UCLouvain

École polytechnique de Louvain

Artificial Intelligence in Orthopaedic Recovery

Author: **Anat Levi**
Supervisors: **Pierre-Antoine Absil, Charles-Eric Winandy**
Readers: **Siegfried Nijssen, Emilie Renard**
Academic year 2018–2019
Master [120] in Data Sciences Engineering

Université catholique de Louvain

Abstract

Ecole Polytechnique de Louvain

Master in Data Sciences Engineering

Artificial Intelligence in Orthopaedic Recovery

by Anat Levi

Dealing with the timely challenge of helping medical teams increase their efficiency and identify crucial points in time where their intervention is needed, is the main objective of this research. Utilizing the data of orthopaedic patients, recovering from a knee/hip joint replacement surgery, we implement several Machine Learning (ML) models aimed at predicting patient pain levels and clinical interventions.

We start with the construction of a flat dataset, holding patient daily reportings, consisting of over 400 different features which are collected from various sensors. In order to reduce noise and data dimensionality, we then apply different statistical techniques for both building new features from existing ones and selecting the most relevant features for each of our prediction tasks. Data cleansing and analysis methods, including feature distribution plots, Pearson correlation, data scaling and data imputation were used in order to create the narrowest, fullest and most appropriate dataset for each prediction task and ML model. Finally, we implement and train decision trees, Support Vector Machines (SVMs) and Recurrent Neural Networks (RNNs) for obtaining high performing classifiers for our objectives.

The models built were carefully analyzed and tuned in order to extract maximum information about the most significant features and to be able to translate the results into recommendations for the clinical team. Based on separate blind test set evaluation, the different models (built for each research question) were able to predict the target with similar balanced accuracy (75% for manual intervention prediction and 85% for substantial pain prediction), however with a significantly better interpretability for the decision trees, making them a much more useful and insightful tool in this framework.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Pierre-Antoine Absil, Charles-Eric Winandy and Emilie Renard for their help, attention and good ideas which were very beneficial and motivating during the work on this thesis.

I would like to thank moveUP for taking me in as one of their own. Thank you for the opportunity, the vote of confidence, and for all your time and attention spent in helping me realize and achieve this project. Specifically, I would like to thank Bruno Neckebroek for his technical help and advice, which were crucial for obtaining and manipulating the data, and to Femke Smith for her patience, thorough explanations, sharp instincts and insights, which were very helpful for understanding the clinical aspects discussed.

On a more personal note, I would first like to thank my mother, without which these master studies would have never taken place. Mom, you are my inspiration. You showed us that through strong-will, hard work and persistence we can achieve anything we set our minds to.

I would also like to thank my father, who was my inspiration for choosing this specific topic, for his genuine interest and support.

Most of all, a big thank you to my family. My husband Itamar, who is always supporting me and pushing me forward to learn and grow. Thank you for the moral and technical support, for your help and ideas during these two years, and specifically throughout the work on the thesis. Thank you to my amazing kids who allowed me to study (even on weekends) and gave me the power to continue.

Contents

List of Figures

List of Tables

List of Abbreviations

EHR	Electronic Health Records
CDSS	Clinical Decision Support System
AI	Artificial-Intelligence
ML	Machine-Learning
PT	Physical Therapist
ANN	Artificial Neural Network
SVM	Support Vector Machine
PROM	Patient Reported Outcome Measures
bACC	Balanced Accuracy
TP	True Positive
TN	True Negative
CV	Cross Validation
CART	Classification and Regression Trees
VAS	Visual Analogue Scale
RNN	Recurrent Neural Network
mSBT	modified STarT Back Screening Tool
LSTM	Long short-term memory

Chapter 1

Introduction

We currently live in an era which is characterized by information and data explosion in almost every field of life. When making decisions, we can choose to consult many different data sources, which are easily and freely accessible and available. The main challenges have shifted from being able to obtain, collect and sense the data, to being able to efficiently filter and analyze the (too) large amounts of existing data.

With the introduction of Electronic Health Records (EHR), this challenge is becoming more and more common in the medical domain as well. EHR, represent the experience of both patients and doctors on the time axis [1]. Thus, these days, most clinicians have access to volumes of patients data, including a variety of data points per patient. While the ability to consult this data upon making a clinical decision can be very beneficial, clinicians usually lack the time or tools to do so.

A Clinical Decision Support System (CDSS) is a computerized information system designed to provide health professionals with assistance in clinical decision-making tasks [2]. A good CDSS is able to accurately predict the relevant information for a specific patient in a specific point in time, and recommend actions to be taken accordingly.

To this end, we see increasing use of Artificial-Intelligence (AI) and Machine-Learning (ML) models in healthcare. Building these models requires digitizing and preparing the data, performing a thorough feature selection process for identifying the subset of features which are most influential for solving the problem at hand, training models and assessing their performance and finally comparing between the different models and choosing the best one for use.

In this chapter we will first introduce the concrete real-life example which will be examined in this thesis and the objectives of the research. We will continue with a historical overview of the evolution of using AI systems in healthcare followed by a brief survey of relevant state-of-the-art research activities and conclude with the thesis structure and contribution.

1.1 Objectives

In this subsection we will list the objectives of the research. Before doing so, we introduce the medical-data which we have access to and the general goals of the company which gives us access to the information. Noteworthy that even though the discussion and analysis throughout the thesis are rather specific (discussed over a specific application), the analysis, tools, data-representation and conclusions are quite general and can be extended to other CDSSs.

"MoveUP" is a mobile-health company, developing a mobile application for patients undergoing a knee or hip joint replacement surgery. The application stores and monitors all data indicating the patient's pain levels, exercises, activity and medication intake. The data is being collected starting a couple of weeks before the surgery and throughout the patient's rehabilitation process, and is being used to provide real-time medical support and advice to ensure the best and fastest rehabilitation.

In the current setting, the medical advice is given by a team of Physical Therapists (PTs) and surgeons. The physical therapists regularly examine the data of each patient, and based on metrics and best practices, decide whether an intervention is needed for a specific patient. An intervention could be changing the recommended exercise scheme of the patient, making changes in the medications-intake dosage, or giving a specific advice to the patient regarding their daily activity or behaviour.

The follow up of the patients is carried out through an internal medical dashboard, presenting specific data points regarding all the patients currently rehabilitating. This allows the medical staff to get a quick picture of all the patients and decide on the order in which these patients should be addressed. For an illustration of the internal dashboard see Figure 1.1. Once the overview line of a patient strikes the attention of any of the clinicians, they can click on the *Details* link of that patient and see more detailed information.

In order to make the medical team's work more efficient, moveUP wishes to make use of all historical and current data collected about the patients, in order

Patient number	Limb	Surgery Date	Steps	Pain (0 = OK, 10 = Bad)	Feeling (0 = OK, 10 = Bad)	PT Review	Surgeon Review	Technical Review		
TP398	Knee right - total	1 Aug 2018 (1)	1652	0.2	2.4	-1	-10		Details	Expand
PP434	Hip right	1 Aug 2018 (1)	0	Not filled in	Not filled in	-1	-10		Details	Expand
PP430	Hip left	1 Aug 2018 (1)	18	Not filled in	Not filled in		-10		Details	Expand
PP442	Hip right	1 Aug 2018 (1)	824	1.6	1.8		-10		Details	Expand
TP387	Knee right - uni	31 Jul 2018 (2)	575	6.2	4.3	-1	-10		Details	Expand
PP424	Hip right	31 Jul 2018 (2)	0	6	2.2		-10		Details	Expand
TP372	Hip left	26 Jul 2018 (7)	932	4.9	4.1	-1	-4		Details	Expand
P431	Knee right - total	26 Jul 2018 (7)	4620	2.2	2.2	-1	-10		Details	Expand
PP461	Hip right - Resurfacing	25 Jul 2018 (8)	0	2.3	2.2		-10		Details	Expand
TP426	Hip right	24 Jul 2018 (9)	780	2.3	2.5	-1	-4		Details	Expand
TP357	Knee right - total	24 Jul 2018 (9)	697	1.8	2.1	-1	-4		Details	Expand
TP401	Hip left	18 Jul 2018 (15)	0	Not filled in	Not filled in		-4		Details	Expand
TP423	Hip left	17 Jul 2018 (16)	4107	3.8	3.3	-1	-10		Details	Expand
TP408	Knee right - uni	17 Jul 2018 (16)	2274	1.9	1.9	-1	-4		Details	Expand
TP301	Hip right	13 Jul 2018 (20)	3818	3	1.8	-1	-4		Details	Expand
PX012	Knee left - total	11 Jul 2018 (22)	0	1.5	0.2	-1	-10		Details	Expand

Figure 1.1: Illustration of the internal medical dashboard

.

to be able to predict in advance patient outcomes and change the treatment protocol accordingly. In addition, they ask to identify key features which affect the rehabilitation process, in order to expose them in the dashboard and furthermore, in order to make use of them in sorting the patients' list displayed.

In this work, we will try to address two objectives by applying AI and ML techniques for the daily prediction of:

1. The daily occurrence of a manual intervention for a patient
2. Whether or not the next day pain levels reported by the patient will be substantial (above 40 on a scale of 1-100)

By solving these problems, we will be providing the moveUP PT team with an automatic classification of the patients, such that the patients in need of assistance would be automatically identified. This way the team's work becomes quicker and more efficient such that each PT is able to monitor more patients per day, and avoids wasting time on examining the information of patients which, at the moment, do not require care.

1.2 Historical Overview and State of the Art

The main goal of AI applications in health-care systems is to identify and analyze relationships between prevention or treatment mechanisms and patient outcomes [3]. Such applications have been developed and applied in many different medical fields, including: diagnosis processes, drug development, personalized medicine, and patient monitoring and care. Numerous medical institutions and large technology companies such as IBM and Google have developed AI algorithms for improving different healthcare related aspects [4]. In addition, hospitals are looking into AI solutions in order to save costs, improve patient satisfaction, and better deal with staffing and workforce related constraints.

The first AI application in healthcare was a decision support system known as Dentral [5], the product of a research taking place during the 1960s and 1970s. Its primary aim was helping organic chemists identify unknown organic molecules, by analyzing their mass spectra and using knowledge of chemistry. While initially not being aimed directly at healthcare, it provided the basis for a subsequent system MYCIN [6], which was able to identify bacteria causing severe infections, and recommend the antibiotics type and adjusted dosage for each patient. As such, MYCIN was considered as one of the most significant early applications of AI in medicine [7].

During the 1980s and 1990s there had been significant advancements in microcomputers and network communications, allowing the development of more computationally extensive algorithms and systems. This caused many researchers and developers to believe that AI systems should also be used in the medical domain, in order to compensate for the absence of perfect (or complete) data to make decisions, instead of relying on the expertise of physicians [8]. Thus, systems applying fuzzy set theory [9], Bayesian networks [10], and artificial neural networks (ANN) [11, 12], have been developed to be used as intelligent computing systems in healthcare.

Since then, advancements in both technology and medicine have enabled the growth in research and development of AI and ML techniques for healthcare applications. More recently, decision trees [13] have been widely used in building prediction and classification applications for medical use. Decision trees institute one of the most effective machine learning based methods as they are very interpretable, free of ambiguity, can make use of both discrete and continuous variables in the learning process, and can be used for classification as well as regression purposes [14, 15]. Support Vector Machine (SVM)-based classifiers [16] have also been widely

used for medical purposes in recent years and, in many cases, have been shown to achieve better performance than other machine learning methods [17, 18]. State of the art solutions make use of deep learning and Recurrent Neural Networks (RNNs) for using sequential patient data to predict future outcomes [1, 19, 20].

In this thesis, the above mentioned techniques will be implemented on a dataset including continuous temporal data of patients. Unlike other medical applications, the moveUP application records numerous patient attributes on a daily basis. This provides continuous visibility to the patient state, rather than having just several data observations per patient, taken in different points in time along a large time interval. The chosen algorithms will therefore be adapted to the type of data and data-structure in our case.

1.3 Thesis structure and contribution

There exist several researches dealing with the prediction of pain levels after a joint replacement surgery. While in most cases the features available for the research include clinical and radiographic variables measured pre-surgery and several times during the first year after surgery [21], we have observations following the patient on a day-to-day basis, allowing for a more accurate analysis. In this work we use all the data at our disposal to predict patient outcomes, which makes data-preparation, data-structuring, data-access and the analysis process more complex. In turn and owing to the general importance of the types of data we have access to, the thesis starts with the important stage of understanding the data in Chapter 2. The thesis structure is listed in the following, as well as the main novelty and contribution of the different chapters.

