MASTER'S THESIS | LUND UNIVERSITY 2017

Predicting and Analyzing Osteoarthritis Patient Outcomes with Machine Learning

Per-Victor Persson, Hans Rietz

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2017-10

Predicting and Analyzing Osteoarthritis Patient Outcomes with Machine Learning

Per-Victor Persson
dat12pp1@student.lu.se
mail@pvpersson.se

Hans Rietz
dat12hri@student.lu.se
hans.rietz@gmail.com

22nd June 2017

Master's thesis work carried out at Arthro Therapeutics.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Erik Rehn, erik.m.rehn@gmail.com

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

The use of machine learning as a tool in medicine is increasing and has provided new avenues for research into a number of diseases. Creating better predictive models for these diseases could provide opportunities for better care, which we have applied to osteoarthritis, a degenerative disease that affects a large part of the older population. We have sought to answer "Is it possible to predict patient outcomes?" and "What factors contribute to the patient outcome?" by constructing and evaluating machine learning models. In order to do this, a dataset containing 75,366 patients who have participated in an osteoarthritis treatment program was used and analyzed. The selection of models included neural networks, logistic regression, and gradient boosting machines among others in order to capture the performance of several types of machine learning models. Our results show that it is possible to predict patient outcomes on a test set with 60% accuracy.

Keywords: MSc, machine learning, neural networks, osteoarthritis

Acknowledgements

We would like to thank both our supervisors Pierre and Erik who were invaluable in providing feedback during our work. We would also like to thank Arthro Therapeutics for the opportunity, the whole team working there for taking care of us and also in particular Leif Dahlberg who helped us better understand osteoarthritis and osteoarthritis care. Finally we would like to thank Ludwig Andersson at Registercentrum who helped provide us with the data and get a grip on it.

Contents

Chapter 1
Introduction

1.1 Background

Osteoarthritis (OA) is a frequently occurring joint disease affecting a quarter of all Swedes over the age of 45, or almost 1 million people (Turkiewicz, 2016) but younger people can also be affected. Symptoms of OA include joint pain as well as stiffness of the joints resulting from a breakdown of joint cartilage and the underlying bone (Poole, 2012).

The national guidelines in Sweden (Socialstyrelsen, 2012) prescribe exercise, information, and weight reduction as the best treatment for OA. However, it is not uncommon for patients to receive surgery for pain relief, though 10-30% of the patients do not show any improvement or display a worsened condition afterward (Desmeules et al., 2013).

Additionally, only a minority of the patients who receive surgery have seen a physiotherapist or occupational therapist at any time before surgery. These two facts prompted the creation of the initiative "Better management of patients with OsteoArthritis" (BOA) in 2008. The aim is to reduce sick leave and improve rehabilitation of patients with osteoarthritis by following the national guidelines, promoting exercise and providing information to the patients (BOA-Registry, 2015).

The BOA initiative also includes a national quality registry to ensure the quality and results of the program. Several metrics are evaluated using a standardized set of questions: the quality of life, health, physical activity, and level of pain are all among the included metrics.

Simultaneously, the advances in machine learning have created an opportunity for better quantitative models in health care (Miotto, Li, Kidd & Dudley, 2016). A usual approach is for a domain expert to specify patterns and clinical variables to analyze, as well as constructing the models. Machine learning models can bypass this by automatically creating representations and identifying latent patterns in the data. These models enable researchers who are not domain experts, for example in the field of osteoarthritis, to find these patterns and draw conclusions.

In this project, we have used the data from the BOA quality registry to create machine learning models, with the goal of predicting patient outcomes, i.e if the patient improved.

1.2 Problem Definition

To give the best care, identifying the patients who are most in need of intervention is desirable. We will investigate the possibility to predict the patient outcomes in a database of osteoarthritis patients using machine learning (ML) tools. With this, we will attempt to answer the questions:

1. Is it possible to predict the outcome?
2. What factors contribute to the outcome?

1.3 Related Work

Though BOA produces an annual report containing results from the medical units participating in the initiative, the analysis is primarily focused on the results of the program and between units.

As such we cannot compare our results with others when studying osteoarthritis predictions, though there have been similar approaches to ours for other medical conditions.

Most closely Weng, Reps, Kai, Garibaldi and Qureshi (2017) used machine learning methods to predict cardiovascular events. They used electronic medical records from 700 UK family practices containing 30 variables for 378 256 patients. Four machine-learning algorithms were used and compared to an established algorithm to predict cardiovascular events over ten years. They found the best performing algorithm (a neural network) correctly predicted 7.6% more patients compared to the established algorithm. Though our methodology is similar, we differ in the kinds of features used for the predictions, with the majority of our features being patient-reported and theirs measured from blood samples.

Similarly, Lee et al. (2014) analyzed and identified Health-Related Quality of Life (HRQoL) factors of the elderly with chronic diseases which they used to construct predictive models. With similar methodology, they measured HRQoL using the patients' EQ5D-index, which is present in our dataset as well, and the selection of machine learning algorithms to use in the predictive steps contained decision tree and random forest. With a dataset consisting of 716 patients, they achieved a final accuracy of 0.93 using stepwise logistic regression. Their models predicted the patient final EQ5D-index in two classes as above and below an absolute cutoff value while we predict an outcome relative to the initial value which may explain the higher accuracy they achieved.

The main difference between both of these studies and ours is the diseases studied along with the datasets used, which makes it interesting to see how the predictive capabilites of the models transfer between types of data and diseases.

1.4 Contributions

We worked together the entire time spent producing this thesis, collaborating on all parts.

Chapter 2
Fundamentals of Machine Learning

Machine learning has its roots in statistics, and also involves terms not commonly found in other fields. To fully understand the method used in this report, we recommend enjoying a quick rundown.

2.1 Prediction

To be able to make predictions, two things are needed, a model and the data on which to fit the model. Regarding the model, for making predictions on this type of data there are two approaches: classification and regression. **Classification** aims to, given a sample, determine a category to put it in. For example, given a patient, predict the outcome as improved, stable or worsened. For **regression** the output is instead a continuous value. An example of this would be that, given the same sample, the model predicts a value of 4.42. In our case, by comparing the predicted value to the original value we can also determine a category.

One advantage of using regression is that the models include the distance to the correct answer, in other words, the larger the error, the more the model would correct its predictions due to its loss function. This might seem like a reason to always use regression, but the classifier also outputs probabilities along with each class, which makes it easier to set a threshold to reduce the number of false positives.

2.2 Training

Fitting the model to data is usually referred to as **training** the model. It is important to note that training a model on a set of data and then using the same data to test the model will result in a overfit. Doing this is very likely to result in a model that archives a perfect score on the test data but, when faced with new unseen data it would not always make accurate

predictions, i.e., the model would fail to generalize, when this is the case the model is said to be **overfitting**. To avoid this lack of generalization, we withhold part of the data during training and use this data to test the performance of the models afterward. We call these two sets of data the training set (to train on), and the test set (to evaluate the final performance on).

When experimenting and evaluating model parameters, a situation can occur where these parameters improve performance on the test set. However, the act of continuously evaluating them on the test set leaks information on the data in the test set, and we also end up in a situation where the model has overfitted and does not work as well on unseen data.

There are two main solutions to this problem. The first is to split the data into three parts instead, a training set, a validation set, and a test set. This way the validation of the model parameters is performed on the validation set and final evaluation is done on the test set. However, this can reduce the number of available samples and which in turn may result in worse performance. The other solution is to perform cross-validation. The basic approach to doing this is to use k-fold cross-validation (CV), which splits the training set into k smaller sets. For each of these **folds**, the model uses the fold as validation set and the remaining $k - 1$ folds as the training set. The performance is then reported as the average over all the folds. Using cross-validation is more computationally expensive since we need to train the model k times, but the upside is that cross-validation wastes less data.

Since the models usually pose several requirements on the input, the raw data usually needs to be processed. One of the problems faced may be incomplete data, fields or columns may be missing due to any number of circumstances. We need to address this to prevent the model from inferring unwanted patterns from the distribution in this missing data, as well as improve performance with more samples. A simple strategy is to remove the offending features or samples, but this can exclude useful data. The process of filling in the missing data is known as **imputing**, and can also result in better performing models. Imputation can be done in several ways, by filling in the missing values with the mean, median, most frequent value (the mode) or a random sample selected from the rest of the set. For the models to perform optimally, the data also needs to be scaled to have unit variance and norm, done by scaling the data.

To evaluate the models, we need to have some evaluation framework, and most importantly metrics to use. The simplest metric to consider is the ratio of true to false predictions or the **accuracy**. However, consider the case where the data is imbalanced, and 70% of the data belongs to one class, a model that simply always predicts the dominant class would achieve 70% accuracy. So to gain more insight into the behavior of the model, we are also going to use the related metrics **precision**, **recall**, and **F-1 score**. **Precision** can be stated as "of the selected items, how many were relevant" and **recall** as "of the total relevant items, how many were selected". The F1-score is the weighted harmonic mean of precision and recall. Precision and recall can also be plotted against each other which gives us a **precision-recall curve**, and calculating the area under this curve gives us our **average precision**.

