Università degli Studi di Ferrara

Università degli Studi di Ferrara

Corso di Laurea in Informatica

A First Experiment with an Intelligent System for Therapy Recommendation for Osteoporotic Patients

Relatore:
Prof. Guido Sciavicco

Laureando:
Maxim Nitsenko

Anno Accademico 2020 – 2021

Contents

Riassunto

Si descrivono le specifiche pratiche e teoriche di un sistema intelligente per l'aiuto alla decisione sviluppato per il centro di menopausa e osteoporosi di Ferrara.
Il sistema si serve di un metodo di apprendimento automatico che identifica un insieme di regole di classificazione. L'aspetto che caratezza il sistema è la rappresentazione simbolica delle regole che favorisce l'interpretabilità, una caratteristica fondamentale in informatica medica.
Nella prima parte si offre un'ampia panoramica dei sistemi di aiuto alla decisione nel settore medico, in seguito si discute la struttura e le scelte implementative del sistema.

Chapter 1

Introduction

The CMO (Center of Menopause and Osteoporosis) is a conjoined clinic with the University of Ferrara specializing in the diagnosis and treatment of osteoporosis and related disorders. Osteoporosis is a kind of a skeletal system disease mainly categorized by alterations in bone mass density and its structure that makes the bone prone to fracture.

To operate effectively in the current IT landscape, health organizations must include digital technologies in their clinical workflow. Accordingly, in the year 2017, the CMO clinic, alongside the department of informatics, embraces the shift to digitalization, replacing paper for electronic health records (EHR). The process of creating, retrieving, and storing medical records was considerably simplified, making healthcare services more efficient, improving the quality of service, and reducing costs in terms of paper usage and time. Today the CMO clinic collects a significant volume of patient data in the form of EHR. Unfortunately, all this digital information remains untapped. This situation is common to the healthcare environment, which is frequently information-rich yet knowledge poor. As John Naisbitt, a well-known futurist, once said, "we are drowning in information, but starving for knowledge." However, by means of machine learning techniques, knowledge and meaningful patterns may be extracted from biomedical data.

This thesis discusses the implementation and theoretical aspects of a self-learning clinical decision support system. The proposed system applies machine learning techniques to aid diagnosis by predicting the appropriate therapy and its active ingredient.

The remainder of this thesis is organized as follows: Chapter 2 presents a literature review of different classes of medical decision support, as well as several implementation examples. Chapter 3 outlines the internal functioning of the adopted machine learning algorithm. Chapter 4 focuses on dataset preprocessing. The system's infrastructure is presented in chapter 5. The sixth chapter is dedicated to classification rules, a central figure in the system. Chapter 7 is a brief discussion on model evaluation. The final chapter completes the thesis with conclusions regarding the software application and my internship.

Chapter 2

Background

2.1 Definitions

The concept of a decision support system (DSS) is broad, and its definitions vary depending on the author's point of view. it can take many different forms. From the simplest one "a computer-based system to aid in decision making" [23] to a more elaborate definition as "an interactive system that provides ultimate users who make decisions and easy and convenient access to data and models for decision making in semi-structured and unstructured situations from different fields of human activity" [20]. This variety of definitions is caused by a wide range of different shapes, sizes, and types of DSS.

Decision support systems find extensive application in medicine and healthcare. Therefore require their very own definition. A clinical decision support system (CDSS) is a health information technology system that is designed to provide physicians and other health professionals with knowledge and person-specific information, to enhance, assistance with clinical decision-making tasks in one or more component steps of the diagnostic process [24].

The intention of CDS is to support, rather than to replace, the clinician. The CDS may offer suggestions, but the clinician must filter the information, review

the suggestions, and decide whether to take action or what action to take. With the increased focus on the prevention of medical errors that has occurred since the publication of the landmark institute of Medicine report, "To Err is Human" [17], computer-based physician order entry systems, coupled with CDSSs, have been proposed as a key element of systems' approaches to improving patient safety. There are already several research studies suggesting that AI can perform as well as or better than humans at key healthcare tasks, such as diagnosing disease [10]. Also, in addition to a better understanding of clinical situations and improved decision making, CDSSs contribute to saving time, expenses, or resources.

2.2 DSS Taxonomy

Clinical decision support systems are widely categorized into two major groups, namely knowledge-based CDSS and non-knowledge based CDSS. The knowledge-based system consists of an attempt to model human knowledge in computational terms, often, but not always, in the form of if-then statement [5]. The non-knowledge based systems employ machine learning and other statistical approaches that allow the computer to learn from past experiences and recognize patterns in the clinical data. Knowledge based systems combine knowledge from human experts, from medical literature or both; its acquisition and formalization is thus a "bottleneck," which consumes development time and requires a significant initial effort. On the other hand, data-driven methodologies are currently receiving much attention, thanks to a large amount of data available in electronic form, to the availability of powerful computing architectures, and to the significant advancement of machine learning techniques that can extract characteristic features and to identify patterns from data with a high level of accuracy. Data-driven ML is most useful when a priori knowledge is limited or nonexistent. The resulting functions and weights are often not easily interpretable (black-box systems), and the system cannot explain or justify why it uses certain data the way it does, which can make the reliability and accountability of these systems a concern. Sometimes, however, hybrid approaches are preferred since they can take advantage of both formalized knowledge and available data [19].

2.2.1 Knowledge Based CDSS

Most of the knowledge-based systems have three parts: an inference engine, a mechanism to communicate, and a knowledge base.The inference engine synthesizes existing rules of knowledge base with the patient's data. The communication mechanism enables systems to show the results to operators. The knowledgebase explicitly represents knowledge as symbols, most commonly, in the form of rules, ontologies, past cases, or other types of knowledge structures [22].

Rules: Symbolic representation in the form of if-then rules constitutes the most common knowledge representation scheme. Any rule consists of two parts: the IF part, the antecedent, and the THEN part called the consequent. The initial state of the system is represented as a set of facts or assertions in working memory. The inference engine performs a pattern match of the antecedents of the rule against the set of assertions in the working memory. When there is a match, the rule is tagged for possible execution, building a set of rule activations. Because more than one rule may match the current state in working memory, some mechanism for resolving conflicts is usually provided. The selected rule is then executed or "fired", resulting in some action being performed or one or more facts being asserted to (added) or retracted (removed) from the working memory [12].