Chapter 2 presents the data structure and describes all the data points which were available for this research. It further describes the preliminary analysis which was conducted on the data and the methods used for data cleansing and feature selection. Our contribution is the generation of our own clean and ready to use dataset, composed of 105 features, carefully chosen from the originally available 340 features.

Chapter 3 focuses on the models built for predicting a patient's need for manual intervention by the moveUP medical team. It explains the theory behind each model and details the steps taken in the modeling process. We introduce our method for building decision trees and SVMs on sequential data, and the construction of new features for embedding the time series information into the data. For each model type, the final model is presented including its performance assessment. This

chapter is concluded by a quantitative and qualitative comparison between the different models, and an additional construction and analysis of an ensemble model, combining the different models discussed. The main contribution of this chapter, is the revelation of the key features which take part in the decision to intervene for a specific patient. This analysis exposed some features which are not used in the current protocol, thus allowing the clinicians the gain of new insights.

The aim of **Chapter 4** is to describe the models built for predicting whether next day pain levels reported by the patient would be substantial. This chapter opens with significant changes to the initial dataset, including additions of new features, data analysis and feature reduction. We then continue to build decision trees and RNNs, describing the design choices made along the way. The chapter is concluded with a quantitative and qualitative comparison between the models and an analysis of the results. The work described in this chapter reveals the most significant features responsible for pain increase and provides intuition to the clinicians regarding the medical recommendations that should be given, in advance, to the patient, in order to avoid substantial pain.

To conclude, we come back in **Chapter 5** to the initial goal of this project, using AI and ML techniques for making the work of the PT team more efficient, and reflect on how this research contributed to its achievement. In addition, we mention some topics that could further contribute to this goal, but were not covered in the present work.

Chapter 2

Data Preparation

A first challenge in using data for machine learning algorithms is having the data digitized, well structured and cleansed. In the medical domain, still today we have many data points which exist only in hard copy, and are not recorded into any computational database. In other cases, the data is digitally stored, but either in a non-relational database, making it much harder to access and query in an efficient way, or the data is stored in several different systems and, therefore, needs to be extracted and loaded into a new flat structure [22]. Restructuring and cleansing the data is at the heart of applying any machine learning based technique and is therefore a crucial first step.

The moveUP production data is stored in a MongoDB database, which is a NoSQL database, used for storing json documents. While this database may be very convenient and efficient for storing and pulling information for a certain key value (e.g. patient), this database poses a big challenge when it comes to reporting and analysis needs. Thus, in order to be able to easily query and analyze the relevant data, a preliminary data-preparation stage was conducted with the objective of pulling the data from the MongoDB and re-structuring it in a more convenient relational schema.

The data stored by the moveUP solution can be divided into five groups:

1. Patient data – all static information about a patient, including: age, gender, surgery details, marital status, education etc. This data-set represents mostly discrete categorical features.

2. Medical scores – medical questionnaires, such as PROMs (Patient Reported Outcome Measures) [23], taken by each patient several times during their moveUP process. Typically: pre-surgery, six weeks post-surgery, three months post-surgery and six months post-surgery. The PROMs' questions try to measure the patient's health status, their health-related quality of life, their

functional status, their symptoms and symptom burden and health-related behaviors such as anxiety and depression. This dataset is represented by mostly discrete numerical features, on a scale of 1-100 and binary yes/no features.

3. Daily data – all data points collected on a daily basis both from the wearable-device (reporting number of steps taken, calories burnt etc.), and from a series of daily questionnaires taken by the patient. The questionnaires include information indicating pain levels, general feeling, medications taken and ability to perform physical exercises and daily activities. This dataset is also represented mostly by discrete numerical features, on a scale of 1-100 and binary yes/no features.

4. Exercise scheme – all data regarding the physical exercises a patient needs to preform throughout the rehabilitation process. This includes both the instructions regarding the exercises a patient needs to perform each day, and measurements regarding their actual performance.

5. Chat – history of all chat messages transmitted between the patients and the physical therapists.

At this stage we had at our disposal in the relational schema only the features belonging to groups 1-3 above, such that the exercise scheme and chat data were not taken into account. In order to use both the static and dynamic features available in the data, SQL queries were applied to join all tables (over 20), in order to form a big flat dataset to be fed to the different ML algorithms. The data in this dataset was carefully examined and multiple reportings for a patient in the same day were removed. After the cleansing process, this initial dataset consisted of 28,610 data rows, each of which including 340 features. This dataset corresponds to 800 different patients. Each row in this dataset is a tuple representing a single patient on a single day, as can be seen in Table 2.1, where Patient ID is the unique patient identifier in the system, DATE is the date in which the results were reported, S_1, ..., S_n represent the set of patient static features and D_1, ..., D_m represent the set of patient dynamic features.

For a past patient (which had already recovered) we have about 90 tuple rows representing 90 rehabilitation days, each consisting of:

- 65 static features representing the patient's profile. These features repeat the same values over all rows of the same patient
- 45 dynamic features reported on a daily basis

Patient ID	DATE (Y-M-D)	Static				Dynamic			
		S_1	S_2	...	S_n	D_1	D_2	...	D_m
0	2017-01-01	30	2	...	A	0	89		5
0	2017-01-02	30	2	...	A	1	98		8
1	2017-03-01	57	0	...	C	2	74		2

Table 2.1: Dataset structure

- 230 medical score features, representing the temporal state of the patient and taken several times along the rehabilitation process. For simplicity, just the first occurrence (pre-surgery) of each medical score was considered in the above mentioned dataset, and treated as a static feature.

2.1 Feature Selection

When using ML algorithms to model a problem, the size of the feature set used and its content are crucial for both the performance of the classifier and for the efficiency of the training and classification processes. Including irrelevant or redundant features may degrade classifier performance. Moreover, the bigger the feature set, the more computationally extensive the learning process will be, and also the classification for new unseen data points. In addition, the use of fewer features reduces the complexity of the model, and results in a simpler model, which is also easier to understand and explain [24]. Therefore, it is essential to select the subset of features which are the most relevant to the predictive modeling problem at hand.

Feature selection techniques can be used to automatically select an optimal subset of features for a specific problem and dataset. These techniques can be divided to two main groups: *Filter Methods* provide a ranking over the features, denoting the usefulness of each feature to the classification task. Based on this ranking, the best N features can be chosen as the optimal subset; *Wrapper Methods* subsequently choose different subsets of features, build a classifier based on each subset and assess the classifiers' performance. The latter approach is much more time consuming and computationally extensive, as it needs to fully train models for evaluating many different feature subsets.

In this thesis, we are dealing with two different classification problems, and we aim to use several different ML algorithms. Therefore, we were searching for

methods to help reduce the dimensionallity of the initial dataset, such that the resulting data subset would remain optimal for the different modeling tasks at hand. We, therefore, found it appropriate to use the *Pearson's correlation coefficient*, which is a filter method. In essence, the method applies a statistical test measuring the correlation between two continuous feature variables. Features which are highly correlated are more linearly dependent and, hence, have almost the same effect on the dependent variable. Thus, when finding a correlated pair, one of the features in the pair can be removed from the dataset.

Before applying this method, an important pre-processing stage has to take place in order to cleanse the data. A common restriction of many ML models is that the dataset has to be clear of null (missing) values, otherwise the modeling tools are unable to compute the classifier. Therefore, any null value originally appearing in the dataset, must be replaced by an actual value, or alternatively the whole data example needs to be removed from the dataset.

Thus, a first step of preliminary analysis was performed, in which the number of null values was counted for each of the features. Features with more than 3000 null values (10% of the initial data size) were extracted for examination. For each of these features, a solution for imputation (replacing the missing data with an actual value) was considered. When no appropriate solution was found for a feature, this feature was removed from the data set.

For example, some questions are dependent on one another, such that if a patient answers *No*, the next question is not presented and, thus, ends up with a null value in the database. For these kind of questions the imputation rule was relatively easy and was computed based on the value given in the primary question. On the other hand, some questionnaires were discovered to be introduced to the application at a significantly later stage than others, causing a large number of records to have missing values for the entire questionnaire. In these cases, we decided not to impute the answers of an entire questionnaire, and simply omit these questionnaires entirely from the research. This step resulted in a major reduction of the dataset, excluding 210 features.

In order to eliminate redundant features from the reduced dataset, the above mentioned Pearson correlation was used. The correlation of every two-features pair was calculated, and all feature pairs having a correlation of above 85% (in absolute value) were recognized. For each of these pairs, the most accurate or meaningful feature was selected and the other removed. For example, *number of steps*, *calories burnt*, and *distance walked* are very highly correlated. Looking into these measurements we discovered that while *number of steps* was measured by the wearable device, *calories burnt*, and *distance walked* were calculated measurements

and are thus less accurate, so they were the ones to be removed.

It is clear that there exist features that might exhibit very high informativeness, however, the information they provide is not useful (e.g. patient ID, patient email address or reporting date). In practice, these features are very dangerous when different ML algorithms are being used. For example, when a decision tree algorithm is used (see Subsection 3.3), it might actually and falsely rank these features first. Therefore, the third and final step of the data-preparation included going over the remaining features and manually removing those that seemed irrelevant for the prediction tasks. Overall, the feature selection resulted in removing 235 features in total, resulting in a dimensionality of 105 features.

Chapter 3

Manual Intervention Prediction

The first research objective of this thesis is to build a classifier that will be able to predict *whether or not the patient will require manual intervention in the coming day*. This is actually a two-class classification problem, which aims at automatically identifying those patients which require the PT's attention, such that the medical team can attend to them first.

For this task we evaluated and optimized two different types of models and compared between their performances: Decision Trees (Subsection 3.3) and SVMs (Subsection 3.4). Before discussing the models and results we next define the Problem, the optimization variables and the metric which was used to evaluate performance.

3.1 Problem Definition and New Features

First and foremost, the target binary feature of manual intervention needed to be defined and computed, as there was no direct representation for this event in the existing database. We defined this event as *True* when either the exercise scheme was changed or a chat message was sent from the PT to the patient. This feature was thus composed from data extracted from the historical log, documenting changes made in the exercise scheme, as well as the chat history reporting. For extracting this information, two binary $[0, 1]$ features were defined per patient per day: was a change reported in the patient's exercise scheme; was a chat message from the PT reported. Once the target feature was added to the training data, the class labels' distribution in the training data was calculated and we found that manual intervention appears in 55% of the cases and does not appear in 45%, as illustrated in Figure 3.1.

The correlations between each of the features and the target were calculated. The results showed very low correlation (absolute) values, such that the most

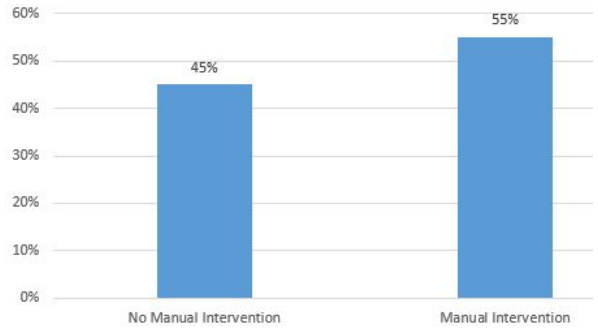


Figure 3.1: Distribution of the target (manual intervention)

correlated feature was *number of days after surgery*, with a negative correlation of -0.16, indicating that the farther a patient moves from the surgery on the time axis, the less likely they are to experience a medical intervention. The second most correlated features was the indication that a patient uses crutches, with a correlation of 0.14.

Second, each of the data rows to be used as input to the model needs to represent the sequential time series information in the data and somehow include the trend of the past behaviour at each time step. To achieve this, new features were defined based on the dynamic features reported on a daily basis. For each dynamic feature, its last three days average, and both the difference and the ratio between the current point and the last three day average were added to the data set. It should be noted that the decision to base the historical trend on the last three days alone was motivated by information obtained from the PTs, which took part in the process of characterizing the medical data.

Finally, all rows including one or more null values, at any of the columns, were removed in order to prepare the data for modeling, as discussed in Subsection 2.1. The outcome of this step was that from a full dataset including 28,610 observations, each consisting of 125 features, only 19,480 observations were left after the null removal.