As we are using these binary metrics for multi-class problems, we are going to treat each class in the data as one binary problem. We are then faced with the issue of how to average or sum the performance of each class across the set of classes. The two most used strategies are **macro** and **micro**. The macro strategy simply calculates the mean of the

binary metrics, while the micro strategy sums all the false positives, negatives and true positives for the different classes to get the metrics. We are also going to plot the **receiver operating characteristic** (ROC) Curve, which is the true positive rate plotted against the false positive rate.

Finally, we analyzed the learning curve (Perlich, 2011), which shows how the behaviour of the model changes with the number of samples used.

2.3 Models

We evaluated a set of models, both a group of models implemented in Scikit-learn as well as a neural network we built ourselves. The models we used included logistic regression, decision trees, random forests, gradient boosting, adaptive boosting, and a multi-layer perceptron.

Logistic regression is a linear model for classification (despite its name), which fits data to a logistic function. This function follows a sigmoid curve which gives it its S-shape, given by the equation

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

For **decision trees** (Quinlan, 1986), the goal is to infer simple rules from the data, an example being "when $X > 3$, output: Improved". This results in models that are simple to understand and interpret since the model is just a tree of these rules. The downside is that the model can also create overly complex trees that are prone to overfitting, as well as show a bias towards the most dominant classes.

One solution to this is to use an ensemble method such as **random forest** or **gradient boosting**. **Random forest** (Ho, 1998) is an averaging method, which builds a forest of decision trees and then averages their predictions, creating a model with lower variance than the individual trees.

Gradient boosting (Friedman, 2001) on the other hand is a boosting method, where the idea is to combine several weaker models into a powerful ensemble by building them sequentially. In essence it performs gradient descent, attempting to find a local minimum by following the negative gradient of a function, on a arbitrary loss function. The implementation in Scikit-Learn uses several decision trees which are fit on the negative gradient of the deviance loss function.

Adaptive boosting (Freund & Schapire, 1995) is also a boosting model, though the two models differ in how they are trained. Adaptive boosting instead begins by fitting one classifier on the data, and then proceeding to fit copies of that classifier with modified weights on the incorrectly classified instances, this improves performance on the more difficult classes.

2.3.1 Neural Network Models

The basic principle behind a neural network is to stack layers of functions on top of each other, and during training, the model creates an internal representation of the relevant features. This overall architecture remains constant, though there are many types of neural

networks. A neural network can be split into three separate parts, the input layer, the hidden layers, and an output layer. The input layer simply passes the data on to the hidden layers. The layers following the input are referred to as the hidden layers, and these layers contain the logic in a neural network. For the model to return a useful result, it has a dedicated output layer that is responsible for translating the internal representation into a format that matches the desired output. A very simple illustration of this concept can be found in Figure 2.1.

Since the patient values both before and after the program are known, we can calculate the outcome as our ground truth and train the models using supervised learning. **Supervised learning** compares the predicted result with the known correct value, for example, a label of what an image depicts, what category it belongs to or an output value. In contrast, training can also be performed unsupervised, by instead providing the model with a function to be maximized instead of a ground truth.

2.3.2 Multi-Layer Perceptron

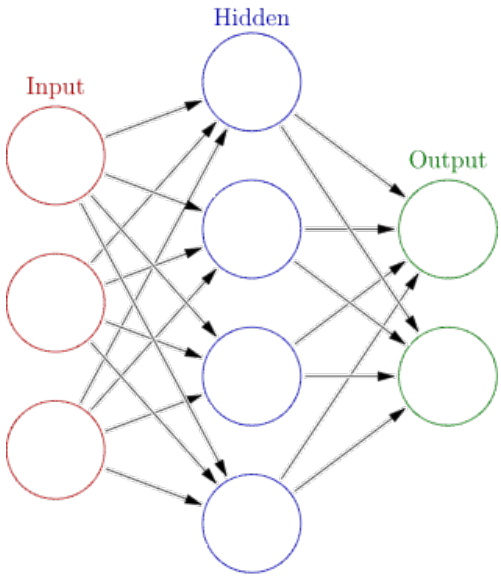


Figure 2.1: A two layer deep multi-layer perceptron (Wikimedia Commons, 2013)

A multi-layer perceptron (MLP) is a simple type of neural network that has three distinct features that make it simple but still powerful. It utilizes a **feed-forward** design, all the layers are **fully connected** with their respective previous one, and is uses **back-propagation** during training.

Feed-forward neural networks are simplest and most straightforward neural network architecture. In this kind of network, the information flow is restricted to the forward direction, compared to other network architectures which might have dependencies backward in the network or other exotic behaviors.

In a **fully connected** layer each node in the layer is connected to all the nodes in the previous layer and during training, weight is assigned to all the individual connections.

Backpropagation is a multi-step process used during training for calculating and updating the weights used in the neural network. When the training data has passed through the neural network, the result is compared to the expected value. The expected value is then propagated backward through the neural network, and a delta is calculated for all the weights based on the current weight and what it needs to be to produce the correct result. This delta is then in turn used by a **optimizer** function to update the weights optimally.

The **optimizer** is a function that is responsible for calculating the new weights with the overall goal of minimizing the loss value.

2.3.3 Additional Types of Neural Network Layers

In addition to the simple feed-forward layers previously described, more complex layer types have been constructed and used in several machine learning applications. These layers can incorporate memory or act as convolutional filters. There are also layers specifically designed to be used during training to prevent overfitting; one such layer is the dropout layer.

A **dropout layer**[1] (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014) provides regularization in the network and is special in the sense that it only affects the result during training. During training, the layer will drop units randomly with a probability p. Dropping a unit means that the node and all of its incoming and outgoing connections are temporarily removed, preventing the network from depending on only a few nodes to provide its predictive strength. A desirable effect of using dropout layers is that each time the layers are updated it constructs a new neural network within the existing one, and by doing this repeatedly during the training the resulting neural network is a combination of several smaller networks, which can improve performance.

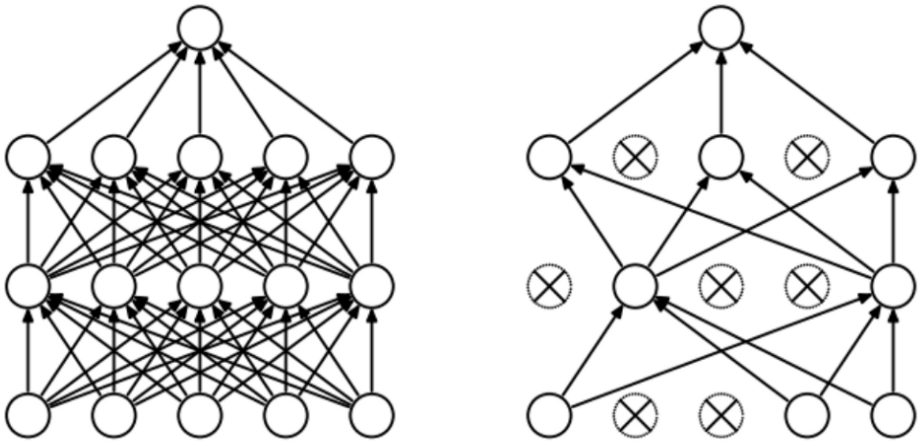


Figure 2.2: Dropout illustrated (Srivastava et al., 2014)

Batch normalization is also a sort of regularization layer and allow for higher learning rates to be used (Ioffe & Szegedy, 2015). During training it normalizes the input to the following layer and it does this for each batch.

[1] In Keras dropout is applied on a layer by layer basis.

2.3.4 Training Neural Networks

Training a neural network works very similarly to training the earlier models but requires a bit more time spent optimizing to get the best performance. Two parameters that we often needed to change were the **batch size** and the number of training **epochs**. Both of these concepts can be applied to the earlier models as well, but the increased computational complexity of the neural networks necessitates their inclusion.

By collecting the samples into batches before feeding them to the network, we can simultaneously decrease the memory needed for the computations as well as speed up training. The **batch size** controls how many samples the model will be trained on at a time, i.e., how many will be passed through the model before updating the model weights. That means that the model update is based on an average of all the corrections generated by the samples in a single batch.

The batch size can affect the result of the training greatly. A small batch size results in the weights being updated many times based on a few samples, and as a result, the model might converge quicker, but the weights will be updated more erratically. In contrast, bigger batch sizes led to the weights being updated fewer times but based on a more substantial number of samples. This can cause it to converge slower but also more stable as the new weights are a result of the average gradients of all the samples, which helps to filter out the noise in the data but can also lead to reduced performance for minority classes.