Management of uncertainty is an intrinsically important issue in designing a rule-based system because much of the information in the knowledge base is often imprecise, incomplete, or not completely reliable. A rule-based system often uses fuzzy logic instead of boolean logic, and it is known as a fuzzy rule-based system. A fuzzy rule-based system is a collection of fuzzy rules and membership functions that are used to reason about data. Fuzzy logic is a set of mathematical principles for knowledge representation based on degrees of membership rather than the crisp membership of classical binary logic. Unlike two-valued Boolean logic, fuzzy logic is multi-valued. Fuzzy rules express expert knowledge in vague and ambiguous terms. This kind of logic attempts to model a human's sense of words, decision making, and common sense, leading to a more human intelligent machine [13]. In [18] the authors discuss a fuzzy rule-based decision support system for automatic drug delivery in patients undergoing general anesthesia. The system performed

satisfactorily and outperformed the manual administration in patients in terms of accuracy through the maintenance stage.

Ontologies: In recent years, the use of ontology as a mechanism for representing knowledge in CDSSs has gained momentum in supporting and solving decision problems. An ontology is a formal description of knowledge as a set of concepts within a domain and the relationships that hold between them, generally a set of "part-of" or "is-a" relations. Ontologies are designed in a way that allows knowledge inference and reasoning. One of the main features of ontologies is that, by having the essential relationships between concepts built into them, they enable automated reasoning about data. Another valuable feature is that ontologies are easy to extend. Ontologies also provide the means to represent any data formats, including unstructured, semi-structured, or structured data [1],[16],[11]. In [7] the authors propose a successful implementation of an ontology-based model for diagnosis and treatment of diabetes (OMDP). From the experimental results, OMDP not only offers better performance for predicting and diagnosing diabetes in individuals but also has better accuracy for recommending useful treatment to patients.

Case-based reasoning: CBR's goal is to identify the case that best reflects the current situation and adapt it to solve a new problem. One widely adopted CBR model is the R4 model. The R4 model consists of a four-stage cycle: retrieve, reuse, revise, and retain. First, a new problem is presented to the system consisting of features and feature-values, then a similar case is retrieved. The retrieved case is then reused to solve the new problem; this may involve some form of adaptation to resolve any discrepancies between the proposed problem and the retrieved case. A solution is then presented and revised by the user or system. If the proposed solution is accepted, it is then retained in the case-base as a new case. This cycle then repeats to enable the system to continuously improve suggestions as its knowledge (case-base) grows. In the medical field, the use of the CBR approach is very appealing because the core of the reasoning process shows a clear similarity to clinical reasoning. indeed, the physician often tries to make the connection between the case and those already experienced in his practice, and it is precisely the principle of this method [6].

2.2.2 Non-Knowledge Based CDSS

Unlike knowledge-based decision support systems, nonknowledge-based CDSSs use a form of artificial intelligence called machine learning. Taking advantage of the accumulated clinical data stored, e.g., in electronic patient records, powerful data-driven clinical decision support systems can be implemented, possibly leading to more effective outcomes in practice [19]. Despite these significant results, the existing methods cannot describe how a learning result is reached and why a prediction decision is made. The explanation of how a decision made is vital for the acceptance of machine learning technology in bioinformatics [15]. Luckily there exists a class of ML algorithms with these desirable properties called rule-based machine learning.

Non-interpretable Machine Learning

While there are several reasons why the interpretability of ML models is essential, not all prediction problems in supervised machine learning predictions require explanations. Alternatives to explanations include domains where the system may have theoretical guarantees always to work or empirical guarantees of performance when the system has historically shown to have high performance, e.g., deep learning applications radiology with superhuman precision, or in work pioneered by Gulshan et al., the developed deep learning algorithm was able to detect diabetic retinopathy from retinal fundal photographs with extremely high sensitivity and specificity. The exceptional performance supports the fact that this prediction does not require an explanation. However, findings such as this are quite rare. Also, many healthcare problems are complex, and simplifying them to point solutions with accompanying explanations may result in suboptimal outcomes [3]. In regards to the adopted methodologies, conventional approaches include Support Vector Machines (SVM), Decision Trees, Genetic Algorithms, Bayesian models, and Neural Networks (NN). NNs deserve special consideration since they are the key technique adopted in many DSSs. Indeed, image interpretation and classification are fields where NN/deep learning approaches excel [19].

Rule Based Machine Learning

Due to the if-then semantic structure resembling natural language, decision rules are the most interpretable prediction models. Rule-based machine learning applies some form of learning algorithm to automatically identify classification rules, rather than requiring prior expert domain knowledge to build and curate a rule set manually. Their explicit nature makes them particularly attractive for decision support. Rule-based classification systems have been widely applied in medical diagnosis and are competitive with other classification algorithms.
There are many machine learning algorithms and data mining techniques to learn rules automatically [2]. They can be divided into three categories: rule induction, association rule mining, rule extraction via other classification models.

Rule induction: Rule induction algorithms adopt the sequential covering strategy, whose basic idea is to learn a list of rules from the training data sequentially (one by one). Once a new rule has been learned, the corresponding training examples that it covers are removed. This learning process is repeated until rules can cover the whole training data, or no new rule can be learned from remaining training data. Some of the well-known algorithms are CN2, RIPPER, and FOIL [2].The system discussed in this theses is located precisely at this classification level, and the details of the underlying algorithm will be expounded upon later in chapter three.
Association rule mining: Association rule mining is a well-known technique in data mining and has been widely adopted in applications such as heart disease prediction, healthcare auditing, and neurological diagnosis. The basic idea is to find a strong correlation or association between the frequent itemsets and class labels based on association rule mining techniques. Classification based on association rule mining detects all rules that satisfy the user-specified minimum support and minimum confidence constraints [2]. Different from greedy algorithms, association-rule mining searches globally for all rules that associate class labels with combined attributes; thus, it can achieve global optimality. Study results show that this approach can achieve lower error rates than greedy algorithms [26]. Apriori is the most popular algorithm in association rule mining.

Rule extraction via other classification models: One commonly used rule extraction model is a decision tree. A decision tree is a simple tree-like recursive structure for expressing a sequential classification process in which a case, described by a set of attributes, is assigned to one of a disjoint set of classes. Each leaf of the tree denotes a class. An interior node denotes a test on one or more of the attributes with a subsidiary decision tree for each possible outcome of the test [21]. Rules can be easily derived from decision trees by generating one rule for each path from the tree's root to its leaf.
The strength of ANNs or SVMs techniques lies with their ability to model non-linearities, resulting in incomprehensible complex mathematical models. However, there are techniques for extracting interpretable rules from these models as well, keeping much of the original accuracy [14].