3.2 Performance Evaluation

Before evaluating the models, we briefly discuss the performance metric that we use. As the classes in our problem are not perfectly balanced (as most real-life problems), the performance measurement which was chosen to compare between different model configurations and different models was the *Balanced Accuracy*

(bACC) [25]. The commonly used *accuracy* metric does not perform well with imbalanced data sets, as it tends to favour prediction for the majority class. bACC overcomes this problem, by normalizing true positive (TP) and true negative (TN) predictions by the number of positive and negative examples, respectively, and dividing their sum into two, thus giving an equal importance to type 1 and type 2 classification errors. This is computed by the following formula (where FN stands for false negative, and FP stands for false positive):

$$\frac{TP}{2(TP+FN)} + \frac{TN}{2(TN+FP)}$$

In order to avoid over-fitting, the data was initially shuffled and divided to 80% training and 20% test. For tuning the meta-parameters of each of the models, a 10-fold cross validation (CV) protocol was used [26], such that the 80% training data was divided to ten equally sized subsets, and in each of the CV folds, a different set of nine-subsets was used for training and the remaining subset was used for validation. The results and performance of each fold were calculated on its validation set and saved, and finally the average performance of each configuration over the 10-folds was calculated and stored. The meta-parameter values used in the best performing configuration were chosen to be used in the final model, built on the whole training set containing 80% of the original data. For comparing between different final models, the performance of each model was computed on the 20% test set.

3.3 Decision Trees

A decision tree [27] is a flowchart-like structure representing a decision procedure for determining the class of a given instance. Each internal node of the tree specifies a test on a specific attribute. Each such test partitions the data examples at the node to branches, representing the possible outcomes of the test. Each leaf node represents a class label. The paths from root to leaf represent classification rules.

The process of learning a decision tree from training data is a top-down process, starting from the root of the tree and recursively splitting the tree nodes down to its leafs. At each step, the algorithm greedily chooses the attribute which maximizes the information gain, with respect to the classification problem, and assigns this attribute with the current internal node. The information gain [28] is the expected reduction in uncertainty, represented by entropy. The information gain of splitting sub-tree with data sample D by attribute x_i is defined by:

$$Gain(D, X_i) = entropy(D) - \sum_{v \in values(x_i)} \frac{|D_v^i|}{|D|} entropy(D_v^i)$$

where D_v^i are the examples in sub-tree D holding value v for attribute x_i.

There are several reasons for choosing a decision tree for our purpose:

- It is able to model a problem with both continuous and discrete features.
- It outputs a set of decision rules which are very easily understood and interpreted by humans.
- It is robust to outliers in the data, as the decision at each leaf node is based on the majority vote at that node. Assuming outliers are not numerous, they will not affect the final outcome.

However, it should be noted that decision trees are in some cases unstable, such that a small change in the training data can result in a completely different tree. In addition, when the dataset includes categorical variables, each having a different number of levels, the information gain metric favors those attributes consisting of more levels, even when these are not the most relevant ones to the classification task.

In order to avoid over-fitting when growing the tree, some meta-parameters need to be trained in order to choose the degrees of both the pre-pruning and post-pruning mechanisms. These mechanisms make sure that a node does not branch when either the sample size left on that node is too small, or the branching is not statistically significant enough.

3.3.1 Modeling and Results

The models were built using the *rpart* R library for implementing the *Classification and Regression Trees (CART)* algorithm, which had been applied in numerous applications in the medical domain. CART [29] builds a binary tree by recursively splitting the data space into hyper-rectangles. At each step, the space is split to two parts, based on the attribute that maximizes the drop of impurity. A node is considered pure when all training examples assigned to it belong to the same class. In the CART algorithm, the *gini*-impurity measure is used for calculating the drop of impurity, similarly to how we have seen before the entropy is used in information gain. The impurity drop of splitting sub-tree with data sample D by

attribute x_i is defined by:

$$Drop(D, X_i) = Gini(D) - \sum_{v \in values(x_i)} \frac{|D_v^i|}{|D|} Gini(D_v^i)$$

where D_v^i are the examples in sub-tree D holding value v for attribute x_i.

The learning process of the trees included tuning the model meta-parameters *minsplit* and *cp*, in charge of the pruning of the tree. Minsplit defines the minimal number of training examples assigned to a node, allowing it to further be split. The smaller the value for this parameter, the more the tree will fit to the training data and the greater is the risk of over-fitting. Similarly, the cp parameter is used to save computing time, by pruning off splits that are considered not worthwhile. Namely, a split that does not decrease the overall lack of fit by a factor of cp is pruned. During the meta-parameter tuning process, three values for minsplit and four values for cp were examined.

The performance of the first classification tree built for predicting manual intervention was quite poor (58% bACC on test). These results triggered a deeper analysis into the target variable. In essence, it illustrates just another example that model automation needs careful and insightful human control and explanatory parameters. Figure 3.2 displays the distribution of the manual intervention scenarios. The fact that 54% of the data accounts for chat messages seemed quite extensive and required some extra analysis. After consolidation with the PTs, it was revealed that they obviously have to answer any chat conversation initiated by the patient, but the new information learnt is that when there was no communication with the patient in the last three days, the PTs' protocol requires them to send a message to the patient. The implication of this is that in many cases, a manual intervention in the form of a chat message is being reported, when actually there is no clinical

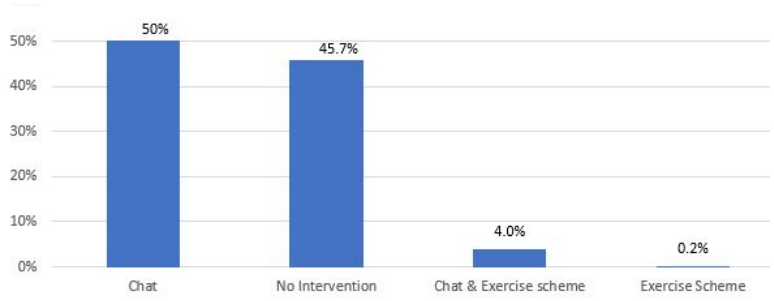


Figure 3.2: Distribution of manual intervention types

indication for it. In order to remove this noise from the model, two additional features were added: a binary indication to who initiated the chat conversation (patient or PT) and a binary indicator for having no communication in the past three days.

The correlations between the newly added features and the target were calculated, revealing that these two new features are now the most correlated features in the dataset with the target, and with much larger correlation values. The indication that the patient started a chat conversation had a 0.42 correlation with the target, and the indication that there was no chat conversation with the patient during the last three days had a 0.32 correlation with the target.

Figure 3.3 presents the final classification tree for predicting manual intervention (with the meta-parmeters configuration of cp=0.002 and minsplit=15). Each terminal node includes the following information about the data associated to it from top to bottom: indication of the class predicted (0 representing no intervention and 1 representing an intervention); the proportion of cases out of the data associated to this node, in which the labeled class was 1; the percentage of cases out of the full dataset that are associated to this node (such that the sum of this value over all terminal nodes is 100%). The second metric is very useful, as it can be treated as the extent of certainty of the prediction at each node.

First we note that the tree obtained goes hand in hand with our preliminary analysis results, as the first three features at the top of the tree are the ones found to be the most correlated with the target. The tree shows that in 20% of the cases, the patient initiates a chat conversation, for which the PT is obliged to

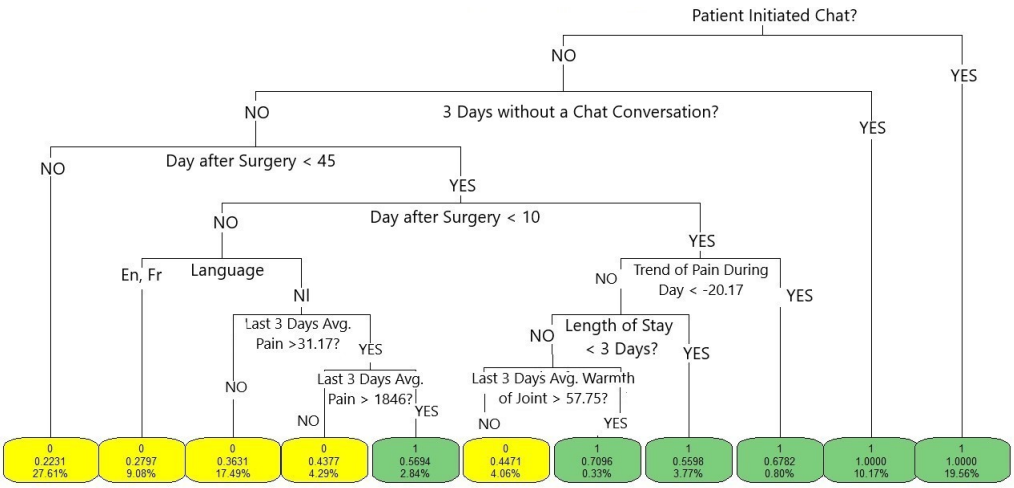


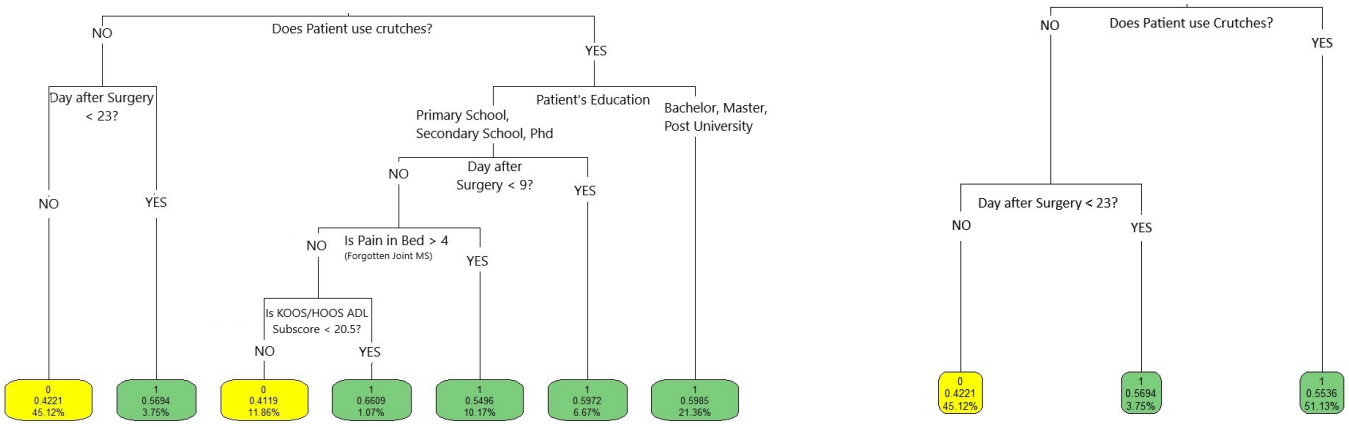
Figure 3.3: Classification tree for predicting manual intervention

answer. Additional 10% of the cases reflect the situation in which there was no communication with the patient in the last three days, which results in a chat initiation by the PT. The classification task itself then addresses the remaining 70%, from which the more critical patients need to be identified. We see that the most important feature chosen by this sub-tree is the number of days after surgery, and the following features include the trend of reported pain level of the patient in the operated joint during the day, the language spoken by the patient, the patient's length of stay in the hospital after surgery, the average number of steps taken in the past three days, and the average level of warmth felt in the operated joint in the past three days. The bACC achieved by this decision tree on the test set was 75%.

Data-Sensitive and Large classification Variance:
As the application is on-line, throughout the research we had new patients joining the system on a daily basis, such that we observed a significant growth in the data volumes over time. Therefore, trees based on different data volumes, including slightly different feature sets were constructed throughout the modeling period. It should be noted that these trees were quite different from one another, demonstrating one of the biggest limitations of this model type, its high sensitivity to the data, and the creation of classifiers with large variance.

Seeing the large percentage of data accounting for patients initiating a chat conversation (20%), raised another interesting question: What triggers the patient to do so? For answering this, an additional tree was built, as shown in Figure 3.4a, predicting whether or not a patient will initiate a chat conversation. We can see that the crutches use is the most influential feature in this model. This model, however, performed quite poorly, with only 56% of bACC. Reaching the same performance, this tree can be simplified to the one shown in Figure 3.4b.

To conclude this subsection, we highlight that using decision trees we were able to create comprehensive models, which are easily translated into a set of rules. A nice advantage of this model is that it also provides the distribution of the data between the leaf nodes, as well as the certainty level of the classification given at each one. This allowed us to easily find erroneous data points or other faults in the dataset construction, as well as to share the models with the PT team, and receive their feedback at the different modeling stages. It is further important to stress that it provides large intuition and baseline understanding of the data. As such, it is recommended as an important sanity and pre-processing stage.