The number of **epochs** denotes the number of times the model has been trained on the entire training set. Tweaking this number can make the model overfit either more or less since it lets the model see the data several times.

We also used **oversampling**, a method for balancing imbalanced data sets (Estabrooks, Jo & Japkowicz, 2004), which consists of adding extra samples to minority classes in to even out the class distribution. There are many ways this can be done, including copying existing samples or by imputing.

2.4 Statistical Tools

For us to perform the validation of our assumptions, we also need some statistical tools.

A **hypothesis-test** is a common way to confirm the statistical significance of any results, or in other words asserting the certainty that the results are significantly not random, which we used to verify that our models learned real patterns.

Before performing the test, a threshold is selected for the level of statistical significance needed, this threshold is then compared with the **p-value** obtained in the test.

Student's t-test can be used to perform a hypothesis test to determine if two sets of data are significantly different. A simple One-sample t-test uses the statistic

$$t = \frac{\bar{x} - u_0}{s/\sqrt{n}}$$

where $\bar{x}$ is the sample mean, s is the sample standard deviation, n the sample size, and u_0 the value to compare with, in our case the sample mean of the other set. The mean is assumed to be normal, and the central limit theorem can be used in certain situations to

strengthen that assumption. To get the p-value simply look up the value for t at the required significance in a table for Student's t-distribution.

A **confidence interval** can also be constructed and used to perform a hypothesis test. Similarly to a threshold, a critical area is defined and compared with the result.

Though these tests can be used to distinguish a result from random noise when dealing with large sample sizes even a subtle difference in sample means can prove to be significant. **Cohen's d** is a measure of effect size (Sullivan & Feinn, 2012), which is commonly used to show the practical difference for populations by calculating the distance of the means in standard deviations[2]. Cohen also defined the guidelines shown in Table 2.1 for what constitutes a small, medium, and large difference.

d	Practical difference
>0.2	Small
>0.5	Medium
>0.8	Large

Table 2.1: Guidelines for Cohen's d

[2] A good visualization is available at http://rpsychologist.com/d3/cohend/

Chapter 3
Method

Our approach and method is very similar to established data mining methodology such as CRISP-DM (Wirth, 2000) and can be split into several phases.

- Data understanding
- Data preparation
- Modeling
  - Model comparison
  - Model optimization
- Evaluation
  - Model understanding
  - Statistical Validation

3.1 Tools

We used several libraries and tools during the process, which decreased the amount of time spent on implementation and freed up more time for experimentation with different models.

We stored our dataset in **SQLite**, an open source database having good interoperability with the other tools.

The primary tool used to create and evaluate the models was **Scikit-Learn** (Pedregosa et al., 2011). Scikit-Learn contains many built-in models and utilities, for everything from data pre-processing to evaluation and visualization.

However, the facilities included in Scikit-Learn for designing and building neural networks are not very comprehensive and do not utilize the GPU for a computational speedup,

so instead, we used Keras for building neural networks. **Keras** is an open-source library that aims to provide high-level abstractions for neural network layers and concepts. Keras is then able to interface with a number of backends that provide the lower level building blocks to model the neural networks and perform the computations, of which we used TensorFlow. **TensorFlow** is also an open-source library, built at Google to provide the low-level building blocks for neural networks. A major advantage for TensorFlow is the fact that it is GPU accelerated, resulting in the feasibility of training larger networks or iterating faster on smaller networks.

3.2 Data Understanding

Understanding the data includes studying the dataset and its patients' characteristics. As previously mentioned the data used is from the Swedish quality registry BOA "Bättre omhändertagande av patienter med artros", the file we received included answers for 75,366 patients in total. We received this data as an Excel document, which we then imported into an SQLite database so that querying and analyzing the data would be easier.

This data is collected using forms filled in on three occasions, before the program (First Visit, FV), a checkup three months directly after the program (M3), and one year after the program (Y1). The forms include questions for the patient and also a small section for the physiotherapist. A description of the questions in the set can be found in Table 1 in the appendix. The majority of the questions are answered on each occasion, though demographic data (Age, BMI, Sex) is only collected on First Visit, and for M3 questions are added for the physiotherapist regarding the patient's participation in program activities.

These questions primarily answer:

- Which of the patient's joints are affected?
- The amount of pain the patient is experiencing
- Health Related Quality of Life for the patient
- The patient's level of physical activity and exercise
- Does the patient want, is in the progress of getting, or has received surgery to treat their joint?
- Is the patient taking any medication, and if so which one(s)?

In order to gain an understanding of the data, we generated histograms to visualize the distributions of each feature in addition to calculating the mean, median and variance for the different features, taking note of any missing data.

We also noted that the answer rate is 65% after three months, and it falls to a bit below half the total after one year, as shown in Table 3.1. Since the program is still ongoing, some patients have not finished their three months yet, though that does not account for all of the missing answers. Using the BOA annual report and looking through the dataset we selected the features shown in Table 3.2 to use as indicators for improvements in health. These indicators measure patient pain, quality of life, physical activity and joint related

	Answers	Percentage of total
Total Patients	75365	
3 months	49283	65%
1 year	34157	45%

Table 3.1: Answer overview

Metric		Answers	Improved	Stable	Worsened
EQ5D Index	3M	36568	24.6	66.7	8.64
	1Y	23742	22.4	66.1	11.5
EQ5D VAS	3M	40093	19.5	62.7	10.4
	1Y	24977	18.0	61.1	14.1
Pain Frequency	3M	48196	34.4	53.6	12.0
	1Y	33360	38.0	47.7	14.3
NRS	3M	48932	56.9	22.9	20.2
	1Y	33385	52.2	20.5	27.3
Fear Movement	3M	48751	12.9	84.2	2.9
	1Y	33425	11.8	83.5	4.6
Exercise Weekly	3M	40547	43.6	32.8	23.5
	1Y	25975	37.4	32.6	30.0
Physical Activity	3M	40606	36.0	34.1	30.0
	1Y	26079	32.4	33.2	34.5
Drugs	3M	47765	18.7	74.4	6.8

Table 3.2: Indicator metrics overview, values as percentages

drugs. These features are also present on all three occasions (FV, M3, Y1) which we need to predict future values.

We narrowed our focus to four of these indicators to preserve time and effort, and we are going to use the three-month answers as our ground truth for the patient outcomes. Though there is also the one-year follow-up data, after an initial comparison we found it similar to the three-month data in addition to restricting the number of patients available for training if used. Due to this, we decided to only perform predictions for the three-month outcomes.

- EQ5D Index, an index for overall life-quality, on a scale from 0-1
- EQ5D VAS (Visual Analog Scale), a patient-reported scale, also for overall life-quality, on a scale from 1-100
- Pain Frequency, a scale answering the question “How often does your joint hurt?” with never, every month, every week, every day, always.
- NRS (Numeric Rating Scale), a 1-10 scale for the amount of pain experienced in the joint marked as most painful.

When categorizing the values into Improved, Stable and Worsened we used the clinical definition of change found in the annual BOA report.

- For the EQ5D Index we counted an improvement when the index changed at least 0.1 units.
- For EQ5D VAS a change of 15 units, which is a clinical improvement was required.
- For pain frequency and NRS we used one unit on each particular scale.

3.3 Data Preparation - Pre-processing and Feature Engineering

Preparing the data for the models required a few simple sets.

We split the dataset into a training and a test set. For the test set, we randomly picked 25% of the patients. When modifying and searching for optimal parameters we used five-fold cross validation on the training set.

Additionally, we imputed the missing data filling in the missing data with the mean value for each feature. We scaled the data to $0 \ldots 1$ by dividing by the max absolute value for each feature. Later, while optimizing the models we experimented with different imputation and scaling strategies, though we found the best performance using the mean for imputation and $0 \ldots 1$ as endpoints for scaling.

3.4 Modeling

We ended up comparing five different models, all are included in Scikit-learn and all used the default parameters.

- Logistic regression
- Random forest
- Adaptive boosting
- Gradient boosting
- Multi-layer perceptron

Using only the first visit data to predict the three-month result after the program gave us the results visible in Table 3.3. The models had overall similar performance, both in regards to the different models and the different metrics. Only random forest had 100% training accuracy, while the training accuracy of the other models are lower but remain ahead of the test accuracy. This discrepancy could be an indicator to increase the complexity by adding more features or changing the model parameters.

The accuracy of the models looked to be between 60%-70% for the best performing model, which was gradient boosting and made it a good candidate to optimize further.