It is worth mentioning this recent paper [27], in which, the authors propose a new clinical diagnosis and treatment system in which rules are generated by a voted ensemble of state-of-art decision-tree-based classification algorithms. The multi-classification model simultaneously integrates multiple classifier algorithms and votes for the best performance as the classification results. Each rule is followed by statistical output (e.g., accuracy rate). To fulfill the reliability of classification, decision trees are updated continuously. Instead of re-constructing a classification model each time new instances arrive, the classification algorithms support incremental learning and, thus, reduce the cost in terms of the time and error rate.

In [8], the authors researched and developed a real-time clinical decision support system for an intensive care unit (ICU) based on associate rule mining to assist clinicians in making optimal decisions, decreasing medical errors, and minimizing life-threatening events caused by delayed or uninformed medical decisions. The graphical user interface enables the user to input real-time patient clinical scenarios and to extract confident association rules from the database. The authors adopted the "support" and "confidence" metrics suitable for ICU clinical application from conventional association rule mining. in addition, they defined and developed two new rule-wise metrics, "importance" and "dominance" and one item-wise metric "effect."

2.2.3 Hybrid CDSS

The advantage of knowledge-based CDSS lies in the fact that it can directly embody human experts' knowledge. However, this is also where the problem with knowledge-based systems lies. The experts cannot always fully translate their knowledge and experience into a set of rules. Such a system may appear limited or unreliable. Besides, knowledge-based systems are rigid; that is, they do not learn. As for data mining based CDSS, the main advantage is that no expensive expert knowledge acquisition step is required. Additionally, such systems learn with time and are incredibly flexible. There are disadvantages, however. In addition to being mostly black-box models, machine learning algorithms learn the general pattern in the data and may fail to recognize rare or unusual cases due to predictions being limited to the cases that have been observed [4]. This may pose a concern in delicate fields like medicine. It is straightforward to see that hybrid approaches sometimes may be preferred. By combining knowledge-based systems and data-driven machine learning algorithms, they're able to take advantage of both formalized knowledge and available data.
In [4], the author proposes a web-based hybrid CDSS that incorporates elements of both approaches. This strategy strengthens the benefits and limits the deficiencies of both algorithms. The system has a knowledge base acquired from experts in combination with a learning module that will take a database of records, learn from the data, and augment the existing knowledge base via refinement or addition. The learning module employs FOLC, a sequential covering algorithm that learns a set of first-order Horn clauses to cover the observed training examples. The knowledge base management module handles all the modifications to the knowledge base. Doctors through an interface can augment the knowledgebase adding new rules. An adaptation of the FOLC algorithm, called ACM-FOLC (Attribute Cost Minimized FOCL), is proposed to minimize the diagnostic cost. While some expensive tests can ensure the highest accuracy, it is possible in many cases to correctly determine diseases with less expensive tests. AMC-FOCL identifies which tests are absolutely required.

2.3 CMO

The following section outlines a brief history of the CMO's information system and its components.

Except for visit and bone density scan details recorded on the clinic's densitometers, the clinic had been relying on paper for medical record-keeping. The densitometers run a proprietary software that stores the patient data on a Microsoft Access database format. This configuration turned out to be quite restrictive as it could not support appropriate data access to EHRs for elaboration or statistical analysis. For this reason, the entire software and physical structure had to be reconsidered.

From the moment that the partial information stored in the database Access was not enough to manage a comprehensive medical reporting, the new database scheme had to complement it with additional tables for patient history and diagnosis. Periodically opportune software modules copy the latest data from the densitometers to the main database.

One challenging aspect was to pair every scan image to the corresponding patient visit in the main database. The problem was that densitometers do not have a saving option for scans; they can only print them. This issue was resolved by digital prints to a pdf file with a third party software and storing those in the server. The corresponding visit is then identified by digitally reading patient details from the pdf itself.

Report management happens via a web-based application. The web app being pivotal to the medical practice was tailored to the clinic's specific needs and requirements. The application lets authorized users, instantly and securely, consult clinical profiles of patients under their care, not only from the office but also from home.The clinician has a comprehensive view of the patient state; local to the website, all the densitometric measurements, images, and laboratory test analyses are offered to the user for an accurate and reliable report formulation. The app also reduces the movements of physical documentation, accelerating timelines of medical procedures.

There are four stages to a complete report formulation: body scan signature, anamnesis, diagnosis, full report generation, and its signature. The sequence is carried out under a finite state automaton, which enforces the right order of execution.

The last major feature developed for the web app is an interrogation module. The module is a simplified querying mechanism on the health record set for research or study purposes. No relational data language knowledge is necessary as the set of all possible queries the medic may be interested in can be expressed in the web app through the application of (yes, no, yes or no) conditions to the health record's variables. The app supports saving, copying, or storing queries for later execution.

Chapter 3

JRip

The learning algorithm is JRip, a java implementation of the RIPPERk algorithm originally proposed by Cohen [9],[25]. JRip is a propositional rule learning algorithm that performs efficiently on large, noisy datasets, scales nearly linearly with the number of training examples, and is competitive with analogous algorithms in the same class. Similar to a standard separate and conquer algorithm, JRip builds a rule set in a greedy fashion one rule at a time. After a rule is found, all examples covered by the rule, (both positive and negative,) are deleted. This process repeats until some stopping condition is satisfied. After the initial ruleset is obtainded, it's then optimized.

Building stage
In order to build a rule, JRip uses the following strategy. First, the examples which have not yet been covered by any rule are randomly partitioned into two subsets a growing and a pruning set. Next, using only the growing set a rule is "grown" by greedily adding conditions until the rule is 100% accurate (ei the rule covers no negative examples) by testing every possible value of each attribute and selecting the condition with greatest information gain. The above procedure most certainly produces rules that overfit the growing set. This is solved by immedi-

ately pruning the rule so as to maximize its performance on the pruning data. The pruning considers deleting any final sequence of conditions from the rule and chooses the deletion that maximizes some function w. This process is repeated until no deletion improves the value of w. Lastly all the positive and negative examples covered by the rule are removed, (and the process restarts). JRip stops adding rules when there are no more positive examples left or when a rule found by JRip has an unacceptably large error rate or when the last rule added is too complicated (the description length is more than 64 bits larger than the smallest description length encountered so far).