(a) (b)

Figure 3.4: Classification trees for predicting chat initiation by patient. (a) Full Tree (b) Pruned Tree

Even though the model at hand was able to generate a tree with sufficient bACC for predicting manual intervention, this was not the case for understanding in which circumstances the patient will initiate a chat conversation. Trying to further improve the bACC obtained for the primary question, and allow for a comparison between different models, we further tried to build a classifier using SVMs.

3.4 Support Vector Machines

The classification problem can be modeled in space, such that each data example is a data point, whose coordinates are defined by its feature values, and its color represents the class it belongs to. In this setting, we are searching for a hyper-plane separating between the data points associated with the different classes. A SVM is a discriminative classifier, formally defined by a hyper-plane separating between the classes [16]. As there could exist infinite separating decision boundaries for given data and classification task, the uniqueness of the SVM is providing the one which ensures a maximal margin between the training patterns and the decision boundary. See Figure 3.5 [30] for illustration.

The SVM linear discriminant g(x) is defined by a linear function of the training examples matrix X ($n\ examples\ X\ p\ features$), as follows:
$g(x) = w^T X + w_0$, where vector w and constant w_0 are the model parameters. The geometrical margin is defined as $\frac{1}{||W||}$, such that maximizing the margin is equivalent

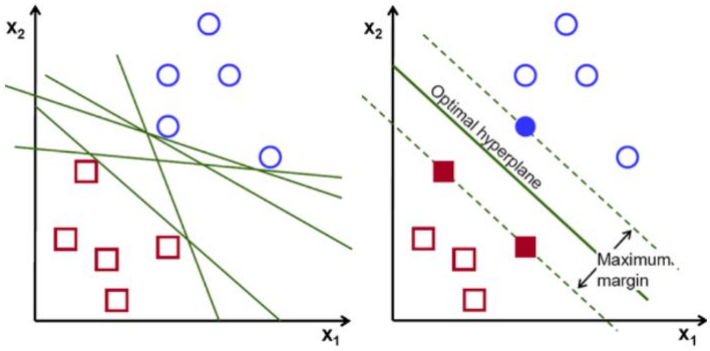


Figure 3.5: SVM illustration

to minimizing $||W||$, or equivalently minimizing $\frac{1}{2}||W||^2$, which is a mathematically easier task, as it deals with a quadratic form in the positive quadrant [31].

The training examples are not always linearly separable in the input space, in which case the decision boundary is non-linear. SVM can also be used for solving these problems, by applying a projection of the input data to another, higher dimensional space, referred to as the feature space. This technique is applicable to different classification functions, including polynomials, Sigmoids, and Radial Basis Functions.

The effective number of parameters is adjusted automatically by the model during the learning process, to best generalize the problem. The separating hyper-plane itself is expressed as a linear combination of the *support vectors*, which are the subset of training examples lying closest to the decision boundary.

Outliers in the training sample may cause the classes to not be linearly separable in the feature space. The distances of outliers, lying within the margin or on the wrong side of the decision boundary, from the decision boundary itself, are measured by slack variables. The optimization problem is then relaxed by increasing the margin constraints and allowing for some slack, turning the "hard" margin into a "soft" one. In order to balance between the two objectives: maximal margin and good generalization, the optimization problem is regularized such that C, a meta-parameter which stands for cost, is the regularization term. The optimization problem then becomes:

$$min_{w,\xi}\frac{1}{2}||W||^2 + C\sum_{i=1}^{n}\xi_i$$

Where C is responsible for the size of the soft margin of the SVM and ξ_i represents the slack of training example i. The smaller the value of C, the less the model penalizes for slacks, and thus the greater the soft margin.

The most significant meta-parameter of this model is the kernel type, varying between Linear, Polynomial, Radial basis and Sigmoid. Other meta-parameters are specific to each kernel type and include the degree of the polynomial kernel, and the γ parameter used in all kernels except for the linear one. This parameter defines the linearity degree of the decision boundary. The smaller γ is, the more influence training examples which are located far from the decision boundary have on the final solution, and thus the closer the decision boundary will be to a straight line. Conversely, as γ grows bigger, the more curvy the decision boundary gets, as only the closest points to it are taken into account, and thus outliers have a great influence on the final solution. Therefore, choosing a too large value for γ might result in over-fitting.

One of the strengths of the linear kernel is that the parameters of the model, represented by the coefficients of the hyper-plane, could be interpreted as the importance of the features. Features which are identified by the SVM training algorithm as insignificant are fitted with a coefficient value close or equal to zero, whereas features which are identified as relevant for the classification task are fitted with a relatively large coefficient (in absolute value). This attribute, however, is not kept when fitting a non linear SVM. In that case the coefficients of the model implicitly correspond to the higher dimensional space and are therefore not useful in interpreting the model.

SVMs are among the most successful techniques for pattern classification due to several advantages they hold. Firstly, as the optimization problem deals with maximizing a quadratic form, the solution obtained is always the global optimum, and is unique under some conditions on the training examples. Secondly, margin maximization improves the classifier's generalization, as it makes it insensitive to small changes in the model parameters. Thirdly, using a non linear kernel, changes the dimensionality of the problem from the original data dimension to the feature space dimension, which is determined by the number of training examples. This makes SVM very well fitted for classification problems in which there is a relatively small number of data examples, and the data's dimensionality is relatively high. For these above mentioned reasons, and specifically given the size and dimensionality of our dataset, it seemed fit to use the SVM as a competing model.

3.4.1 Modeling and Results

The SVM model uses the representation of the data points in space. For this reason, categorical features cannot be used as-is in the model. If a categorical feature is ordinal, its values should be transformed into an ordinal numerical value. Otherwise, the feature could be broken down to dummy variables. i.e. a feature having N possible values, can be broken down to N mutually exclusive dummy variables, such that each i dummy variable represents if feature value i was or was not seen in each example.

In turn, the dataset which was used for modeling the decision trees was taken, and all categorical features extracted for examination. Out of the the group of all categorical features, only seven were not ordinal. From these, four were broken down to dummy variables and three were removed.

The models were built using the *e1071* R library [32], for training SVM models. As the SVM optimization problem deals with minimizing the decision vector W, the optimal hyper-plane is influenced by the scale of the input data. It is therefore considered good practice to standardize the data prior to training the SVM model for obtaining better results. In the library used here, data columns are standardized by default to zero mean and unit variance. The centered and scaled values are stored internally and are the ones used both for training and for later predictions.

In this research we have trained four SVM models, one of each kernel type. In the following we present the training process and results for each kernel type. During the training process of all kernel types, three values of the meta-parameter C were examined: starting from 1, representing the soft margin approach, allowing training examples to lie within the margin or to lie on the wrong side of the separating plane, and gradually elevating the value towards the hard margin approach.

During the training process of the linear kernel, the best performing model was obtained for C value 1. Noteworthy that for very high values of C, the model could not find the optimal solution. This implies that the training data is not linearly separable.

Table 3.1 presents the coefficient values of the top ten rated features in the final linear SVM built, based on the full training data. We note that the two most dominant features of the linear SVM are consistent with the ones defined by the decision tree and are the two chat indicators. We also note that their coefficients are significantly larger than the ones obtained for the features that follow. On

Feature	Coefficient
patient_initiated_chat	1.220953
3_days_without_chat	0.9591257
day_after_surgery	-0.1780687
crutches_use	0.1212989
oos_pain_sub_score	0.117831
last_3_days_avg_Dafalgan_Paracetamol	0.1135305
dutch	0.1070717
french	-0.1026608
pain_walking_more_than_15_minutes	0.1015974
Dafalgan_Paracetamol	0.09732956

Table 3.1: Linear SVM top ten weighed features

the other hand, we see that the top ten rated features in this model include some interesting features which were not used by the tree, such as medication intake and pain levels.

The test performance of this model based on the bACC metric was 74%, similarly to the corresponding tree model.

For the Polynomial kernel, the projection used is defined by: $(\gamma W^T X)^{degree}$. While tuning the degree meta-parameter, polynomial degrees of 2, 3 and 5 were examined. The best performance achieved in the training process was achieved for a C value of 1 and a degree of 2. The degree implies on the form of the projection that allowed best separation between the classes, and the C value again implies that the best separation was obtained with a soft margin rather than a hard one. The test performance of this model based on the bACC metric was also 74%.

For the Radial basis function kernel, the projection used is defined by: $e^{-\gamma|w-x|^2}$. While tuning the γ meta-parameter, three values were tested: the default one provided by the tool which is equal to $\frac{1}{data\ dimension} = 0.008$, one smaller and one larger value. Analyzing the CV results, the performance was improved for lower values of γ. Therefore, additional, smaller values were tested, and the final model was built using a γ of 0.0033. Also here, the value of $C = 1$ was found to be best. The bACC calculated on the test set for this model was 75.5%.

For the Sigmoid kernel, the projection used is defined by: $tanh(\gamma W^T X)$. The γ meta-parameter was tuned here in a similar way to the one described in the radial

Model	bACC
Decision Tree	75%
Linear SVM	74%
Polynomial SVM	74%
Radial Basis SVM	75.5%
Sigmoid SVM	74%

Table 3.2: Model performance comparison

basis function, with similar results. Thus, the final model here was also built with $C = 1$ and γ=0.0033 and its bACC test performance was 74%.

3.5 Model Selection

Table 3.2 summarizes the test performances of the final models from each model type. It is clear to see that there are no significant differences between these performances, such that quantitatively speaking the models are equally efficient.

This triggered an analysis into the specific predictions of each of the 5 models, and more specifically the differences in prediction between them. Using the predictions made by each model on the test set, we first analyzed what we refer to as *the agreement distribution between the models*. As displayed in Figure 3.6, in 75% of

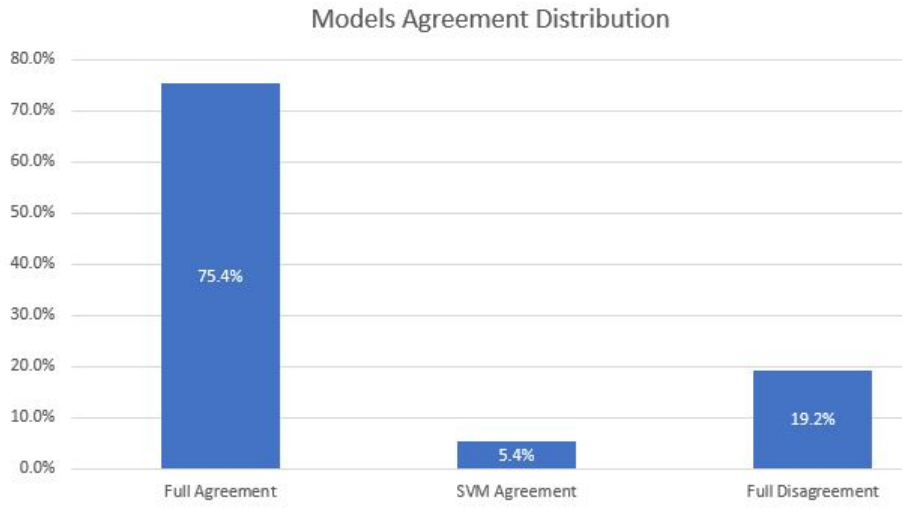


Figure 3.6: Classifiers' agreement distribution

the cases there was a complete agreement between the five different classifiers, such that all gave the exact same classification. In additional 5% of the cases, the four SVM classifiers gave the same classification, in disagreement with the tree. In the remaining 20% of the cases, there were disagreements between the different SVM models, such that each time at least one was in agreement with the tree.

We then continued analyzing this data in order to understand the accuracy of the prediction in each of the above cases. Results are described in Figure 3.7 and show that out of the 75% of the test set, in which we witnessed full agreement between the models, their classification was correct 80% of the time. Out of the 5% of the test set, in which we witnessed a full agreement only between the SVM models, we see that the SVMs were slightly more accurate than the tree, with 55% accuracy. In the 20% of the test set, in which we witnessed a mixed pattern of agreements, the tree was able to classify correctly 52% of the time.

Given these results, we were wondering if an ensemble model, incorporating the predictions of the five models into a single prediction based on the majority vote, could enhance the models' performance. Thus, an additional ensemble model was built and tested. The bACC performance of this model on the test set was 75%, such that it was not able to out-perform the two best models previously identified.