Model	Metric	Train	Accuracy	Precision	Recall	F1	ROC AUC	Avg Prec
Logistic R.	EQ5D Index	0.742	0.705	0.532	0.404	0.393	0.715	0.511
Random F.		1.000	0.703	0.438	0.494	0.463	0.724	0.531
Adaptive B.		0.749	0.670	**0.645**	0.517	0.498	0.722	0.533
Gradient B.		**0.759**	**0.689**	0.619	**0.524**	**0.509**	**0.745**	**0.558**
MLP		0.752	0.607	0.545	0.399	0.368	0.660	0.489
Logistic R.	EQ5D VAS	0.672	0.668	**0.700**	0.436	0.422	0.744	0.543
Random F.		**1.000**	0.662	0.622	0.430	0.417	0.724	0.527
Adaptive B.		0.673	0.665	0.560	0.455	0.453	0.717	0.512
Gradient B.		0.693	**0.674**	0.601	0.460	0.458	**0.750**	**0.554**
MLP		0.708	0.653	0.535	**0.467**	**0.462**	0.719	0.510
Logistic R.	Freq	0.579	0.570	0.547	0.438	0.449	0.701	0.511
Random F.		**1.000**	0.582	0.562	0.418	0.412	0.703	0.510
Adaptive B.		0.582	0.578	0.543	0.455	0.457	0.672	0.488
Gradient B.		0.611	**0.594**	**0.564**	**0.483**	**0.495**	**0.724**	**0.535**
MLP		0.627	0.573	0.543	0.454	0.467	0.695	0.503
Logistic R.	NRS	0.590	0.591	0.395	0.427	0.407	0.708	0.500
Random F.		**1.000**	0.578	**0.586**	0.415	0.397	0.689	0.483
Adaptive B.		0.593	0.590	0.526	0.431	0.422	0.674	0.466
Gradient B.		0.614	**0.596**	0.580	**0.436**	**0.429**	**0.716**	**0.513**
MLP		0.627	0.575	0.513	0.428	**0.429**	0.691	0.482

Table 3.3: Performance of three month result prediction, best performance per model and metric is highlighted

F1-scores and average precision scores were low overall, owing mainly due to bad recall. The best F1-score and accompanying average precision score belonged to predicting EQ5D with gradient boosting, achieving a F1-score of 0.509 and average precision of 0.558.

After picking gradient boosting to use as our baseline, we needed to search for better parameters and try to optimize its performance characteristics. To do that we had several tools at our disposal.

First, we tried changing the feature representations to less noisy and better ones. We used **recursive feature elimination** with cross-validation on the training set (RFECV) to obtain a smaller feature set with higher performance. An addition we also tried applying a dimensionality reduction algorithm to combine highly correlated features. We also experimented with different strategies for imputing the missing data, scaling and normalizing it.

Second, we searched the parameter space using both a grid search and a randomized search, which yielded slightly improved performance.

Since we thought that regression models might result in lower errors due to the way far off predictions are penalized more we also tried to train regression models. These models did not show a significantly different initial performance, as well as losing prediction probabilities as an output from the models which led us to instead try to optimize the classification models using other techniques instead. We found that applying principal component analysis as dimensionality reduction did not improve the performance for any of the metrics, the same was true for RFECV due to the low amount of features eliminated. Since gradient boosting is based on decision trees, this already provides a robust way to capture non-linear dependencies between features, and we do not gain much of the benefit from either of these two methods. Experimenting with imputation and scaling likewise did not result in any substantial gains.

3.5 Neural Networks

After having explored and optimized the more traditional models, we wanted to determine if it was also possible to get similar or better scores using a neural network.

From our the four indicators we decided to only include the pain value (NRS) in this stage since we mainly wanted to know if there is any drastic difference when applying more advanced neural networks.

3.5.1 Challenges

There are several challenges with designing and training a neural network. The models that we studied earlier were somewhat limited in the number of hyper-parameters that could be changed, though this had the benefit of limiting the number of parameters that needed to be tested to be able to find the optimal parameter set. Neural networks are generally more challenging to tune correctly in practice since each layer and component that make up the network all have a set of parameters to tune leading to a combinatorial explosion of possibilities to test.

Ideally, we would try to find the optimal parameters each time when we changed the architecture or layout of the neural network, but this turned out to be unfeasible. Because a neural network is computationally heavy on its own adding the time required to search the parameter space during every training became unfeasible on our machines. Instead, we focused on experimenting with the internal structure and only when we were happy with the performance of the structure did we perform a grid search of the parameters.

3.5.2 Initial Design

As we had to start somewhere, we used a multi-layer perceptron structure similar to the one included in scikit-learn as a starting point in the design process.

Most of the work done at this stage went into figuring out what designs and parameters worked reasonably.

When starting out with the design process, we settled on using a batch size of 128 and that the training period should last for 30 epochs.

The initial design used **fully connected** layers, which are referred to as **dense** layers in Keras and the activation function **ReLU** or **rectified linear unit** which is a very simple mathematical function. A ReLU layer simply takes the output from the last layer and sets any value that is negative to zero according to $f(x) = max(0, x)$ before passing it on to the next layer.

3.5.3 Optimizer

To be able to train the neural network we needed to select an optimizer for the training process. We made the initial choice from among the seven optimizers that are included in Keras, and the choice was made based on how well they performed when trained on a very basic neural network with three dense layers each with ReLU but without any regularization.

To get the best performance possible without regularization, we added an early-stop- and checkpoint callback to the training process. The early-stop callback allowed us to stop the training process automatically when it started to overfit to the training data and the checkpoint callback was used to store a copy of the model at each epoch, this allowed us to load the model from the specific epoch when it performed the best.

The results were not very conclusive due to overfitting despite the precautions we had taken with the callbacks. The networks often performed well on some metrics but worse on others, which meant that there was no clear winner. Despite the varying performance we settled on using the optimizer **adam**.

This initial choice ended up being not very important, as the optimal optimizer would change later as the architecture of the neural network changes during the experimentation. Because the relatively short training times, it is possible to test other optimizers at a later stage easily.

3.5.4 Class Weights and Class Imbalance

As seen in Table 3.2 the class distribution for **NRS 3M** is very uneven, and we found that this greatly affected the result of the training. Early models tended to exhibit behaviors

where they either only predicted a single class or only stable and improved. It had many difficulties learning to predict if the patient would worsen. To counteract this behavior, we decided to specify a set of custom weights for the loss function.

That means that during training, the model is encouraged to get better at predicting the minority classes by increasing the punishment for getting them wrong. To calculate the weight for each class, we initially used the inverted fraction of the training data that the class represented according to:

$$w_{\text{class}} = \frac{\text{Total number of patients}}{\text{Total number of patient belonging to class}} \tag{3.1}$$

This had the advantage of giving a weight that was directly correlated to how large the class is compared to all the other. A downside with this approach was that during training the class 'worsened' got a weight that was a magnitude greater than the rest. That resulted in models that always predicted worsened and as such did not solve our initial problem.

It was possible to get the model to perform better using these weight calculations, but it required that we manually scale down the calculated weights. This worked, but it was not a desirable method as the scaling factors were dependent on that specific data set and the current behavior of the model.

To get class weights that didn't require manual tuning, we decided to scale the value by the largest size class instead of the size of the data set according to:

$$w_{\text{class}} = \frac{\text{Size of the single largest class}}{\text{Total number of patient belonging to class}} \tag{3.2}$$

This resulted in class weights that were much closer to each other, which helped to increase the performance. However, there is a limit to how much that can be done with only class weights, they helped when using small batch sizes, but as we tried to increase the batch size, the models returned to the old behavior of only predicting stable and improved or only worsened.

We assumed that this was because there were too few patients in the batch that got worse, and that the large batch sizes caused them to get lost when calculating the average update. In an attempt to solve this we implemented what is called **oversampling** to balance the training set. It is important to note that we did this after the validation set was split from the training data, thus preventing it from containing the same data as the training set.

An interesting consequence of this method of balancing was that the models now tended to perform better without any custom class weights at all, and we ended up removing them from the training.

3.5.5 Training

As there are no clear cut rules governing how to design a neural network for a specific problem, we had to spend a lot of time experimenting. This was mostly a trial and error process which involved both much manual tuning but also comprehensive grid searches of different designs. It was a very iterative process which involved making small changes to the neural network, and after that trying to minimize or prevent overfitting. If the model continued to performed worse than it did before, then the changes were reverted and documented. On some occasions, this process resulted in a neural network that grew too large

Layer names	
dense_21	Layer size: 8
activation_11	Activation: ReLU
dropout_11	Dropout rate: 0.1
dense_22	Layer size: 3

Table 3.4: Final neural network layer parameters

and complex, when that happened we started over by removing as much as possible while still retaining the core architecture that worked earlier.

To try and find which architectures that worked well but without having to try them all manually we implemented a **randomized grid search** that over several iterations randomized a neural network design and then trained it. This allowed us to identify what designs worked better than others.