Optimization stage

The ruleset R produced by the learning algorithm outlined so far is taken as a starting point for a subsequent optimization process. This process re-examines the rules $r_i \in R$ in the order in which they were learned. For each rule r_i, two alternative rules are constructed, the replacement r_p and the revision r_v. The replacement rule is formed by growing and then pruning a rule r_i from the ground up. The revision rule is formed similarly, except that the revision is grown by greedily adding conditions to r_i, rather than the empty rule. To decide which version between r_v and r_p to retain, the Minimum Description Length criterion is used. Lastly, rules are added to cover any remaining positive examples using the building stage. The optimization stage can be reiterated again k times. The rule set is then simplied by examining each rule in turn (starting with the last rule added) and deleting rules so as to reduce total description length.

The algorithm described is for two-class learning problems, however it can be easily extended for multiple classes as well.

Chapter 4

Data Preprocessing

Data preprocessing is an essential step in Machine Learning as the quality of data directly affects the ability of the ML model to learn.
The first step was to examine the raw data and decide which columns are useful and which are not. Many columns (e.g., patient name, visit date, etc.) have been discarded as they do not carry any valuable information for the classification task. Because of improper data acquisition from a web module, many fields containing listings were ill-formatted; therefore, inconsistent and unnecessary item delimiters have been removed. The "last menstruation year" and "visit date" fields are not useful individually, but subtracting one from another creates a new feature: the total number of years in menopause, which is valuable for classification. Some particular characters have been replaced with words for easier text pattern matching. For some categorical variables, empty fields were replaced with words. Different null encodings have been homogenized. The same name columns were renamed.
The preprocessing step often demands the treatment of null values. Apart from uniforming different null encodings, no other work was necessary since JRip can easily deal with missing data.
Handling categorical variables is another integral aspect of Machine Learning. JRip and many other machine learning algorithms cannot be executed on categor-

ical data. For categorical variables where no ordinal relationship exists, a one-hot encoding scheme is applied. One-hot-encoding works as follows: a new column is created for every unique value in a categorical variable. The column contains a one if the variable originally had the corresponding value; otherwise, it contains a zero. The original dataset, being medical in nature, contained many relevant text columns. Unfortunately, machine learning algorithms will not work with raw text directly. So, as for categorical attributes, numerical vectors need to be derived from textual data to reflect various linguistic properties of the text. This process is called vectorization. Plenty of time was allocated to text column vectorization. The first strategy tried was to couple the bag-of-n-grams model with the tf-idf metric. The bag-of-ngrams model is a representation that turns arbitrary text into fixed-length vectors by counting how many times each n-gram appears. An n-gram is a contiguous sequence of n items from a given sample of text. N-grams can be more informative than single words because they capture more context around each word (i.e., "allergic to ASA" is more informative than just "allergic"). The tf-idf metric is more advanced as it considers words relevance based on their rarety across all documents, not only their frequency. So instead of raw counts, each n-gram was assigned a tf-idf score. The bag-of-n-grams plus tf-idf approach yield unsatisfactory results along with a massive increase in complexity. The final approach was to create a dictionary with manually selected n-grams based on perceived importance and frequency.
In any case, for improved performance, the text needs to be cleaned beforehand, standard techniques for text preprocessing have been applied.

Normalization: Text normalization is the process of transforming a text into a canonical form.

- Medical abbreviations have been expanded (e.g., vit $\rightarrow$ vitamin).
- Same-meaning words have been standardized (e.g., ASA $\rightarrow$ aspirin, acetylsalicylic acid $\rightarrow$ aspirin).
- Corrected misspellings for critical words.
- All words have been lowercased.

- Removed punctuation, frequent unrelated words, irrelevant dates or numbers, etc.

Stop words removal: Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are”, etc. All stop words are filtered out in this step since they contribute little to the overall meaning. And it also helps with reducing the number of features.
Stemming: It is a process of transforming a word to its root form. Stemming, for instance, will reduce both "treatment" and "treated" to the word "treat." This is done to reduce dimensionality and because there are no sufficient training samples [for the classifier] to discern such fine differences anyway.
Fortunately, the Python ecosystem has many natural language processing libraries. For this project, the NLTK library was used.

Chapter 5

System Architecture

Based on various factors, such as age, disease progression, family history, etc., the osteoporotic patient is prescribed one or more of the following therapies: hormonal, osteoprotective, or vitamin d and one, both or none of these supplementations: calcium or vitamin d. For each treatment, the intelligent system provides a yes or no suggestion. Considering that the treatments are not mutually exclusive, i.e., multiple treatments may be assigned to each problem instance; it is a case of a multilabel classification problem. There are multiple approaches to multilabel classification. We favored the simplest one, that is, training one binary classifier for each label independently. This scheme will require five separate JRip binary classifiers.

After deciding on the treatment type, the physician chooses its active ingredient, dosing, frequency, and timing. For each therapy or supplementation, there can be many combinations leading to numerous classes. So we restrict ourselves only to predict the active ingredient and sometimes the dosages as there is not enough data to support a more detailed classification. For instance, calcium supplementation can have four distinct formulations:

Calcium citrate 500mg once a day

Calcium citrate 500mg twice a day
Calcium carbonate 600mg once a day
Calcium carbonate 600mg twice a day

This four-class classification problem will reduce to a binary one with only calcium citrate and calcium carbonate as classes. So if we also wish to predict the active ingredient, this will require an additional five multiclass classifiers. Likely JRip supports multiclass classification, thus no need for one-vs-all or one-vs-rest like strategies. Five classifiers for treatment and supplementation, plus five for their active ingredients, in total JRip, runs ten times, and each time it records decision rules for each target variable.
Based on the previous discussion on interpretable machine learning in medical settings, an algorithm with such characteristics is highly desired. We favored WEKA, an open-source machine learning suite containing algorithms for data mining tasks. Weka includes two rule-based classifiers: PART and JRip. After some testing, we decided upon JRip due to shorter rules and overall higher accuracy. The reason for adopting two programming languages is that Weka's API is in Java, and Python has excellent data preprocessing libraries. Python's pandas library was used extensively for this project. Pandas is a versatile open-source library with many tools and features for data analysis like the DataFrame object.

5.1 System Workflow

This chapter describes the system's general "life cycle" and the software modules it consists of. The system's data flow between different components can be logically divided into two functionally distinct cycles: the *rule-base-update* cycle and the *fetch-and-show-suggestion* cycle. Briefly, the first cycle's function is to update the rule base, and the second's is to query the knowledge base and show a suggestion.