However, when one chooses a model to base a decision support system on, there are additional considerations, other than the performance, to be taken into account. First and foremost, the interpretability of the model. It is always nice to have a model which can easily be understood. It helps gain the trust and cooperation of the professionals using the model, it aids in finding bugs and inaccuracies when they exist, and it sheds light and new intuition about the classification problem. While in some fields this quality is just a "nice to have", in the medical domain this

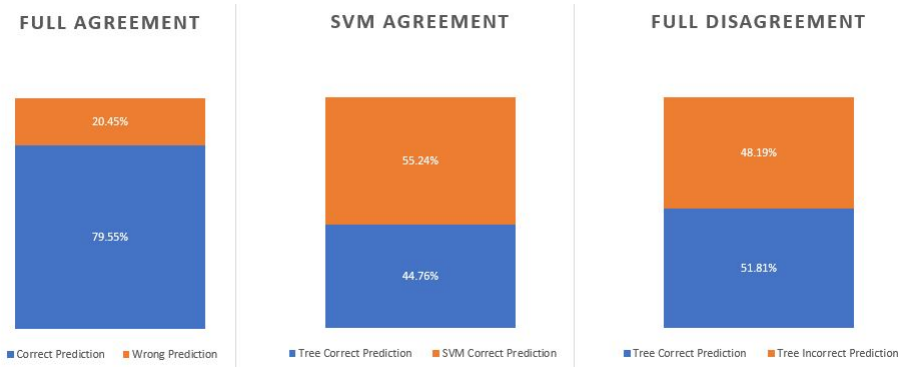


Figure 3.7: Prediction accuracy distribution

quality is often a must. Care takers will be reluctant in many cases to use a "black box". They will want to make sure that the advice they are giving to patients is based on true logic, and on things they understand and can explain. This is the primary reason for which the decision tree is the preferred model among the five examined.

Secondly, model complexity and computational intractability also play a role here. Implementing the model into the application is not a static process. The model chosen will be recomputed on a regular basis on growing amounts of data, and the results will be compared to the existing model in order to make modifications and improve model performance. Training an SVM is a much more computationally extensive mission then training a decision tree. The number and relevant space of meta-parameters to be tuned is much larger, the computation of the model itself is much more complex and this results in much more time needed in order to maintain and improve the SVM model in comparison to the decision tree.

Given these two qualitative reasons, along with the similar quantitative results obtained, the decision tree was chosen to be the leading model, and is indeed being incorporated into the moveUP application, for providing professional recommendations to the PT team.

Chapter 4

Pain Level Prediction

Our second research objective was to build a classifier that will be able to predict *whether the pain levels reported by the patient the next day will be substantial.*

The pain reporting via the moveUP application is based on the Visual Analogue Scale (VAS). VAS [33] is an instrument aimed at measuring an attitude that is believed to range across a continuous range of values and cannot be easily and directly measured. To this end, measuring pain perception by patients is vastly done by using the VAS score. In effect, the VAS is usually represented by a 100 mm horizontal line, anchored by word descriptors at each end, on which the patient marks the point best depicting their perception of their level of pain. The VAS score is determined by the distance in mm between the left hand end of the line to the point marked by the patient. The VAS used in the moveUP application for the question which is being examined here is illustrated in Figure 4.1

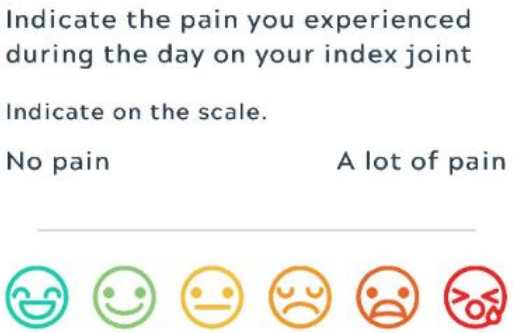


Figure 4.1: Pain level Question from the moveUP application

For defining the cutoff value to be used for calculating the target, *substantial pain*, on our data, we used a pain model appropriate to be used in efficacy studies in acute pain resulting from a major orthopedic surgery [34]. According to this model, moderate to severe pain is defined as a VAS score greater or equal to 40. Thus, this was chosen as the cutoff value for recognizing substantial pain in this work.

For modeling the problem at hand, we decided to use a tree (Subsection 4.2) as our baseline model, given its interpretability and given the results presented in the previous chapter. As a second model we decided to use a state of the art solution for sequence modeling, an artificial RNN (Subsection 4.3). We found RNNs to be the best fit here, as our data is a sequential set per patient, representing their clinical journey over time and RNNs are a family of neural networks specialized for processing sequential data.

In this chapter we describe the data preparations made in order to model this problem and the preliminary data analysis. We further describe the modeling process using each of the above mentioned techniques, and finally make the comparison between the results obtained for each model type.

4.1 Data Preparations

Before starting the modeling task we examined the structure of our existing data set, as used for the previous research question, and decided to make some changes and additions to best approach the modeling task at hand. These are detailed in the following subsections.

4.1.1 Additional Features

First of all we decided that the exercise scheme data, holding information about the exercises actually performed by the patient each day, is crucial and needs to be added to our relational schema. This decision was motivated by the nature of the current research question, dealing with perceived pain levels. We had a strong feeling that the exercises performed by the patient have some impact on their pain in subsequent days, and that specific exercises may trigger higher pain levels.

Adding this data to the relational schema required a first step of learning and understanding the content and structure of this data in the original MongoDB database. We learned that the data available for each patient for each day includes the list of exercises they performed, specifying their frequency (how many sets of a given exercise were performed per day) as well as their intensity (how many

Patient ID	DATE (Y-M-D)	Exercise 1		Exercise 2		...	Exercise 54	
		Freq.	Intn.	Freq.	Intn.	...	Freq.	Intn.
0	2017-01-01	3	2	0	0		0	0
0	2017-01-02	3	2	1	1		0	0
1	2017-03-01	0	0	0	0		2	2

Table 4.1: Exercises data structure

times each exercise was performed in a single set). The full list of existing exercises consists of 54 different possible exercises. In order to be able to conveniently feed this data to the modeling environments, the set of features per patient has to be identical, and thus cannot include for each patient only those exercises that they performed. Ergo, the data per patient per day was pulled out from the MongoDB and reconstructed in the relational schema such that for each patient for each day we have the full array of exercises, and their performance frequency and intensity, as presented in Table 4.1 (Intensity is denoted by Intn.). For each exercise that was not performed by a specific patient in a specific day, the value of zero was reported in the corresponding cell.

Second, we decided to add additional static features that we believed might also be influential for the current problem. We further detail these features and the way they were constructed in the following paragraphs.

One of the medical scores used in the system is the modified STarT Back Screening Tool (mSBT) [35]. This tool is composed of nine questions, used to divide patients with low back, neck, shoulder or knee pain to risk-subgroups, based on their physical and psycho-social factors taken pre-surgery. These subgroups represent the risk of persistent disability and/or elevated pain intensity, and are defined by low, medium and high risk. The mSBT has been proven to demonstrate predictive validity for long-term disability and pain outcomes [36], and thus we found it important to be included in this model.

While available, this data was excluded from the previous research question as it was missing for many of the patients and thus resulted in many null values. To overcome this obstacle we created three new binary, mutually-exclusive, calculated features, using the data obtained from the mSBT. The features are defined as *is_low_risk, is_medium_risk* and *is_high_risk*. Patients with missing data for this medical score are simply defined by a (0,0,0) feature vector, representing *unknown_risk*.

Additional static features include a set of 14 binary features, each representing a different sport activity (such as running, swimming, dancing etc.) and indicating whether or not the patient participated in each activity pre-surgery. In the same manner, an additional set of 14 binary features was added, for indicating whether or not the patient plans to engage in each of these sport activities after their recovery.

Besides the addition of new features to the dataset and in order to be able to use more of our available data, we decided to impute null values in the steps column. This was done by a linear interpolation, such that for each patient the interpolation for each null point (day with no reported steps) was done by the linear line passing between its two immediate neighbors (days with reported steps for that patient), one from each direction.

Finalizing this stage, our new dataset consisted of all previously used features (125) and additional 139 features. We next present the analysis that had been done on this dataset, in order to clean it, remove redundant features and fully prepare it for the modeling task.

4.1.2 Preliminary Analysis

We first separately analyzed the biggest group of new features consisting of 108 features indicating the intensity and frequency of each exercise performed by the patient each day. As this group includes many features and as we knew that each patient only performs two to three exercises per day, we were interested in understanding the frequency of occurrences of each of the exercises in the full dataset. This was computed, for each exercise, as:

$$\frac{\#\ of\ rows\ in\ which\ the\ exercise's\ frequency\ >\ 0}{Total\ \#\ of\ rows\ in\ the\ dataset}$$

The exercise occurrence frequency in the dataset versus the exercise index is shown in Figure 4.2. We found that ten of the defined exercises had never been done by any patient and thus can be omitted from the dataset. We further discovered that there is a small number of exercises which have been performed in 40% of the cases, while about half of the exercises have been performed in less then 5% of the cases. This analysis revealed the sparsity of the sub-matrix containing the exercise data (illustrated in Table 4.1), with a majority of elements at zero.

Checking correlations between all the exercise-related features, we found that for most of the exercises (65%) there is a correlation of 85% or above between the

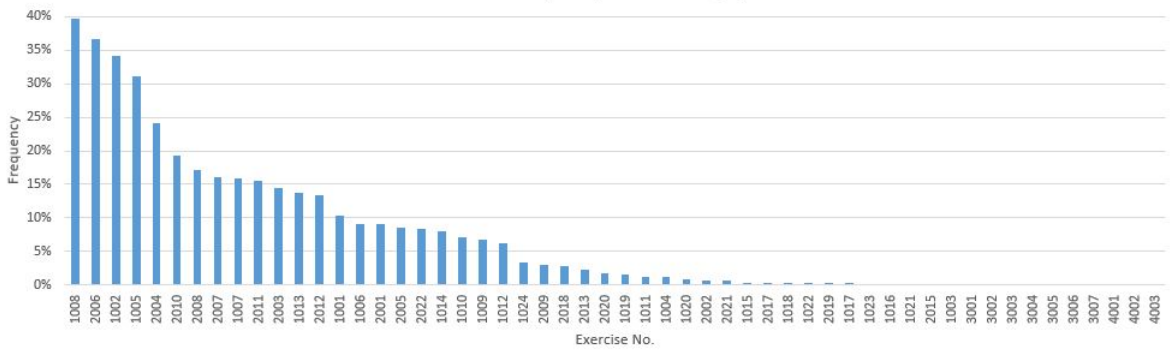


Figure 4.2: Exercises occurrence frequency in the dataset

frequency and intensity of the exercise, meaning that a change in frequency and in intensity for most of the exercises are usually performed together. As we prefer to keep the dimensionality of the problem as low as possible, we thus decided to only use the features describing the frequency of the exercises and leave out the intensity ones. At this point we remained with 44 exercise-related features out of the original 108.

In order to deal with the sparsity of this data, we decided to divide these 44 features into three groups, representing the level of the exercise (easy, medium, hard) and to additional four groups representing the type of exercise (basic strength, functional strength, mobility, reduction of symptoms). This way we are introducing only seven new features to the model instead of 44. These features are all binary, indicating for each patient for each day whether they performed an exercise from each of these groups. Noteworthy, these seven features are not mutually exclusive, as a patient can perform different exercises, belonging to different groups at a given day.

In the same way, we grouped the five medication features, reporting the daily dosage taken of five different medications, to three binary medication groups, based on their strength and purpose. In addition, the 13 daily activity features (e.g. driving a car, walking, going to work etc.) were grouped to three groups, based on their activity level (easy, medium, tough).