The final neural network design is shown in Figure 3.1 and Table 3.4

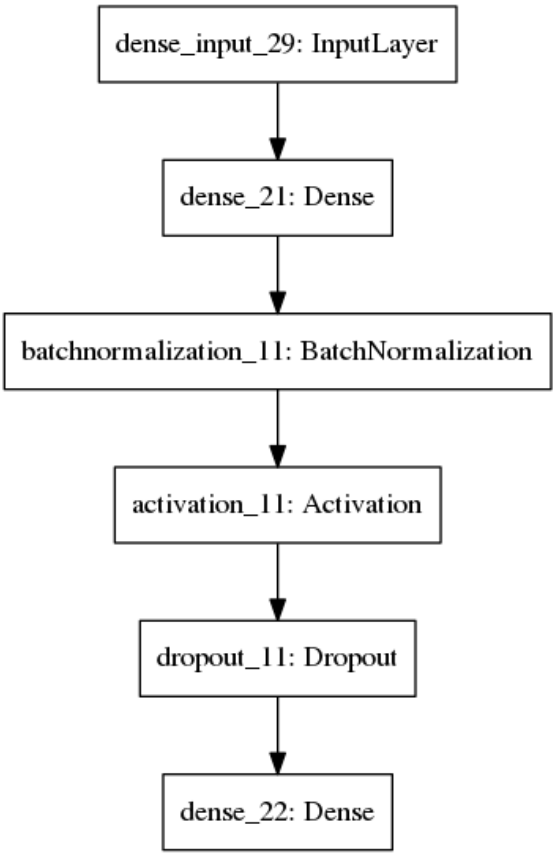


Figure 3.1: Final neural network design as Keras layers

3.6 Model Understanding

Understanding the how and why of the models is important for several reasons. We wanted to validate that the models are capturing the interesting phenomena, that can be actionable in practice. Also, it can provide a possibility to create simpler models with the same characteristics, or improve the performance of the model.

With our main goal being able to understand which features are the most important and what feature interactions are present we used three methods: calculating the feature importance and comparing them with each other, plotting partial dependency plots and tree diagrams for decision trees.

Also, we also tried recursive feature elimination, but the features that were left after elimination were very closely related to the highest ranked features from our feature importance rankings, which did not provide any new information on the interactions between features.

3.6.1 Feature Importance

To understand which features the models based their predictions on, we calculated the feature importances shown in Table 3.5. This table shows the top 15 features ranked by their importance as given by the models. From this we can see that the most important feature for predicting the outcome is the initial value for each of the indicators.

The next features are the other indicator variables as well as Age, Height, and Weight. The location of the worst joint seems to be more important for the joint pain frequency than the other indicators, but the other values are similar for all four models.

3.6.2 Approximation by decision trees

Since the different models performed similarly during our comparison, we attempted to approximate the behavior of our Gradient Boosting classifier using a simple Decision Tree and analyze its behavior instead.

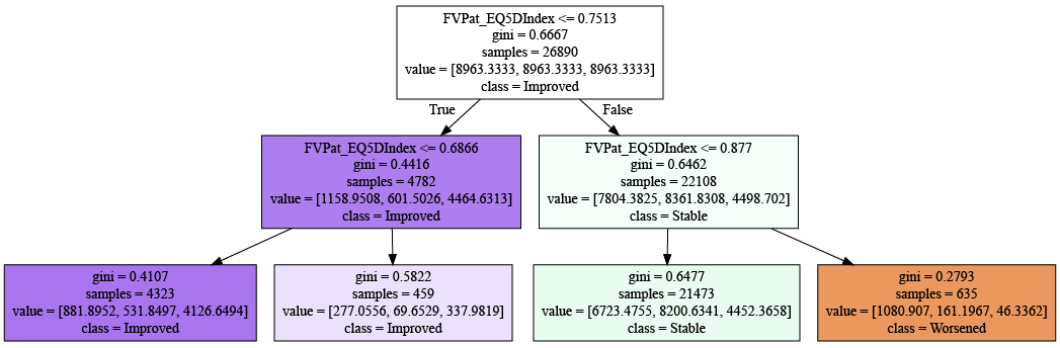


Figure 3.2: Decision tree for EQ5D Predictions

We found that we could approximate our Gradient Boosting model with a simple decision tree for EQ5D to a high degree. A diagram of the decision tree is visible in figure 3.2.

	EQ5D Index		**VAS**	
1	EQ5DIndex	0.300	EQ5DVAS	0.309
2	PainJointNRS	0.057	EQ5DIndex	0.050
3	EQ5DVAS	0.050	Weight	0.041
4	Age	0.040	BMI	0.040
5	BMI	0.039	ASESSymptomScore	0.039
6	PainJointFreq	0.035	PainJointNRS	0.038
7	ASESPainScore	0.031	Age	0.033
8	ASESSymptomScore	0.026	Height	0.027
9	UCLA	0.025	ASESSymptom1	0.026
10	Height	0.023	ASESPainScore	0.024
11	ASESPain2	0.023	CompareActivity	0.023
12	Weight	0.022	ASESPain2	0.017
13	WorstJoint	0.021	WorstJoint	0.017
14	CompareActivity	0.019	PainJointFreq	0.016
15	Hip	0.014	EQ5DPain5L	0.015

	PainJointFreqFreq		**NRS**	
1	PainJointFreq	0.234	PainJointNRS	0.266
2	Age	0.059	EQ5DIndex	0.062
3	PainJointNRS	0.052	PainJointFreq	0.055
4	EQ5DIndex	0.041	BMI	0.044
5	BMI	0.038	EQ5DVAS	0.035
6	EQ5DVAS	0.035	Age	0.033
7	ASESPainScore	0.035	ASESPainScore	0.032
8	Height	0.033	Height	0.027
9	Weight	0.031	Xray	0.024
10	WorstJoint	0.029	Weight	0.024
11	ASESPain3	0.026	XRayOA	0.024
12	XRayOA	0.021	WorstJoint	0.022
13	ASESPain1	0.017	EQ5DPain5L	0.019
14	ASESSymptomScore	0.017	ASESSymptom1	0.018
15	ASESPain5	0.017	Hip	0.017

Table 3.5: Feature importances for three month predictions

From this tree, we can see that the model simply picks a threshold of 0.751 and predicts improvement for patients below this threshold and stable above. Interestingly it also includes a second higher threshold of 0.877 for which it predicts worsened for those above it, which seems to indicate that the patients EQ5D results show a return to the mean.

However, we found that this strategy did not work for the other indicators as even with high tree depths the performance remained below that of our baseline model while the size of the tree prohibited easy inspection.

3.6.3 Partial Dependency Plots

From the feature importances it is possible to gain some insight into what features are important to the models, but it does not explain how the models react to changes in the features. In order to determine these dependencies we used **partial dependency plots**. These plots try to marginalize the effects of all other features except for the one being examined while iterating over different values and approximating the probabilities that each class is predicted.

Partial dependency plots for 8 of the most important features when predicting the NRS value, can be found in Figures 3.3, and 3.4.

In Figure 3.3 we can see that there is no significant change in the probability of any class being predicted for Age, Weight or Height, though there is a slight increase in the probability of being predicted as worsened when the BMI increases.

Next, in Figure 3.4 we can see a higher dependency on the features. In the graph for EQ5D Index we see that the probability of the NRS value remaining stable or become worsened will decrease if the EQ5D Index is larger than 0,43. The graph for EQ5D VAS shows a smaller dependency than the other three, though still larger than Age or Weight. The third graph for NRS itself is the most interesting, showing that the more pain[1] the patient was experiencing before the program the higher probability for improving. The final graph shows that the probability of improving decreases as the frequency of the pain increases, with the other two classes showing a small increase at the tail end.

[1]NRS is inverted compared to the other three, with a larger value implying more pain.