5.1.1 rule-base-update cycle (1)-(6)

On weekends, when the clinic is not operational, an automatic job scheduler (cron utility) will run fase1.py data preprocessing module (1). When this script executes, a SQL query fetches the required data from the MySQL server and places it in a

Python DataFrame object. Then all the records are preprocessed, as described in chapter 6. Fase1.py terminates after outputting perJava.csv. This text file consists of all the preprocessed records, ready for the classifier. Afterward, the cron utility calls fase2.java (4). This module's function is to update the knowledge-base. It reads perJava.csv and trains a distinct classifier for each class we care to predict. After training, the JRip data object returns a string containing the classification rules. This ruleset is parsed with regular expressions and then moved into internal data structures, where it is refined. Lastly, fase2.java commits the decision rules to the server. The knowledge base is now created, and the cycle terminates, until next week. There is no need to update the rule base more frequently since a small number of new instances will not likely alter the rule base.

5.1.2 fetch-and-show-suggestion cycle (7)-(14)

The classification rules await in the database until the physician opens the diagnosis section through the web app. As soon as the referti2.php loads, an asynchronous POST request to the gethint.php script with the patient's private key is made (8). Since the server takes a couple of seconds to respond, the system for better user experience preloads all the therapy recommendations for the patient under consideration. Thus the request sends out immediately and not when the recommendation button is clicked. Gethint.php acts as an interface between the web app and the inference engine. Its few code lines execute fase3.py via shell with the patient's private key as argument (9) and deliver its response back to referti2.php (12). In fase3.py, the decision rules generated at step (6) are finally recovered together with the patient's database record (10). Then, given a class to predict, the inference engine checks the patient's data against every rule until it finds a match. Finally, the script prints to the gethint.php's standard input the class prediction, the motivation, and the rule characterization (11). This last step repeats ten times for each class to predict. If a suggestion is not available, i.e., irrelevant and unreliable, then the "idk" character sequence prints. When fase3.py terminates, gethint.php delivers these results to referti2.php (11), where they are temporally stored until the physician clicks on a therapy recommendation button. In this case, a new window (show_suggestion.php) pops out (14) with all the recommendations for that particular treatment.

This cycle executes each time the physician opens the diagnosis web page, requests a suggestion or a patient field is modified.

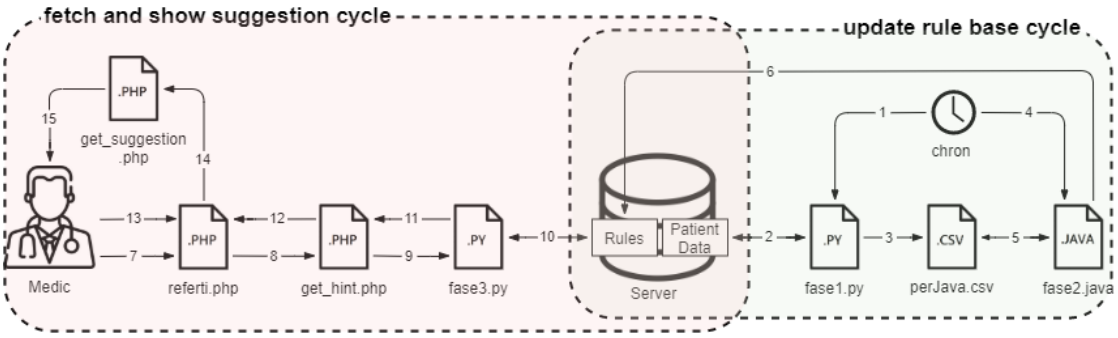

Figure 5.1: The system's lifecycle.

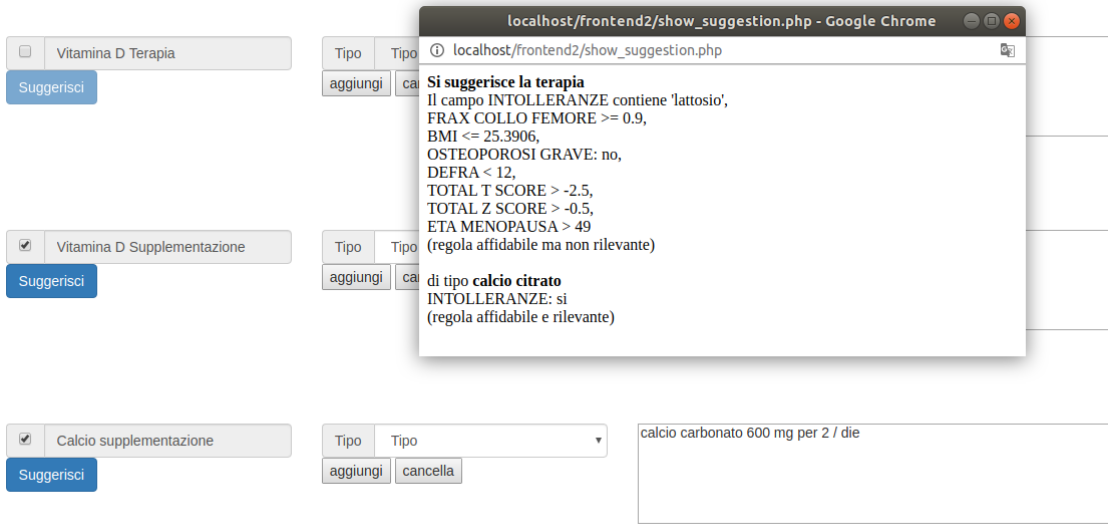

Figure 5.2: Clicking on the blue "Suggest" button displays a new window with a recommendation.

Chapter 6

Classification Rules

JRip learns a ruleset $R = \{r_1, r_2, \ldots, r_d\}$, a collection of classification rules that collectively form a classifier.