We then examined the correlations in the resulting dataset. All feature pairs demonstrated a correlation of below 85%, such that no feature was removed. Regarding the correlations between each of the features and the target, the most correlated features (with a correlation of only 50%) were: pain during day, pain during night, last three days average of pain during day, last three days average of

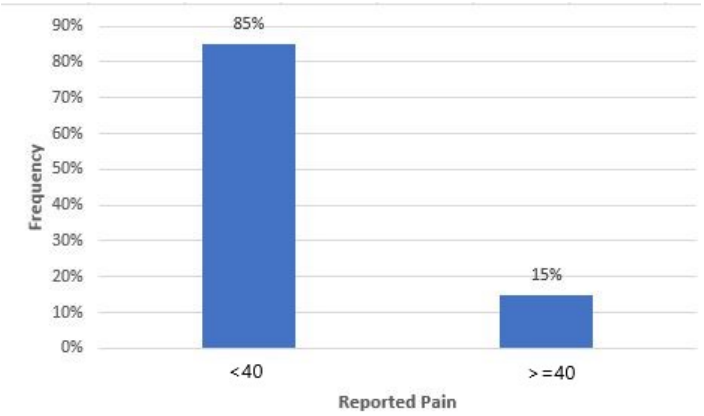


Figure 4.3: Distribution of reported pain levels

pain during night, pain during exercises, and general feeling. This result is quite logical, indicating that the pain level reported each day is correlated with the level of pain reported in the preceding days.

Lastly, the distribution of the target variable was computed, revealing a very unbalanced problem, as presented in Figure 4.3. As this is the case, we keep using the performance measure of bACC for comparing between different models and between different model configurations.

To sum, after the described data preparations and after null removal, we were left with a dataset including the data of 1000 patients, consisting of 30,500 observations of 179 features, to be used as the basis for the modeling task.

4.2 Decision Trees

In general, and specifically for the discussed research question, our goal is to find the key features which will maximize the efficiency of mechanisms to prevent patients deterioration. In this subsection, we discuss exactly the methodology to approach this point efficiently and to demonstrate a methodology using a decision tree. Namely, we show how to perform an analysis to reduce overclouding and noise in the data and to deduce the key features which can be reacted upon to change the trajectory of patient deterioration by treatment.

The approach taken for building the dataset for the decision trees was similar to prior sections, in order to allow the sequential information to be incorporated into

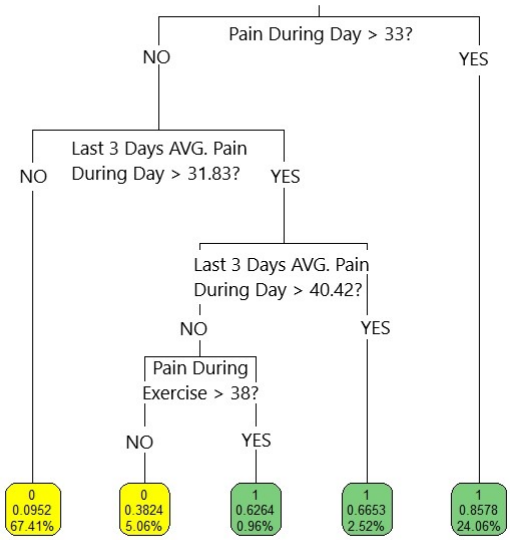


Figure 4.4: Classification tree for predicting next day substantial pain

the model. Namely, for each of the sequential features, its last three days average, and both the difference and the ratio between the current point and the last three days average were included.

For tuning meta-parameters, we applied the same protocol of 10-fold CV as elaborated in Subsection 3.2, on a training set consisting of 80% of the current data set.

The final tree was built using a minsplit value of 15 and a CP value of 0.03, and is presented in Figure 4.4. We can see that the tree uses only three unique features: pain during current day, last three days average pain during the day and pain during exercises in current day. These features chosen by the tree are completely aligned with our preliminary analysis, in which we reported these three features among the short list of features most correlated with the target. The bACC of this tree on the test-set was 86%.

Despite the relatively high bACC value obtained, we found this model to be quite disappointing, as it is only using past pain to predict future pain, thus not giving much insight to the care takers. That is, it is logical that high pain levels in the present are followed by high pain levels in the near future, and we were searching for the underlying causes for the pain. Furthermore, the motivation for trying to predict pain levels in the first place was to be able to control the pain and avoid its elevation, by making changes to the treatment protocol. These include changes in

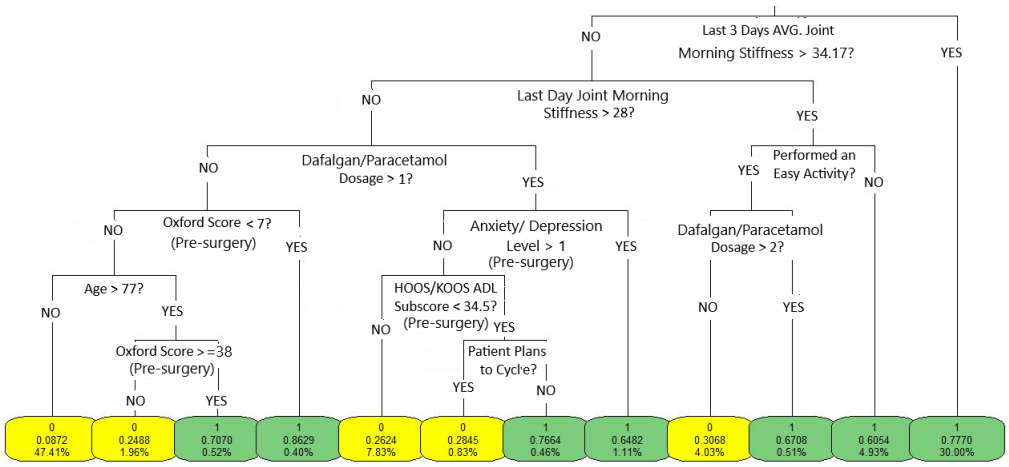


Figure 4.5: Classification tree for predicting next day substantial pain without using pain features

medication dosages, daily activity of the patient and daily exercise plan. For this reason, we were hoping to get some insight from the tree regarding factors (other than pain) which influence pain levels and which can be controlled and reacted upon.

To try and further achieve our goal, we then eliminated all pain-related features from the dataset and rebuilt a tree to predict next day substantial pain with all the remaining features. The resulting tree was built with the same values for the minsplit and CP meta-parameters, and is presented in Figure 4.5.

This tree is indeed much more insightful, and teaches us that pain levels are highly affected by the morning stiffness experienced in the index joint, by the intake of Dafalgan/ Paracetamol and by the daily activity performed by the patient. Another interesting insight is that some static features of the patient (e.g. age and PROM scores taken pre-surgery) also have an effect on the reported pain levels, as presented further down in the tree. This is especially interesting as in the current PT protocol, most of the features appearing in this tree are usually not examined, such that this tree indeed provided added value as to defining which features should be looked into when trying to control pain levels.

In addition, The bACC of this tree on the test set was 80%. This means that by excluding the pain features from the model, we are gaining additional insight, while only losing about 6% of explanatory power.

Trying to further improve the obtained results, we next describe the steps taken in order to build a classifier using a recurrent neural network.

4.3 Recurrent neural networks (RNNs)

In this subsection we first layout the basic notions which relate to modeling using neural networks and the vocabulary used in the rest of the subsection. We follow by elaborating on our model and on specific design selections and characteristics which relate to our prediction problem (Subsection 4.3.1), and finally we discuss our results in Subsection 4.3.2.

ANNs [37] are automatic data processing systems vaguely inspired by the biological neural networks that constitute the human brain. Such systems *learn from examples* without being implicitly programmed with any task-specific rules. An ANN is based on a collection of connected units called artificial neurons. Each connection, like the synapses in a human brain, can transmit a signal from one artificial neuron to another. Once such a signal arrives to the neuron, the neuron can process it and then signal additional artificial neurons connected to it.

In common ANN implementations, the output of each artificial neuron is computed by some differentiable non-linear *activation function* (typically the sigmoid, rectified linear or hyperbolic tangent) of the weighed sum of its inputs. Each connection between two units has a weight that adjusts as learning proceeds. Initially, the weight values are set randomly and the network processes data examples to provide its prediction and update the value of the *loss function*. The Loss function is used for measuring the inconsistency between the predicted value $\hat{Y}$ and the actual label Y. The weights change according to the desired strength of the signal at a connection and in order to minimise the total value of the loss function.

Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from

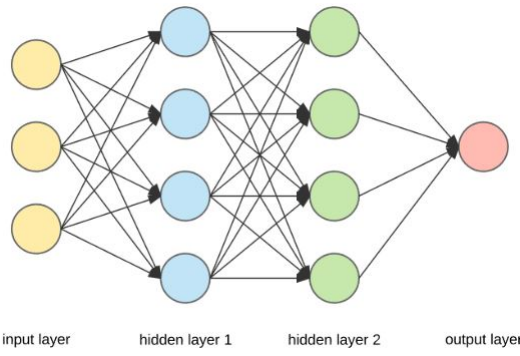


Figure 4.6: Artificial Neural Network illustration

the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

Modern deep learning tools and environments provide a very powerful framework for supervised learning. Deep Neural networks [38] are networks built from numerous inter connected neurons, placed in numerous layers, such that they are able to represent very complex functions. Figure 4.6 [39] illustrates such an ANN, getting as input three features which are subsequently being processed by two hidden layers (with four units each), and finally outputting one value. The inner layers are referred to as *hidden layers* as the training data does not show the desired output for each of them. There exist different types of ANNs, each optimized for different types of tasks and input data.

RNNs are a specific family of neural networks designed for processing sequential data. The main characteristic of RNNs is the fact that some of their neural units have a loop connection, such that at each time step, the neuron's input is the output of that same neuron in the preceding time step along with the current input, as illustrated in Figure 4.7 taken from [40]. This allows information to persist, such that past data as well as current data are taken into account in the prediction task.

One implementation of an RNN uses Long short-term memory (LSTM) units. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. This makes LSTM networks well-suited for classification, processing and prediction based on time series data, since there can be lags of unknown duration between important events in a time series.

RNNs had already proven to be very successful in predicting patient clinical events based on the patient's sequential clinical history [1, 19, 20]. Inspired by these researches, we decided to use an LSTM RNN in order to predict for each patient whether their next day pain level will be substantial, given their historical sequence. In the following we will detail our implementation as well as our design choices, and report our final results.

4.3.1 Modeling

As the RNN inherently takes into account the sequential nature of the data, we first removed from our current dataset the entire set of calculated fields reporting last three day average and distance from average of all the daily reported features.

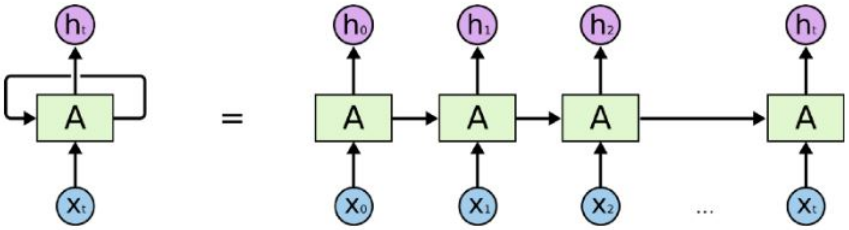


Figure 4.7: Recurrent Neural Network illustration

The LSTM RNN implementation was done in Python, using the Keras environment [41]. This modeling environment offers many different configurations, such that there are many design choices to be made. As the configuration space size is too large to explore (e.g. the number of hidden layers, the number of units in each layer, the activation function used by the units, the loss function to be minimized etc.), we decided to fix some of the meta-parameter values to commonly used and recommended values, and explore a grid of values for the others.

The first challenge we were faced with is deciding what would be the depth of our network, i.e. how many layers it should include. While being an active issue in the research community, there are no definite guidelines as to how this should be decided, such that usually a trial and error approach is the best approach. Generally, two layers have shown to be sufficient to detect more complex features. The addition of layers beyond two, has shown to improve the results for complex functions, but comes on the expense of computational complexity and a much longer training duration.

As a result, we decided to begin with comparing between a RNN structure of two and three hidden LSTM layers and asses performance. As the deeper network did not show better results, we finally decided to use a RNN with two hidden layers of LSTM units followed by a single Dense hidden layer (regular fully-connected NN layer) and finally a single unit output layer. The output of the network is a single value between zero and one, representing the probability that the target value of a given input vector belongs to the positive class. See Figure 4.8 for an illustration of the RNN structure.

For implementing this model, it is first required to pre-process the data in order to create sequences of the desired length per patient. As we have no prior knowledge regarding how many days of data should be taken into account in the prediction, we treated the size of the sequence as a meta-parameter to be trained. After consulting with the PTs, we chose to examine sequence sizes of 3, 5, 7 and 10 days.