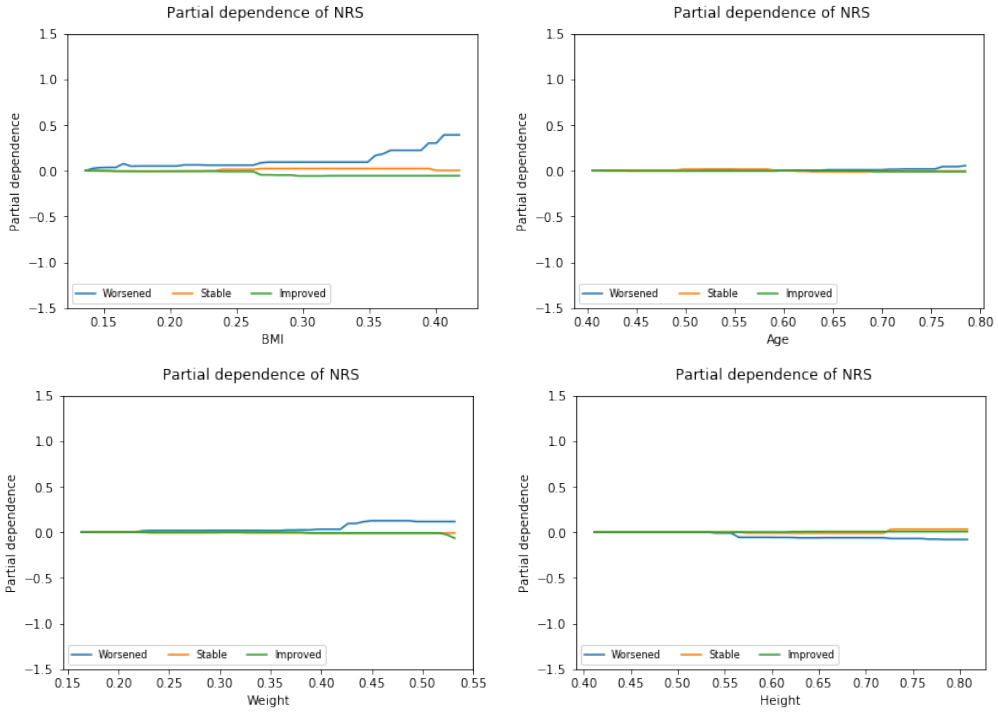


Figure 3.3: Partial dependency plots for Age, BMI, Weight, Height

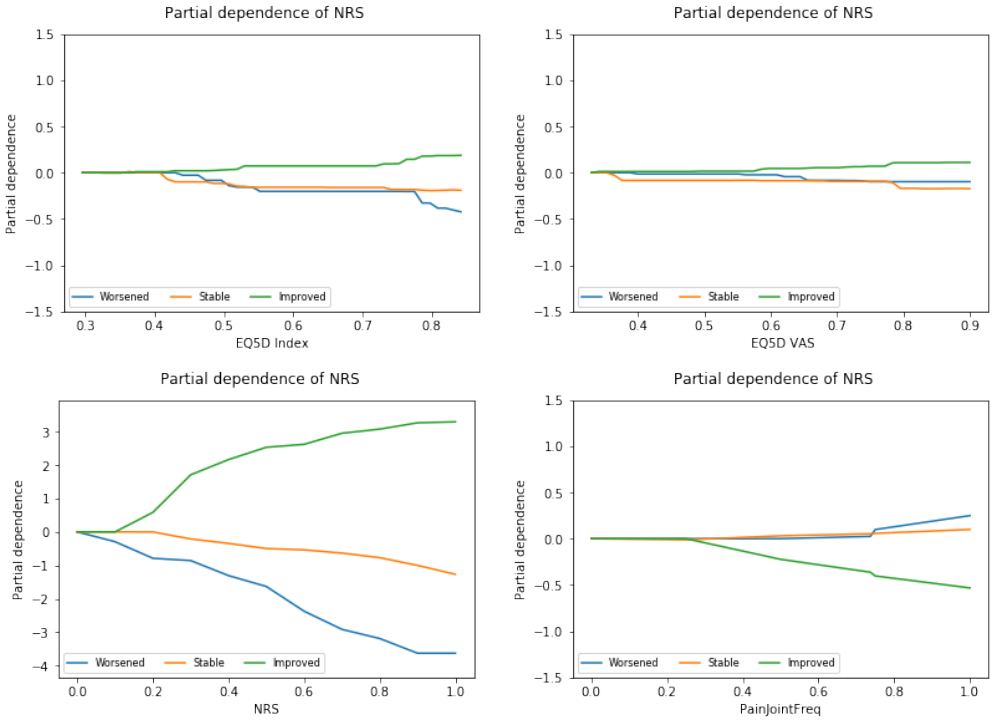


Figure 3.4: Partial dependency plots for EQ5D, VAS, NRS and pain frequency

3.7 Statistical Validation

To validate our understanding of the models, we decided to perform a more traditional statistical analysis of the dataset.

We choose simple tools to do the analysis hypothesis testing, confidence intervals and studying the effect size as Cohen's d. Due to the size of the dataset the statistical power is also very large meaning resulting in very small Standard Errors and p-values. Due to this, we decided to mainly look at the effect size as an indicator of significance. Also, we also decided to study the effects of training, naturopathic drugs, and hip/joint pain separately.

We did this by first studying the influence of the Boolean questions on the starting value for NRS. For each Boolean feature, we split the patients into yes/no groups and calculated a p-value and Cohen's d on the difference in means between the two groups. The results are visible in Table 3.6, the table includes the features where $p < 0.05$, and are sorted by effect size. A positive effect size in this table means that a yes correlates with a higher NRS value, i.e., more pain. A larger value for d in the table corresponds to a larger difference in pain for the two groups, for example the patients who answered yes on 'DesireSurgJoint' are in much more pain than the patients who answered no when using Cohen's guidelines for the difference.

From the results, we can see that a desire for surgery correlates positively with a higher pain value. This is also true for patients who are on medication, afraid to move or on sick leave due to their joint pain. Interestingly this also applies to patients who have previously received information on losing weight, though the effect is slight.

Next we instead looked at the change in NRS after the program, still filtering the table by $p < 0.05$ and sorting by effect size. A positive effect size means a reduced value for NRS, which means less pain and an improvement, and vice versa for a negative effect size.

These results are visible in Table 3.6, and while some them fall below $p < 0.05$ the effect sizes are also small. For many features, Cohen's d falls below the limit for a small effect.

The most interesting features are the continuous ones, especially the starting value for NRS and EQ5D since they were the most important features in the models as well. We calculated the effect size for all the continuous features by splitting the patients into two groups, the patients who responded with a value above the mean of each feature in the dataset, and the patients who responded below.

We also note that one question with a large effect size is XRayOA, if the patients have been X-rayed and the x-ray indicated arthritis, with patients displaying less improvement, had it done so. Comparing this with the feature importances we obtained previously, we note that XRayOA was also placed among the important features, indicating that this is something our models also learned. Overall, these results seem to match the feature importances previously displayed, indicating that the patterns found by the model are significant.

3.7.1 Training

Since a large part of the program is training and improving the physical condition of the patients, we decided to look at the features related to training and its effects separately.

Effect on first visit	d	Effect on change	d
DesireSurgJoint	0.90	PainJointNRS	0.73
Drug	0.53	EQ5DIndex	-0.26
WaitingList	0.52	PainJointFreq	0.17
DrugParacet	0.47	XRayOA	-0.17
SickLeave	0.43	WorstJoint	0.16
DrugOther	0.42	WaitingList	-0.12
DrugNSAID	0.33	DrugNSAID	0.11
DrugCortisone	0.29	Drug	0.11
DrugHyaluronic	0.25	PrevInfoAct	-0.10
FearMovement	0.24	PrevPT	-0.09
PrevInfoWeightLoss	0.18	Age	-0.08
PrevSurgContralat	0.12	Employment	-0.08
XRayOA	0.08	FearMovement	0.08
PrevSurg	0.06	SickLeave	0.08
DrugGlucosamine	0.02	Sex	0.07
PrevInfoAct	-0.01	Education	0.06
PrevPT	0.01	Smoking	0.05
		EQ5DVAS	-0.05
		DrugParacet	0.05
		BMI	0.04
		Xray	0.04
		PrevSurgContralat	-0.03
		PrevInfoWeightLoss	-0.03
		Weight	0.02
		DesireSurgJoint	-0.01

Table 3.6: Effect of features on pain NRS, $p < 0.05$

The first two features related to training in the dataset is ExerciseWeekly and PhysActWeekly; both are 7 point scales describing the amount of time the patient spends each week training or engaging other kinds of physical activities.

Feature	Mean	std
FV ExerciseWeekly	3.070	1.744
M3 ExerciseWeekly	3.564	1.640
FV PhysActWeekly	4.818	1.665
M3 PhysActWeekly	5.035	1.537

Table 3.7: Values for training features

Which implies that the patients do become more active after the program, which we can confirm with a hypothesis test. $p < 0.0001$, $d = 0.134$.

The dataset also includes features for the number of supervised training occasions the patient attended and if the patient had received individual training from either a physiotherapist or a work therapist. We split the groups for supervised training occasions into those who had attended at least one, and those who did not attend any. As before, we looked at how these features correlated with the change in NRS, in addition to if the patients were active or started training during the program. The 'active' group is the group of patients who have answered 3 or higher for both Exercise and Physical Activity at M3; the other group is the group that has two or lower for any of the two.

Title	Mean (No)	Mean (Yes)	d	p
Trained	0.860	1.158	0.138	<0.0001
Started training	0.914	1.094	0.084	<0.0001
With Physiotherapist	1.075	0.928	0.070	<0.0001
With Work therapist	0.626	0.998	-0.151	0.001
Supervised training	1.156	1.143	0.006	0.044

Table 3.8: Training Results on pain NRS

In Table 3.8 we can see that although the effect size is slight, there is a noticeable difference between the patients that trained and those who did not. The difference between those who started training and those who still did not train by M3 is also small. Interestingly the patients who received individual training from a work therapist correlated negatively with a change in NRS, their condition worsened. We cannot draw any conclusion as to why, but a guess is that those patients are farther along in the condition than the others. Attending supervised training and individual training with physiotherapist both display low effect sizes.