Each rule is expressed in the following way $r_i : (c_i) \rightarrow y_i$
$c_i \doteq P_{i,1} \wedge \ldots \wedge P_{i,n_i}$ is the rule condition
$P_{i,j} \doteq A_{i,j}\ op\ v_{i,j}$ is the jth proposition of the ith condition
$A_{i,j}$ is the attribute of the i,jth proposition
$v_{i,j}$ is a value that the attribute $A_{i,j}$ can take
n_i is the number of propositions in the ith condition
$op \in \{=, \leq, \geq\}$
y_i is called the rule consequent, and it holds the predicted class

The ruleset R is called a decision list. In contrast to an order-independent rule set, decision lists have an inherent execution order. Unordered rules suffer from the drawback that a test example may get multiple classifications; that is, it may satisfy rules that apply to different classes. Decision lists solve the problem of overlapping rules because they are interpreted in order, and execution ends as soon as one rule applies [25].

```
(SITUAZ_COLONNA = Osteoporosi) and (OSTEOPOROSI_GRAVE = 1) => CALCIO_SUPP = 1 (132.0/31.0)
(SITUAZ_COLONNA = Osteoporosi) and (TERAPIA_ALTRO = 1) => CALCIO_SUPP = 1 (245.0/116.0)
(SITUAZ_FEMORE_SN = Osteoporosi) and (TERAPIA_ALTRO = 1) and (FRATTURA_FAM = 0) => CALCIO_SUPP = 1 (73.0/23.0)
(OSTEOPOROSI_GRAVE = 1) => CALCIO_SUPP = 1 (63.0/18.0)
(SITUAZ_COLONNA = Osteoporosi) and (INTOLLERANZE_0 = 1) => CALCIO_SUPP = 1 (10.0/0.0)
 => CALCIO_SUPP = 0 (1296.0/156.0)
```

Figure 6.1: The output string of JRip for calcium supplementation.

The last rule $r_d : () \rightarrow y_d$ is called the default rule, which applies only to instances not covered by any of the preceding rules. The default rule makes the decision list exhaustive; that is, it ensures that there is always a prediction.

6.1 Rule Revision

An instance x is classified by trying each rule in sequence until some rule precondition matches the attributes of x; if so, the class is prediction is returned. This can be schematized as follows:

if (c_1)
    return y_1
else if (c_2)
    return y_2
$\vdots$
else
    return y_d

The intelligent system, in addition to predicting the class, also has to justify the choice. The choice being the selected rule. So, when a rule applies to some test instance, the class prediction, as well as the rule are returned. However, the rules are intended to be interpreted in sequence, whereas taken individually, out of context, may be incorrect [25]. To make each rule context-independent the condition part of each rule is modifed by incorporating the negated conditions of all previous rules:

if (c_1)
    return y_1
else if $(c_2 \wedge \neg c_1)$
    return y_2

else if($c_3 \wedge \neg c_1 \wedge \neg c_2$)
    return y_3
$\vdots$
else if($\neg c_1 \wedge \ldots \wedge \neg c_{d-1}$)
    return y_d

6.2 Rule Memorization

This section will explain the class structuring for rule memorization.
The base class is *Proposition*, it holds a generic proposition $P_{i,j}$. The *Rule* class models each rule r_i, and it contains a list of *Proposition* elements, the rule metrics (m,n), and the prediction y_i. The *Rules* class is a collection of *Rule* items, representing the entire classifier.
This hierarchal structure represents exactly the rules generated by the JRip classifier, and it is used primarily for testing. The rule revision stage requires expanding the class architecture.
A revised rule will not have one implicit condition, but generally many, as it inherits them from the preceding rules. The *Condition* class will be an interlayer between *Proposition* and *RRule*, and it is introduced to model conditions inside a rule. The *Condition* class is a collection of *Proposition* elements and is implemented as an abstract class with *ConjunctiveCondition* and *DisjunctiveCondition* as subclasses (c_i and $\neg c_i$ conditions, respectively). The *RRule* class is a list of clauses, and it represents a generic revised rule. Finally, *RRules* class will describe the revised ruleset.
As for the persistent storage in the database, there exists a single independent table *Regole* with two columns, one for the treatment kind and one for the corresponding rules.

6.3 Rule Refinement

JRip provides, for each classification rule, an accuracy metric (m, n). The integer m signifies the number of training examples covered by the rule and n how many of them were correctly classified. In other words, m indicates rule relevance and

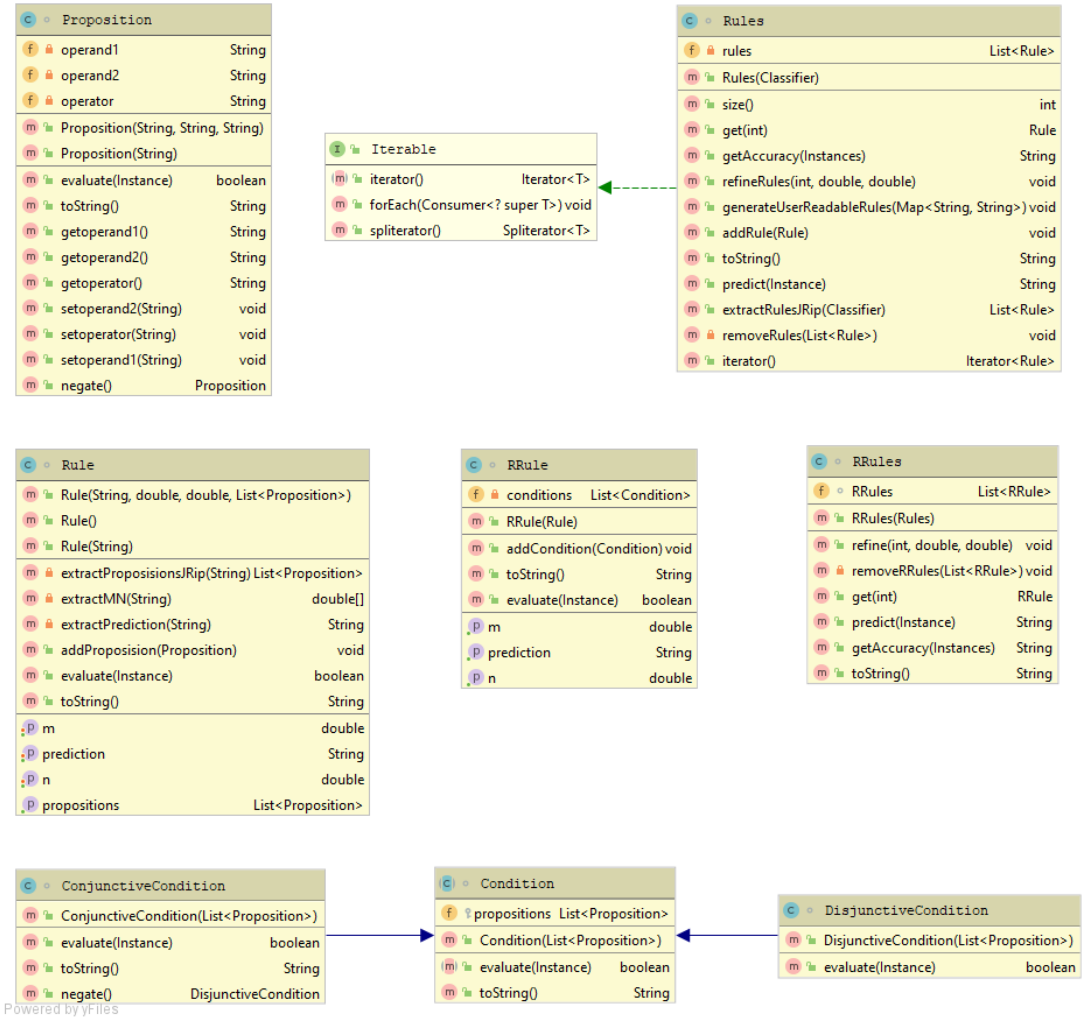

Figure 6.2: UML class diagram.

