific rule bases. They also can assemble Python functions into a customized call graph or program, called a plan, to meet a specific need (*Rules*, n.d.). This paper solely utilizes the forward-chaining methodology in the described rule base, demonstrating its effectiveness in creating a rule-based expert system.

Illustrating the ES’s Inference Engine: An Overview of the Fact and Rule Bases

Figure 3 showcases the *healthcare.kfb* and *fc_healthcare.krb* files, which encompass the fact and rule bases—or the inference engine—of the ES, respectively. Notably, Pyke substitutes the terms ‘if’ and ‘then’ with ‘foreach’ and ‘assert’ in the rule base to symbolize forward-chaining rules.

Interpreting the Rule Syntax: A Case Study

To illustrate, consider a rule from our ES designed to monitor patient health. The rule, named high_BMI_high_BP_abnormal_HR, checks for patients who are obese, have high blood pressure, and an abnormal heart rate. If a patient meets these conditions, the system generates a message advising them to contact their doctor or visit the emergency department.

The rule is written using a specific syntax. The `**foreach**` keyword indicates that the rule will be applied to each instance that matches the conditions specified in the following lines. The `**healthcare.attributes_of**` function queries for the attributes of a patient, including their name, weight, height, systolic and diastolic blood pressure, and heart rate. The `**healthcare.obese**` and `**healthcare.high_BP**` functions check if the patient is obese and has high blood pressure, respectively. The `**check**` function checks if the patient's heart rate is either below 60 beats per minute or above 100 beats per minute. The `**assert**` keyword indicates that the system will assert (or make) new facts based on the conditions specified in the following lines. The `**healthcare.ab_hr**` function asserts that the patient has an abnormal heart rate, and the `**healthcare.message**` function asserts the message that should be communicated to the patient, along with their health attributes.

The specific structure and content of this rule are detailed in Figure 5. This rule demonstrates the power of Pyke in creating a rule-based ES. It seamlessly integrates Python statements within the ES rules, allowing for a versatile and expressive system that can handle complex logic and decision-making processes.

Analyzing the Rule Base: BMI Calculation and Classification

The initial rule presented at the top of Figure 5 conducts the computation for each patient's BMI—a calculation that integrates both the patient's height and weight data. Subsequent rules are deployed to categorize the BMI into segments: underweight, normal weight, or overweight.

Understanding the Fact Base: Vital Sign Entries

The facts that form the basis for rule processing are displayed in Figure 4. Every entry in the fact base signifies an individual's vital signs, with the initial parameter of the fact representing the patient's name.

Mapping Outcome States: The Complexity of Rule-based Systems

Figure 1 presents a matrix demonstrating potential outcome states resulting from the varying combinations of vital sign measurements. In total, there are 18 potential outcome states, which subsequently require a corresponding rule for each state, in addition to preliminary rules required to reach these outcomes. This illustrates how maintenance of classical ES rule-based systems can quickly become intricate and challenging.

Interpreting Alert Messages and Running the Program

The numerical values in Figure 1 correlate with distinct patient alert messages, as detailed in Figure 2. Figure 6 showcases the driver.py file, which serves as the entry point for the application. This file contains the essential programming logic that initiates the expert system and facilitates the interaction between the rules and facts. Following the execution of the driver.py file, the application applies the facts to the rules, as demonstrated in Figure 7. It is important to note that this program operates using Python version 2.6.

Executing the Program and Understanding the Output

To execute the program, the user must simply navigate to the directory of the driver.py file and run the execution command as shown in Figure 6: **`python driver.py`**. The program then outputs a specific, domain-relevant recommendation—namely, the patient instructions—and the patient's vital signs that informed the given decision, thereby enhancing the transparency of the ES.

Incorporating Inquiry Capabilities into the ES

As a recommendation for future development, the integration of a question rule base and back-chaining rules into the ES could significantly enhance its functionality. This would allow the system to not only analyze patients' vital signs but also probe further into their health status based on the data collected. By implementing back-chaining rules, the ES could ask targeted questions, enabling a more comprehensive and personalized analysis of each patient's health. This approach could potentially lead to more accurate alerts and recommendations, thereby improving patient care and outcomes.