To draw more conclusions for the supervised training, we could look at more groups since the feature includes information on how many occasions the patient attended.

3.7.2 Hip/Knee

Since the previous models indicated that the pain location was an important feature we also looked at how this impacted the results. The patients were split into two groups, patients with hip pain, and patients with knee pain. From our previous overview of the dataset, we know that patients with knee pain make up two-thirds of the patients and those with hip pain one-third.

Feature	Mean (Hip)	Mean (Knee)	d	p
NRS	0.821	1.182	-0.169	<0.0001
Weight	77.79	81.67	-0.251	<0.0001
BMI	27.04	28.42	-0.288	<0.0001
EQ5D Hygiene (5L)	1.520	1.290	0.356	<0.0001
EQ5DVAS	64.97	67.18	-0.115	<0.0001

Table 3.9: Hip/Knee Results, $p < 0.05$ and $abs(d) > 0.10$

From Table 3.9 we can see that patients with knee pain improve more than those with hip pain, though the effect size is small. We can also see that knee-pain correlates with increased weight and BMI, also at small effect sizes. There is also a small difference in EQ5D Hygiene, i.e., how much difficulty the patient is having with getting dressed and personal hygiene.

3.7.3 Naturopathic drugs

We also found it interesting to look at the effects of naturopathic drugs. Splitting the patients into two groups, 3281 patients who used naturopathic drugs at M3 and 35712 patients who did not.

	Mean (Used)	Mean (No Use)	d	p
NRS at start	5.335	5.333	0.001	0.422
NRS change	1.075	1.062	0.006	0.270

Table 3.10: Naturopathic drugs

Initially studying the differences between the NRS means of the two groups at First Visit we find a p-value of 0.422, which is larger than 0.05 and thus we cannot discard the null hypothesis and assume that users of naturopathic drugs are in more or less pain at the start. Looking instead at the change in NRS the same is true. We cannot assume a significant difference here either, and the effect size is similarly very small. Having concluded that we cannot find a significant difference in starting value or in the change we can instead test for equivalence. We do this by using a confidence interval for the change, to discard the new null-hypothesis that means are different we pick 10% of the difference for our critical area

$$0.1 \cdot \frac{1.074843 + 1.061799}{2} = 0.107$$

to reject the null-hypothesis. We construct the confidence interval as a TOST (Two One-Sided Test) at a significance of 90%, meaning that the actual value of the difference is with 90% certainty in the confidence interval. Some quick calculations give us the interval $(-0.03, 0.09)$, which contains 0 and is inside our critical area we previously selected as 0.107.

In summary, naturopathic drugs do not contribute to a significant improvement in NRS with 90% certainty.

3.8 Predicting Non-Responders

As previously noted, while studying the data we noticed that the number of patients who did not answer the checkup was quite significant (35% after three months).

As we want to obtain as broad a picture as possible of the contributing factors we also want to look at differences and patterns when comparing this class of patients to the responders. An example of this would be finding that the non-responders have a higher than average pain, which could point to interesting findings that relate to the outcome of the program.

To perform this analysis, we used the framework we had previously constructed for outcome predictions and trained it to predict non-responders instead. For this, a Gradient Boosting model was used, trained on the same data as previously and also evaluated using the same methodology previously established.

Chapter 4
Results

In this section we will present the final results of the models we trained and their performance on our selected test set.

4.1 Final Prediction Performance

With both baseline models and a neural network for NRS, we can examine their behavior in more detail. We use the 'macro' strategy to average the performance over the classes.

Model	Metric	Train	Accuracy	Precision	Recall	F1	ROC AUC	Avg Prec
Gradient Boosting	EQ5D	0.753	0.695	0.660	0.520	0.502	0.751	0.568
Gradient Boosting	VAS	0.693	0.674	0.601	0.460	0.458	0.750	0.554
Gradient Boosting	Freq	0.611	0.594	0.564	0.483	0.495	0.724	0.535
Gradient Boosting	NRS	0.614	0.596	0.593	0.436	0.429	0.718	0.519
Neural Network	NRS	0.525	0.547	0.484	0.502	0.490	0.700	0.495

Table 4.1: Model performance

From the ROC Curves in Figure 4.1, we see that our predictions are better than luck for all indicators. We can also see that the EQ5D curve shows a slightly different behavior from the others by diverging at FPR 0.4.

Studying the precision-recall curves, we note that the 'worsened' class is the hardest for all models, while stable and improved are both easier and closer to each other. In the confusion matrices, we can see that the reason for this is that the models tend to fit towards predicting stable, which is the most frequent value in the dataset.

Prediction performance as shown in Table 3.3 varies between the different metrics with though the models perform similarly throughout. Comparing the different models, we can make the best predictions for EQ5D, though the performance is very similar for all of the indicators.

Most patients are predicted to remain stable as shown in Figure 4.3, which reflects the real data. For the patients who improved, half were predicted to improve and half predicted to be stable. So the models have difficulty distinguishing those who improved. The number of patients predicted as Worsened is overall very low which also matches the data (8.4%) but the majority predicted as such did also worsen. The models also have difficulty distinguishing between whether the patient would improve or worsen. For the rest of the features, the models continue to find it difficult to separate the stable from the improved.

We were unable to build a neural network that beat the gradient boosting model. It managed to get a better F1 score but overall its accuracy was lower, see Table 4.1. Although we attempted to balance the training set by oversampling, the precision-recall curve mirrors the behavior we saw in the other models, with worsened being the hardest to predict accurately, see Figure 4.4.

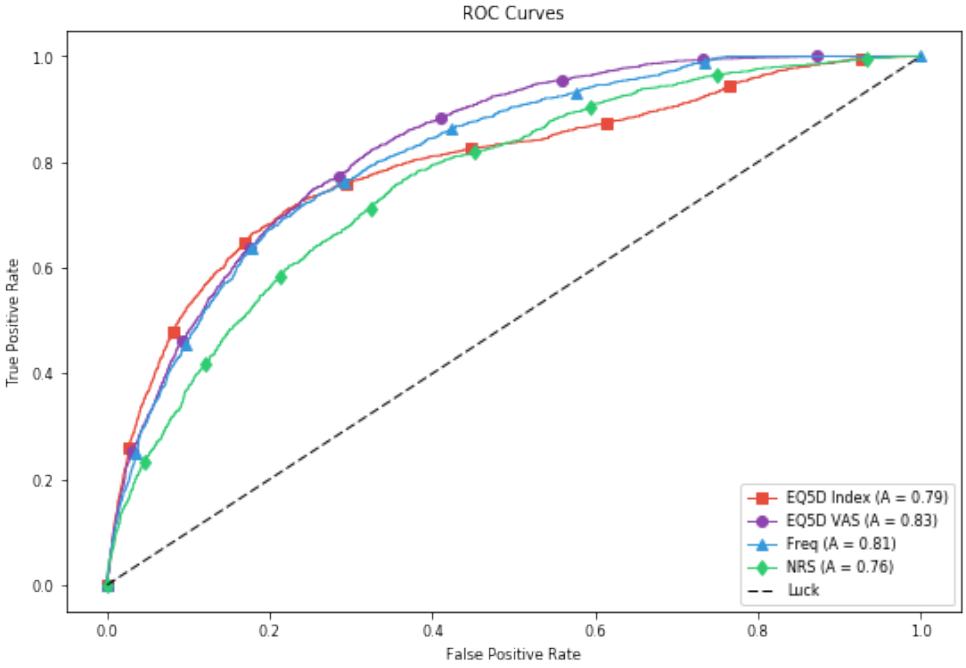


Figure 4.1: ROC curves for predicting an improvement over three months

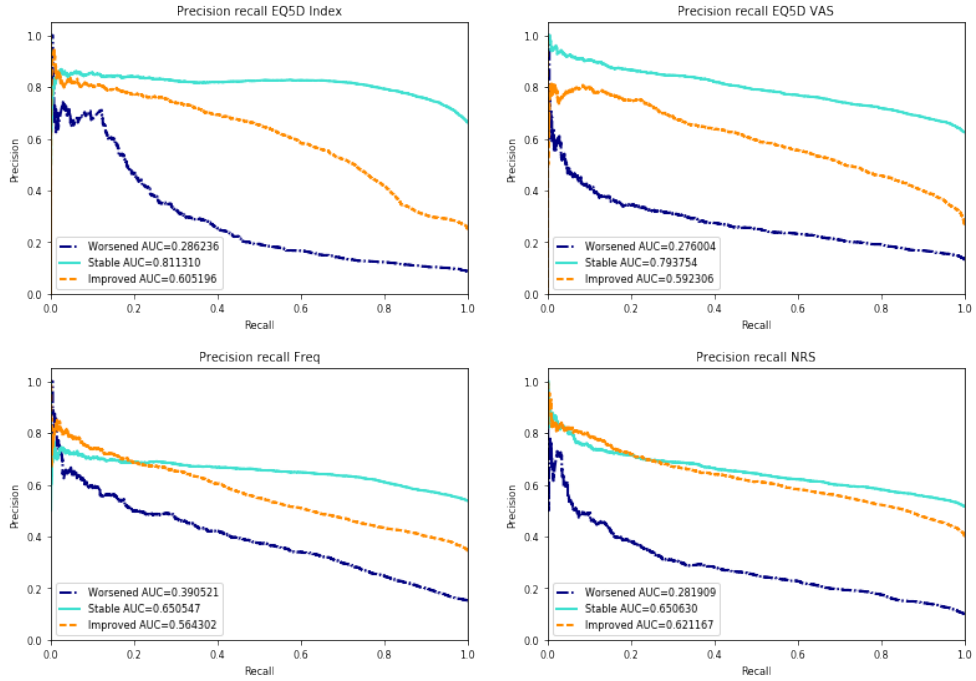


Figure 4.2: Precision recall curves for predicting an improvement over three months