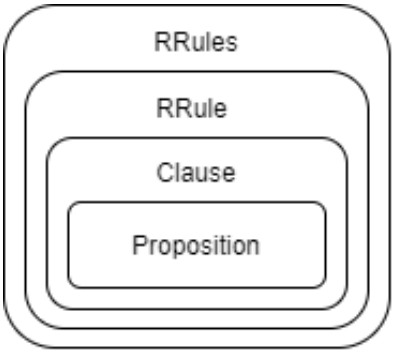

Figure 6.3: Class hierarchy.

n, its reliability.
We require only to keep meaningful rules, hence irrelevant and not reliable rules are discarded. The reliability and relevance metrics can be useful for the physician in the process of decision making. So, each therapy suggestion also details the adopted rule as "reliable and relevant" or "reliable but not relevant" or "not reliable, but relevant."

6.4 Rule Presentation

For legibility reasons, before presenting a rule to the user, it is subject to further processing.
A generic rule r_i has $i-1$ negated conditions plus the original condition. Each negated rule condition is a disjunction of propositions:

$$r_i : ((A_{i,1}\ op\ v_{i,1} \wedge \ldots \wedge A_{i,n_i}\ op\ v_{1,n_i}) \wedge (A_{1,1}\ \neg op\ v_{1,1} \vee \ldots \vee A_{1,n_1}\ \neg op\ v_{1,n_1}) \wedge \ldots \wedge (A_{i-1,1}\ \neg op\ v_{i-1,1} \vee \ldots \vee A_{i-1,n_{i-1}}\ \neg op\ v_{i-1,n_{i-1}})) \rightarrow y_i$$

Evidently, this intricate structure compromises readability. It is, for this reason, the rule r_i is simplified by replacing each negated condition with any true proposition from that condition. The result is, again, a conjunction of propositions, which is considerably more comprehensible:

$$r_i : ((A_{i,1}\ op\ v_{i,1} \wedge \ldots \wedge A_{i,n_i}\ op\ v_{1,n_i}) \wedge (A_{1,T}\ \neg op\ v_{1,T}) \wedge \ldots \wedge (A_{i-1,T}\ \neg op\ v_{i-1,T})) \rightarrow y_i$$

where $(A_{j,T}\ \neg op\ v_{j,T})$ is some true proposition from $\neg c_j$

Another instance of poor readability occurs when a rule contains repeated or equivalent propositions. For example:
$r_i : (BMI < 30 \wedge OSTEOPOROSI\_GRAVE = 0 \wedge BMI < 25 \wedge OSTEOPOROSI\_GRAVE = 0) \rightarrow y_i$
will be simplified to $r_i : (BMI < 25 \wedge OSTEOPOROSI\_GRAVE = 0) \rightarrow y_i$

For a categorical variable A, the one-hot encoding scheme will create one new feature for each value the variable A can take (A_1, A_2,...). Therefore, propositions containing attributes generated by this procedure are meaningless and need renaming. For example, the proposition $ALLERGIA\_3 = 1$, when displayed, changes to "il campo ALLERGIE contiene 'fans'"

There are several other minor purely cosmetic enhancements, which are not reported.

Chapter 7

Model Evaluation

When a machine learning model learns the noise or random fluctuations in the training data, it's called overfitting. If a model overfits, it loses its ability to generalize to new data. So it's standard practice in machine learning to split the data into two mutually exclusive sets, the training set and the testing set (the holdout method). The model trains on the training data, then its quality is evaluated on new unseen data in the testing dataset. A slightly different method, which gives a more reliable accuracy estimate, was used for JRip and PART evaluation. The data was divided into ten subsets, such that each subset contains about the same sample proportion of each target class as the initial set, and then the holdout method is repeated ten times. Each time, it uses one of the ten subsets as the test set and, the other nine subsets are combined into a training set. Then the average error across all ten trials is computed. This approach is called stratified-k-fold cross-validation, with k = 10.

Rule-based classifiers can easily overfit the training set, however, the testing error was close to the training error, which means the model, fortunately, did not overfit, mostly thanks to JRip's auto pruning step described in chapter 2.

Between PART and JRip, the latter was preferred because of higher overall accuracy and shorter rules. Below, for each class, we list JRips's accuracy on the

training set, in terms of the proportion of samples correctly classified.

HORMONAL THERAPY	88%
OSTEOPROTECTIVE THERAPY	90%
VITAMIN D THERAPY	87%
CALCIUM SUPPLEMENTATION	89%
VITAMIN D SUPPLEMENTATION	91%
HORMONAL THERAPY ACTIVE PRINCIPLE	60%
OSTEOPROTECTIVE THERAPY ACTIVE PRINCIPLE	45%
VITAMIN D THERAPY ACTIVE PRINCIPLE	50%
CALCIUM SUPPLEMENTATION ACTIVE PRINCIPLE	65%
VITAMIN D SUPPLEMENTATION ACTIVE PRINCIPLE	60%

The model is capable of accurately reproducing the relationship between the therapies and the analyzed variables. However, one can see a less than optimal performance for some active ingredient classes. Generally, with the number of instances being equal, one expects a multiclass problem having lower accuracy than a binary classification. There are, simply, for each class, fewer examples to train on.
For example, the HORMONAL THERAPY ACTIVE PRINCIPLE feature has four classes, the model trained on 51 instances out of 1819, which means an average of 12 instances per class—clearly, a small sample size. For this reason, it happens that active ingredient suggestions are not available.

Chapter 8

Conclusion

This thesis describes the implementation of a medical decision support system for treatment type and medication. It was imperative to design a system capable of providing the underlying reasoning behind the recommendations. The cdss integrates with the existing web application for patient health record management. The system relies on a rule-based classifier for generating explicit classification rules. It is also capable of self-learning, so the recommendations' accuracy is expected to improve with time.

The clinic is now testing the recommendation system. We'll be waiting for their feedback.