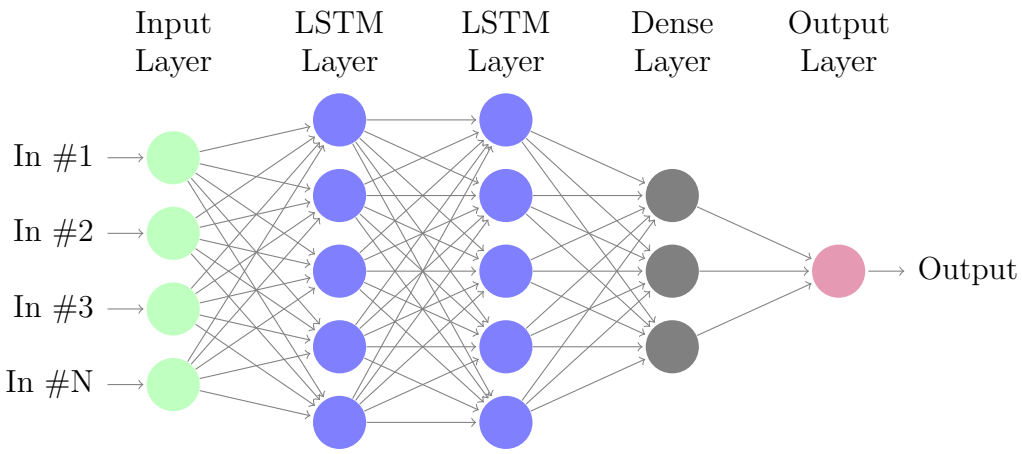


Figure 4.8: RNN Structure

In addition and before building the sequences, it is necessary to scale each of the columns in the dataset. As each group of features comes from a different numerical range and scale, scaling is important in order to prevent relatively large input values from slowing down the learning process and convergence of the network. In some cases, unscaled data can even prevent the network from effectively learning the problem altogether.

Regarding the scaling method to be used, we examined both common options: standardization and normalization. Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is zero and the standard deviation is one. This is done by subtracting from each item the column's mean and dividing the outcome by the column's standard deviation, $(X - \mu_X)/\sigma_X$.

On the other hand, normalization is a rescaling of the data from the original range so that all values are within the range of zero and one. This is done by subtracting from each item the column's minimal value and dividing the outcome by the difference between the column's maximal and minimal values, $(X - X_{min})/(X_{max} - X_{min})$.

Standardization assumes that the observations fit a Gaussian distribution with a well behaved mean and standard deviation. For this reason, it is not always the best scaling decision. However, standardization sometimes behaves well enough even if this assumption is not met, and might even over perform normalized data. For this reason we decided to check both scaling approaches as a second meta-parameter.

The third and final meta-parameter to be tuned is the number of units in the hidden layers. The options here are endless and there are no specific guidelines as to what should be done. The most common rule-of-thumb on this topic suggests that the optimal size of the hidden layer is usually between the size of the input and

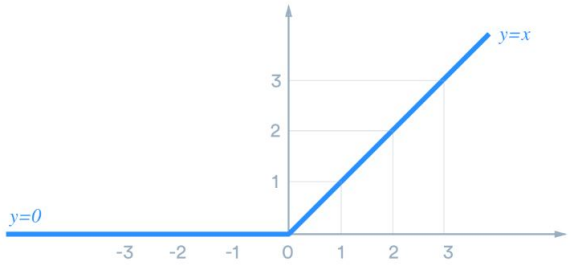


Figure 4.9: Rectified Linear Function

.

the size of the output layers. In our setting, the input consists of 123 features and the output size is one. Given this rule of thumb and common RNN configurations, values of 32, 64, and 128 units were examined. Note that the number of units affects the training duration, such that the more units in the network, the more connections and connection weights that need to be optimized, and thus the longer the training process.

The activation function chosen for the units in the hidden layers was the rectified linear function. This piecewise linear function is defined by $f(x) = max\{0, x\}$, such that it outputs the input directly if it is positive, otherwise, it outputs zero (see Figure 4.9 [42]). This activation function is a very common choice for the hidden layers in many types of neural networks because it is less computationally expensive than other well-known activation functions (e.g. tanh and sigmoid), such that it simplifies the training process, as well as often achieving better performance.

For the output layer, a softmax activation function was chosen. The softmax is often used as an activation function in the final layer of a classification model as it normalizes the outputs for each class between zero and one, and divides by their sum, thus giving the probability of the input value belonging to a certain class.

The loss function was chosen to be cross-entropy. This function is very widely used for measuring the performance of a classification model whose output is a probability value between zero and one. The cross-entropy is calculated by:

$$CE = -y \cdot log(p) - (1 - y) \cdot log(1 - p)$$

Where y is the actual label of the observation and p is its predicted probability. Cross-entropy loss increases as the predicted probability diverges from the actual label. It penalizes both types of mis-classification errors, where the penalty increases drastically as the wrong predictions become more confident. This is illustrated

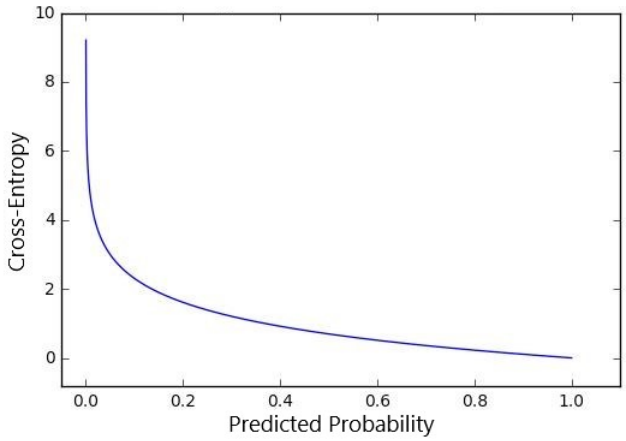


Figure 4.10: Cross-entropy per predicted probability when true label is 1

.

in Figure 4.10 taken from [43], which presents the decrease in the cross-entropy's value as the predicted probability approaches the true label of the observation (which is labeled as 1).

The technical meta-parameters for the training process were kept fixed, such that the number of iterations of going over the data (epochs) was set to 10, and the batch size was set to 64. In order to deal with the great class imbalance reported previously (see Figure 4.3), *class_weight* of 1:5.75 were defined for balancing the proportion between the negative and positive classes in the training process. Using this parameter, the loss function becomes a weighed average, such that the additional loss value of each sample is multiplied by the weight of its true class label. In our case, the negative class is present $\sim$5.75 times more than the positive one, hence the weights. By utilizing this parameter in the training process, the model treats both types of mis-classification errors as equally important, and avoids predicting for the majority class.

For the meta-parameter tuning, a 1-fold CV protocol was chosen. The dataset was primarily divided into a 90% training set and a 10% test set. Within the training set, we split the data to 80% training and 20% validation. For each combination of meta-parameter values, the model was trained once on the training data (holding 80% of the full training set) and the bACC, calculated on the validation set, was saved aside for later analysis. The meta-parameter values used in the RNN configuration with the highest validation performance were then chosen for the construction of the final model.

Noteworthy that the CV protocol used here has two differences from the one used so far. Firstly, the fact that we use 90% of the data for training and not 80% as done previously. This was done as deep learning networks are known to achieve good results when there is a substantial amount of data to be trained on. As this is not our case, we wanted to use as much data as possible in the training process, while still leaving some data for testing. Secondly, the decision to apply a CV protocol that only uses 1 fold. The reason for this is that the training process of a RNN is very time consuming. While training a RNN, the dataset is being ran-over multiple times (epochs) for minimizing the loss function. Hence, the difference in time between applying a 1-fold CV and a 10-fold CV is immense.

We realize that the implications of this approach, however, is that we expose our model to the danger of over-fitting to the training set. In order to try and avoid that, every LSTM layer was followed by a Dropout layer. This layer helps to prevent over-fitting by ignoring randomly selected neurons during training, and hence reduces the sensitivity to the specific weights of individual neurons. The proportion of random neurons to be ignored was set to 20%, as it represents a good compromise between model accuracy achieved and the prevention of over-fitting.

In addition, we monitored the bACC values obtained during the meta parameter tuning process. We compared the bACCs obtained for training, validation and test (when computed) and noticed that they were all very similar (per model configuration), implying that there was no significant over-fitting.

4.3.2 Results

In all the different models trained, the ones using a standardized scaling of the data always performed better (on validation) in comparison to the parallel ones based on the normalized data. In addition, the meta-parameter tuning process yielded that the best performing configuration is achieved with a sequence of length five with 64 units in each of the hidden LSTM layers and 16 units in the hidden dense layer. The validation bACC of this model was 83%. This configuration was used in order to train an RNN on the whole training set (including 90% of the full dataset). The bACC of the final model on the test set was 84.88%.

Trying to further improve the results of the RNN, we examined closely our data and searched for a change that would enhance the model's performance. Although RNN is specialized for processing sequential data, we found evidence suggesting that static features may not always be processed by it in the best way. Furthermore, these evidence suggest that the static features should be processed separately, e.g. by a feed-forward NN. The outputs of the latter together with the RNN output (processing only the dynamic information) should then be combined in order to obtain best results [20].

Model	bACC
Decision Tree - full data	86%
Decision Tree - w/o pain features	80%
RNN - full data	85%
RNN - w/o static features	86%

Table 4.2: Model performance comparison

We leave as an interesting open-challenge to implement this full solution, however, suspecting that the static features might be hurting the model's performance, we decided to remove all static features from the dataset and rerun the meta-parameter tuning process again on the new dataset. This time the tuning process resulted in a best performing configuration achieved with a sequence of length seven with 128 units in each of the hidden LSTM layers and 32 units in the hidden dense layer. The validation bACC observed was 85%. It should be noted that in general, the bACC values seen in the meta-parameter tuning process of the dataset without any static features, were slightly higher then the ones observed on the initial full dataset. Once again, this configuration was used in order to train an RNN on the whole updated training set. The bACC of the final model on the test set was 86.18%.

We were expecting the RNNs to achieve significantly better results than the ones obtained using the trees, what was evidently not the case. In the following we compare between the different models, analyze the results and try to understand what led to the perceived behaviour.

4.4 Model Selection

Table 4.2 summarizes the test performance of the 4 models described in this chapter. The two RNN models had a very similar final performance, which was also similar to the performance achieved by the best tree. As was the case in the previous chapter, the quantitative performance of the models does not allow for a clear choice between them, such that other criteria need to be taken into account.

First of all we examine the interpretability of all models. The trees constitute a very straight forward model, that can be easily implemented and understood. It is very clear which features were chosen as most significant by the tree, and this allowed us to further experiment and analyze the behaviour of the tree without certain features. This model can be easily analyzed by clinicians, which can provide

clinical evidence as to why the features chosen are indeed significant in treatment and recovery.

The RNNs on the other hand, supply a model which is a complete "black box". The weights obtained for each of the connections can be easily extracted from the model, but this information is practically useless in a model including as many features as we have. i.e. it is extremely difficult to analyze over 30,000 weight values, in order to extract from this information the significance of each of the original features given as input. For a given neural network, the best way to understand if a feature is indeed significant is to remove it from the data and observe how it affects the model's performance. This method was indeed used here for examining the static features' importance, and the results obtained show that they have no significant value in the model, as removing them results in similar bACC.

As we were disappointed from the relatively poor results obtained by the RNN models, we were aspiring to understand the cause for this. One strong assumption we had, is that neural networks are very sensitive to the size of the training data provided to them, in the sense that they are able to achieve very impressive results when the data is big enough.

In order to test this hypothesis, we decided to re-build one tree and one RNN design, with growing amounts of data, and analyze their change in performance on a separate fixed test set, as the number of training data rows becomes larger. For the tree, we chose the configuration of the best performing tree out of the two, and for the RNN we chose the configuration without the static features, as it showed slightly better results throughout the whole modeling process.

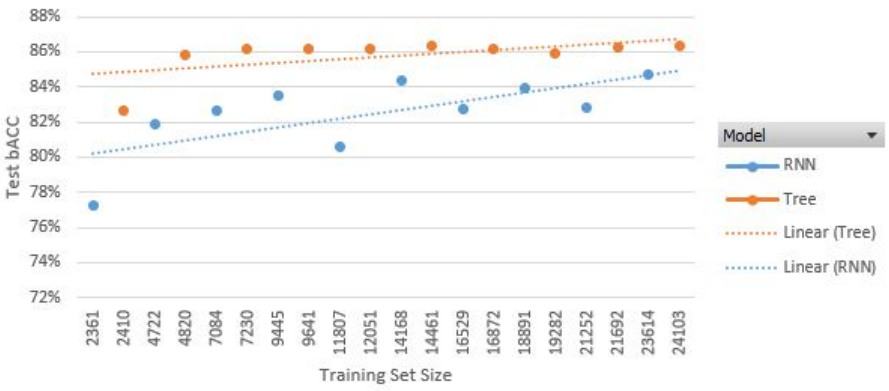


Figure 4.11: Test bACC per training data volume on a tree Vs. RNN

.