Conclusion

This paper has demonstrated the transformative potential of rule-based expert systems in healthcare, particularly in enhancing patient self-care and improving the clinical management of heart failure. The development of the ES in Pyke, a Python-based knowledge-based system, has shown how the logical expressiveness of Prolog can be combined with the wide-ranging functionality of Python to create a versatile and powerful tool. The detailed analysis of a specific rule from the ES has further demonstrated the power of Pyke in handling complex logic and decision-making processes. Looking forward, the incorporation of inquiry capabilities into the ES could significantly enhance its functionality, enabling a more comprehensive and personalized analysis of each patient's health. This approach could potentially lead to more accurate alerts and recommendations, thereby improving patient care and outcomes. The study underscores the importance of continued research and development in this area to fully harness the potential of expert systems in healthcare.

References

Akanbi, A. K., & Masinde, M. (2018, August). Towards the development of a rule-based drought early warning expert systems using indigenous knowledge. In *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)* (pp. 1-8). IEEE.

Deebak, B. D., Al-Turjman, F., Aloqaily, M., & Alfandi, O. (2019). An authentic-based privacy preservation protocol for smart e-healthcare systems in IoT. *IEEE Access*, 7, 135632-135649.

Fact Bases. (n.d.). Retrieved July 10, 2021, from http://pyke.sourceforge.net/knowledge_bases/fact_bases.html

Forward Chaining. (n.d.). Retrieved July 10, 2021, from http://pyke.sourceforge.net/logic_programming/rules/forward_chaining.html

Godman, B., Malmström, R. E., Diogene, E., Gray, A., Jayathissa, S., Timoney, A., ... & Gustafsson, L. L. (2015). Are new models needed to optimize the utilization of new medicines to sustain healthcare systems?. *Expert review of clinical pharmacology*, 8(1), 77-94.

Hwang, G. (1995). Knowledge acquisition for fuzzy expert systems. *International Journal of Intelligent Systems*, 10(6), 525-546.

Jacobs, M., Pradier, M. F., McCoy Jr, T. H., Perlis, R. H., Doshi-Velez, F., & Gajos, K. Z. (2021). How machine-learning recommendations influence clinician treatment selections: the example of antidepressant selection. *Translational psychiatry, 11*(1), 108.

Morgenstern, M. (1987, December). Security and inference in multilevel database and knowledge-base systems. In *Proceedings of the 1987 ACM SIGMOD international conference on Management of data* (pp. 357-373).

Posadzki, P., Mastellos, N., Ryan, R., Gunn, L. H., Felix, L. M., Pappas, Y., ... & Car, J. (2016). Automated telephone communication systems for preventive healthcare and management of long-term conditions. *Cochrane database of systematic reviews*, (12).

Rule Bases. (n.d.). Retrieved July 10, 2021, from http://pyke.sourceforge.net/knowledge_bases/rule_bases.html

Rules. (n.d.). Retrieved July 10, 2021, from http://pyke.sourceforge.net/logic_programming/rules/index.html

Seto, E., Leonard, K. J., Cafazzo, J. A., Barnsley, J., Masino, C., & Ross, H. J. (2012). Developing healthcare rule-based expert systems: Case study of a heart failure telemonitoring system. *International Journal of Medical Informatics*, *81*(8), 556–565. https://doi.org/10.1016/j.ijmedinf.2012.03.001

Shanthanna, H., Strand, N. H., Provenzano, D. A., Lobo, C. A., Eldabe, S., Bhatia, A., ... & Narouze, S. (2020). Caring for patients with pain during the COVID-19 pandemic: consensus recommendations from an international expert panel. *Anaesthesia*, *75*(7), 935-944.

Sharda, R., Delen, D., & Turban, E. (2020). *Analytics, data science, & artificial intelligence* (Eleventh edition). Pearson.

Tecuci, G. D. (1992). Automating knowledge acquisition as extending, updating, and improving a knowledge base. *IEEE transactions on Systems, Man, and Cybernetics*, *22*(6), 1444-1460.

Welcome to Pyke. (n.d.). Retrieved July 10, 2021, from http://pyke.sourceforge.net/