The system is certainly not flawless. There may be many potential improvements. Given that the dataset has several imbalanced classes, I would recommend focusing on other metrics other than accuracy like precision or recall.

A patient coming back to the clinic will be treated as a new one. The system would be more effective if it could differentiate between a new patient and a returning one.

For simplicity in the therapy prediction, we assumed independence between each class. This is not precisely the case as the decision on one therapy influences others. So the next improvement should consider the label inter-dependency.

I would also recommend rewriting the project in one programming language. Splitting the project in Python and Java required replicating the same logic, information, and data structures between the two languages, as well as making testing particularly difficult.
Besides these, many other minor improvements could be addressed.
In this internship, I have developed my skillset and gained valuable applied experience. I have broadened my knowledge in many areas such as version control, python, regular expressions, web development, weka suite, pandas library, machine learning, text mining, osteoporosis, decision trees, and rule-based classification. I believe this internship had a significant impact on my professional development.

Bibliography

[1] What are Ontologies and What are the Benefits of Using Ontologies, https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/.

[2] C.C. Aggarwal. *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis, 2014.

[3] Muhammad Ahmad, Carly Eckert, and Ankur Teredesai. Interpretable machine learning in healthcare. pages 559–560, 08 2018.

[4] R. Al Iqbal. Hybrid clinical decision support system: An automated diagnostic system for rural bangladesh. In *2012 International Conference on Informatics, Electronics Vision (ICIEV)*, pages 76–81, 2012.

[5] P.K. Anooj. Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules. *Journal of King Saud University - Computer and Information Sciences*, 24(1):27 – 40, 2012.

[6] Daniel Brown, A. Aldea, Rachel Harrison, Clare Martin, and Ian Bayley. Temporal case-based reasoning for type 1 diabetes mellitus bolus insulin decision support. *Artificial Intelligence in Medicine*, 85, 10 2017.

[7] Li Chen, Dongxin Lu, Menghao Zhu, Muhammad Muzammal, Oluwarotimi Samuel, Guixin Huang, Weinan Li, and Hongyan Wu. Omdp: An ontology-

based model for diagnosis and treatment of diabetes patients in remote healthcare systems. *International Journal of Distributed Sensor Networks*, 15:155014771984711, 05 2019.

[8] C. Cheng, N. Chanani, J. Venugopalan, K. Maher, and M. D. Wang. icuarm-an icu clinical decision support system using association rule mining. *IEEE Journal of Translational Engineering in Health and Medicine*, 1:4400110 – 4400110, 2013.

[9] William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 115 – 123. Morgan Kaufmann, San Francisco (CA), 1995.

[10] Thomas Davenport and Ravi Kalakota. The potential for artificial intelligence in healthcare. *Future Hospital Journal*, 6:94–98, 06 2019.

[11] Ken Farion, Wojtek Michalowski, Szymon Wilk, Dympna O'Sullivan, S Rubin, and Dawid Weiss. Clinical decision support system for point of care use ontology-driven design and software implementation. *Methods of information in medicine*, 48:381–90, 06 2009.

[12] B.A. Fette. *Cognitive Radio Technology*. Elsevier Science, 2009.

[13] Mir Anamul Hasan, Khaja Md. Sher-E-Alam, and Ahsan Raja Chowdhury. Human disease diagnosis using a fuzzy expert system, 2010.

[14] Johan Huysmans, Bart Baesens, and Jan Vanthienen. Using rule extraction to improve the comprehensibility of predictive models. *SSRN Electronic Journal*, 01 2006.

[15] Jieyue He, Bernard Chen, Hae-Jin Hu, R. Harrison, P. C. Tai, Yisheng Dong, and Yi Pan. Rule clustering and super-rule generation for transmembrane segments prediction. In *2005 IEEE Computational Systems Bioinformatics Conference - Workshops (CSBW'05)*, pages 224–227, 2005.

[16] Alan Jovic, Marin Prcela, and Dragan Gamberger. Ontologies in medical knowledge representation. pages 535 – 540, 07 2007.

[17] Linda Kohn, Janet Corrigan, and Molla Donaldson. *To Err is Human: Building a Safer Health System*, volume 6. 01 2000.

[18] Juan Albino Mendez, Ana Leon, Ayoze Marrero, Jose M. Gonzalez-Cava, Jose Antonio Reboso, Jose Ignacio Estevez, and José F. Gomez-Gonzalez. Improving the anesthetic process by a fuzzy rule based medical decision system. *Artificial Intelligence in Medicine*, 84:159 – 170, 2018.

[19] Stefania Montani and Manuel Striani. *Artificial Intelligence in Clinical Decision Support : a Focused Literature Survey*, pages 120–127. 08 2019.

[20] Nataliia Osypova, Vitaliy Kobets, and Tatyana Bazanova. Design, development and use of decision support systems in the study of economic disciplines in higher education. In *ICTERI*, 2017.

[21] J. R. Quinlan. Generating production rules from decision trees. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'87, page 304–307, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[22] Amir Mohammad Shahsavarani, Esfandiar Azad Marz Abadi, Maryam Hakimi Kalkhoran, Saeideh Jafari, and Shirin Qaranli. Clinical decision support systems (cdsss): State of the art review of literature. *International Journal of Medical Reviews*, 2(4):299–308, 2015.

[23] H.G. Sol, C.A.T. Takkenberg, and P.F. de Vries Robbé. *Expert Systems and Artificial Intelligence in Decision Support Systems: Proceedings of the Second Mini Euroconference, Lunteren, The Netherlands, 17–20 November 1985*. Springer Netherlands, 2013.

[24] Wikipedia. Clinical decision support system — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Clinical%20decision%20support%20system&oldid=971445884, 2020. [Online; accessed 07-August-2020].

[25] Ian H. Witten, Eibe Frank, and Mark A. Hall. Chapter 6 - implementations: Real machine learning schemes. In Ian H. Witten, Eibe Frank, and Mark A. Hall, editors, *Data Mining: Practical Machine Learning Tools and Techniques*

(Third Edition), The Morgan Kaufmann Series in Data Management Systems, pages 191 – 304. Morgan Kaufmann, Boston, third edition edition, 2011.

[26] Jianyu Yang. Classification by association rules: The importance of minimal rule sets. 01 2003.

[27] S. Yang, R. Wei, J. Guo, and L. Xu. Semantic inference on clinical documents: Combining machine learning algorithms with an inference engine for effective clinical diagnosis and treatment. *IEEE Access*, 5:3529–3546, 2017.