Figure 4.11 presents the results of this experiment. Each point on the graph represents the test bACC obtained for a model trained with the number of rows indicated in the x axis. The dashed lines represent the linear trend of each of the models examined. We can clearly see that while the bACC convergence in the tree's case happens quite early on (around 7000 rows), such a convergence is not clearly viewed for the RNN. Moreover, we see that the linear trend in the tree's case is much smaller than the RNN's, again indicating that the bACC of the RNN had not yet converged, and that having yet more data to train on, is likely to result in better performance.

To conclude this chapter, we were able to answer positively the research question, i.e. we were able to predict next day pain levels while utilizing both tools. The tree classifier allowed for a deeper analysis and understanding of the results and the key features. On the other hand, the RNN provided comparable results, however, its test bACC as a function of training data size seems to converge slower than that of a tree, leaving room for increased performance on larger data scales. Nevertheless, with very limited interpretability.

Chapter 5

Conclusion

The contribution of this thesis is in making the first steps in a wide research activity, which utilizes AI and ML techniques for predicting patient outcomes and using this information for adjusting the treatment protocol. Essentially, the goal is to increase both the efficiency of the clinical team and the patients' satisfaction.

Specifically, in this research we aimed to build efficient predictors for manual intervention occurrences and for pain levels, in patients recovering from an orthopaedic surgery. We explored the efficiency, accuracy and interpretability of different ML models, applied to a set of collected patient data.

Broadly, the three main challenges that we faced were: (1) how to handle patient data, which is irregularly structured, contains many missing entries, is very noisy and holds many different data types, (2) how to build new features from existing ones, in order to reduce the noise in the data while also lowering the problem's dimensionality, and (3) how to correctly make use of different ML techniques, such that we can validate, interpret and understand the outcome/model.

Starting from the necessary step of data extraction, we built our flat dataset, holding daily reportings consisting of over 400 different features from several different sensors (wearable device, daily questionnaires, periodical medical scores, exercise scheme and chat messages). Then, after studying, understanding and cleaning the data, different statistical techniques were used in order to both build new features from existing ones and to select the most relevant features for each of our prediction tasks. Data cleansing and analysis methods, including but not limited to, feature distribution plots, Pearson correlation, data scaling and data imputation were used in order to create the narrowest, fullest and most appropriate dataset for each prediction task and ML model.

The core of the research dealt with how to (correctly) apply and train decision trees, SVMs and RNNs for obtaining high performing classifiers. The models built were carefully analyzed and tuned in order to extract maximum information about the most significant features and to be able to translate the results into recommendations for the clinical team.

The main outcomes of this research indicate that for predicting manual intervention, both decision trees and SVMs achieve the same level of prediction balanced accuracy, which did not improve even when using an ensemble model. We found that for the most part, the models were in agreement regarding the most significant features and regarding the classification itself. Similarly, when predicting substantial pain levels, we discovered that both decision trees and RNNs achieve the same level of bACC, however, with some dependence on the dataset size. The analysis indicated that the RNN's bACC is probable to keep rising with the growth in data volumes.

In both cases, we observed that decision trees provide a model which is much more interpretable. This is useful both during the modeling process, for verifying the model's logic and discovering errors, and later, for building the understanding of the clinical team and gaining their trust.

In the following, we list several important challenges and interesting future work for our research:

The current research for predicting pain levels leaves room for additional work, for improving the RNN performance in different aspects. First, as was mentioned in the RNN results (Subsection 4.3.2), future work could include the implementation of an ANN, processing the static and the dynamic features separately and then combining them for producing the output, and comparing the results to the ones obtained in the current work. Second, as was discussed in the subsection presenting the model selection for pain level prediction (Subsection 4.4), the performance of the RNN is likely to increase with a significant increase in data volumes. With the current data growth rate we are expecting the data to double its size within eight to ten months. It will be, then, very interesting to re-run the current RNN models, analyze their results, and asses whether their performance keeps increasing as expected.

Moreover, the time and efforts spent for constructing the full dataset and for building an automatic process, allowing for new data to be added as it arrives, resulted in a robust infrastructure that could be exploited in the future for many different research questions and prediction tasks, saving substantial setup time. In particular, we believe that an interesting and relevant research could focus on the prediction of milestones for a patient. i.e. predicting when a patient would be able

to perform certain tasks, stop taking medications etc. This would be very beneficial for the patient, allowing to reduce the uncertainty surrounding such a surgery, as well as setting realistic expectations for the patient, and increasing patient satisfaction. In addition, we would encourage future research aiming at building a personalized treatment plan per patient. This plan would automatically change over time, according to patient specific parameters observed. This challenging task includes first identifying those feature which are critical for the personalized treatment and using them in order to model the connection between their patterns and the required treatment changes.

It is extremely important to note, that the research conducted and models built as part of the overarching activity, focus on helping, recommending, and making the PT team's work more efficient, and the patient rehabilitation process smoother and quicker. They, however, by no means aim at replacing the clinical team. Thus, when incorporating these insights into the application, a great emphasis should be put on separating between the actual evidence collected from the patient, and the recommendations coming from the ML models. Namely, the prediction models should not trigger an automatic change in the treatment protocol of the patient, but should rather produce an alert for the PT, indicating that something should be changed or requires special attention.

To conclude, we strongly believe in applying AI and ML techniques in the medical domain, for making the clinicians' work more efficient and for improving patient outcomes. It is evident that the research community is increasingly dealing with these kinds of problems, focusing on building robust and high performing models, which are also able to provide fair interpretability. These challenges are of utmost importance, and we trust that medical applications will increasingly incorporate such models and techniques in the foreseeable future.

Bibliography

[1] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*, pages 301–318, 2016.

[2] Lorenzo Moja, Koren H Kwag, Theodore Lytras, Lorenzo Bertizzolo, Linn Brandt, Valentina Pecoraro, Giulio Rigon, Alberto Vaona, Francesca Ruggiero, Massimo Mangia, et al. Effectiveness of computerized decision support systems linked to electronic health records: a systematic review and meta-analysis. *American journal of public health*, 104(12):e12–e22, 2014.

[3] Enrico Coiera and Enrico Coiera. Guide to medical informatics, the internet and telemedicine. 1997.

[4] Manisha Bahl, Regina Barzilay, Adam B Yedidia, Nicholas J Locascio, Lili Yu, and Constance D Lehman. High-risk breast lesions: a machine learning model to predict pathologic upgrade and reduce unnecessary surgical excision. *Radiology*, 286(3):810–818, 2017.

[5] Robert K Lindsay, Bruce G Buchanan, Edward A Feigenbaum, and Joshua Lederberg. Dendral: a case study of the first expert system for scientific hypothesis formation. *Artificial intelligence*, 61(2):209–261, 1993.

[6] BG Buchanan and EH Shortliffe. The MYCIN experiments of the Stanford heuristic programming project. *Reading, MA: Addison-Wasley*, 1984.

[7] William J Clancey and Edward H Shortliffe. *Readings in medical artificial intelligence: the first decade*. Addison-Wesley Longman Publishing Co., Inc., 1984.

[8] Randolph A Miller. Medical diagnostic decision support systems—past, present, and future: a threaded bibliography and brief commentary. *Journal of the American Medical Informatics Association*, 1(1):8–27, 1994.

[9] K-P Adlassnig. A fuzzy logical model of computer-assisted medical diagnosis. *Methods of Information in Medicine*, 19(03):141–148, 1980.

[10] James A Reggia and Yun Peng. Modeling diagnostic reasoning: a summary of parsimonious covering theory. *Computer methods and programs in biomedicine*, 25(2):125–134, 1987.

[11] William G Baxt. Use of an artificial neural network for the diagnosis of myocardial infarction. *Annals of internal medicine*, 115(11):843–848, 1991.

[12] Philip S Maclin, Jack Dempsey, Jay Brooks, and John Rand. Using neural networks to diagnose cancer. *Journal of medical systems*, 15(1):11–19, 1991.

[13] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[14] Hian Chye Koh, Gerald Tan, et al. Data mining applications in healthcare. *Journal of healthcare information management*, 19(2):65, 2011.

[15] Farhad Soleimanian Gharehchopogh, Peyman Mohammadi, and Parvin Hakimi. Application of decision tree algorithm for data mining in healthcare operations: A case study. *International Journal of Computer Applications*, 52(6), 2012.

[16] N Christianini and J Shawe-Taylor. Support vector machines and other kernel-based learning methods, 2000.

[17] C Venkatesan, P Karthigaikumar, Anand Paul, S Satheeskumaran, and R Kumar. Ecg signal preprocessing and svm classifier-based abnormality detection in remote healthcare applications. *IEEE Access*, 6:9767–9773, 2018.

[18] Jerald Yoo. On-chip epilepsy detection: Where machine learning meets patient-specific healthcare. In *SoC Design Conference (ISOCC), 2017 International*, pages 146–147. IEEE, 2017.

[19] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016.

[20] Cristóbal Esteban, Oliver Staeck, Stephan Baier, Yinchong Yang, and Volker Tresp. Predicting clinical events by combining static and dynamic information using recurrent neural networks. In *Healthcare Informatics (ICHI), 2016 IEEE International Conference on*, pages 93–101. Ieee, 2016.

[21] Victoria A Brander, S David Stulberg, Angela D Adams, R Norman Harden, Stephen Bruehl, Steven P Stanos, and Timothy Houle. Ranawat award paper: Predicting total knee replacement pain: A prospective, observational study. *Clinical Orthopaedics and Related Research®*, 416:27–36, 2003.

[22] Prabha Susy Mathew and Anitha S Pillai. Big data solutions in healthcare: Problems and perspectives. In *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on*, pages 1–6. IEEE, 2015.

[23] Theresa Weldring and Sheree MS Smith. Article commentary: Patient-reported outcomes (pros) and patient-reported outcome measures (proms). *Health services insights*, 6:HSI–S11093, 2013.

[24] Matthew Shardlow. An analysis of feature selection techniques. *The University of Manchester*, pages 1–7, 2016.

[25] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 3121–3124. IEEE, 2010.

[26] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14.2, pages 1137–1145. Montreal, Canada, 1995.

[27] Paul E Utgoff. Incremental induction of decision trees. *Machine learning*, 4(2):161–186, 1989.

[28] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420, 1997.

[29] T Santhanam and Shyam Sundaram. Application of cart algorithm in blood donors classification. *Journal of computer Science*, 6(5):548, 2010.

[30] Drakos Georgios. Towards data science: Support vector machine vs logistic regression. *Medium, Available online: https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f*, 12 August 2018.

[31] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[32] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2018. R package version 1.7-0.

[33] Nicola Crichton. Visual analogue scale (VAS). *J Clin Nurs*, 10(5):706–6, 2001.

[34] European Medicines Agency (EMA). Guideline on the clinical development of medicinal products intended for the treatment of pain. *Committee for Medicinal Products for Human Use (CHMP)*, 2016.

[35] Katie A Butera, Trevor A Lentz, Jason M Beneciuk, and Steven Z George. Preliminary evaluation of a modified start back screening tool across different musculoskeletal pain conditions. *Physical therapy*, 96(8):1251–1261, 2016.

[36] JC Hill, EK Afolabi, M Lewis, KM Dunn, E Roddy, DA van der Windt, and NE Foster. Does a modified start back tool predict outcome with a broader group of musculoskeletal patients than back pain? a secondary analysis of cohort data. *BMJ open*, 6(10):e012445, 2016.

[37] Marcel van Gerven and Sander Bohte. *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018.

[38] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[39] Arden Dertat. Towards data science: Applied deep learning - part 1: Artificial neural networks. *Medium, Available online: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6*, 8 August 2017.

[40] Suvro Banerjee. Towards data science: An introduction to recurrent neural networks. *Medium, Available online: https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912*, 23 May 2018.

[41] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[42] Danqing Liu. A practical guide to relu: Start using and understanding relu without bs or fancy equations. *Medium, Available online: https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7*, 29 November 2017.

[43] Loss functions. *Machine Learning Cheatsheet, Available online: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html*, 2017.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**