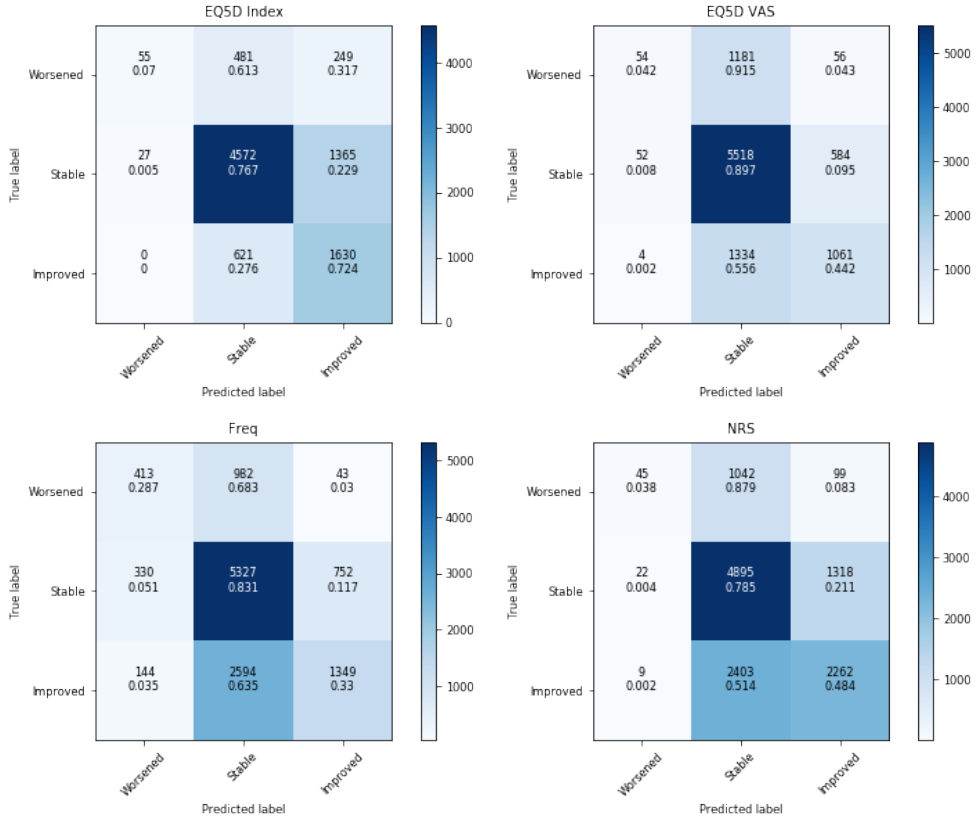


Figure 4.3: Confusion Matrices for predicting an improvement over three months

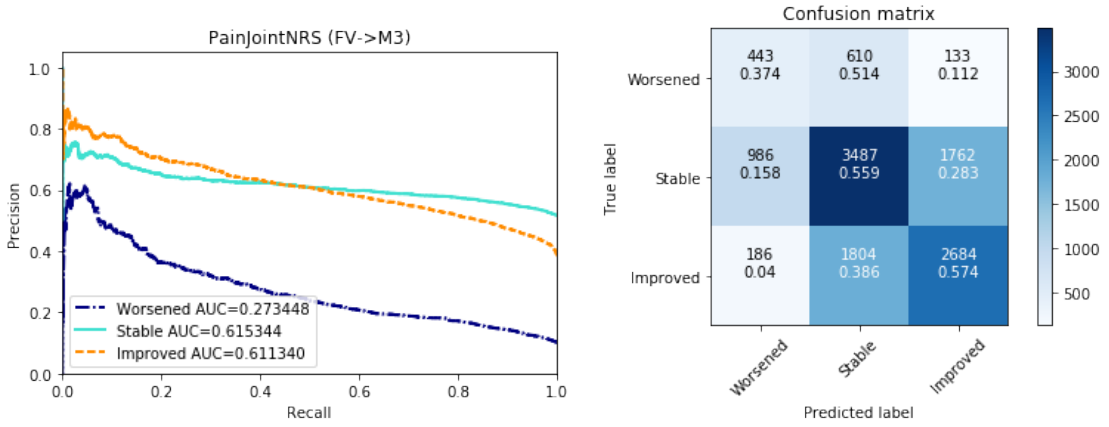


Figure 4.4: Final neural network performance

4.1.1 Predicting Non-Responders

For predicting non-responders we could see that the model overwhelmingly predicts the majority class, which is reflected in the low ROC AUC shown in table 4.2. Considering we are now only dealing with two classes even though we display a higher accuracy than before this seems to indicate that predicting non-responders is a harder problem.

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC	Avg Prec
Gradient Boosting	0.725	0.648	0.520	0.470	0.620	0.787

Table 4.2: Performance for predicting non-responders

4.2 Learning Curve

In order to evaluate the model behaviour we studied the learning curve as shown in Figure 4.5, which shows how the model behaviour changes as the amount of training data is increased.

In the figure the training score and the cross-validation score both seem to converge to the final score we obtained earlier. This indicates that we would not be able to increase generalization by adding more training data, and that to be able to increase performance we need to change the models, probably by adding different variables.

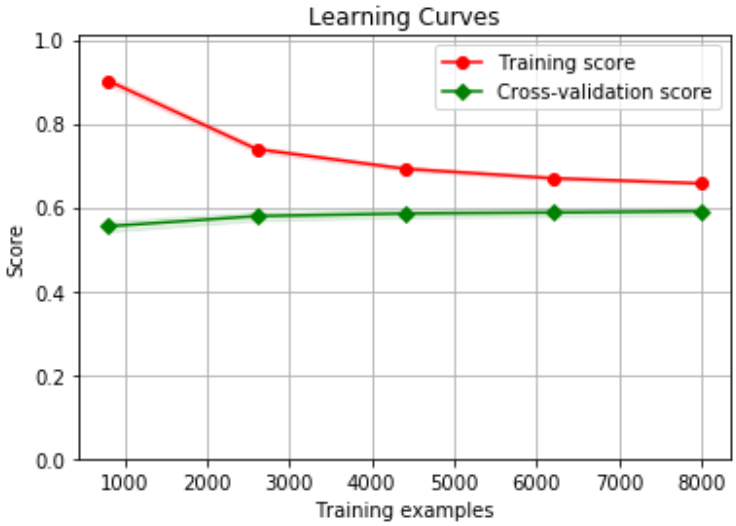


Figure 4.5: Learning curve for Gradient Boosting (NRS)

Chapter 5
Discussion

Since this is the first analysis of this type on this dataset, we cannot compare our results with prior studies. With an accuracy of around 60% on our chosen indicators combined with our evaluation, we are satisfied that the models are finding useful patterns.

The performance of the different models was also very close in most cases, and the models chosen should be able to capture somewhat complex interactions between features, leading us to conclude that major performance increases could be hard to achieve. However, it would be interesting to combine this dataset with different kinds of Electronic-Health-Records, to capture a complete picture of the patient.

We had hoped that designing a neural network by ourselves would allow us to get a solution that performed better than the existing models as we would be able to tune it for our specific problem. Tuning the networks turned out to be very time consuming as many different neural network designs could be combined with many different regularization methods and parameters. The result was a small neural network with a single hidden layer of size 8. The small size indicates that the network only uses a few feature combinations for its predictions, this behavior and its performance is very similar to the other models that we used and seems to confirm that any increases in performance using only this dataset would be unlikely.

Neural Networks are currently highly praised for their versatility and good performance, and while it did perform well, it was not to a significantly higher degree than the other models tested. Our network is of the simple feed-forward type, and more complex architectures with recurrent or convolutional layers could be able to show better performance. However, taking into account the nature of the data, and the lack of dependencies between its features we find this somewhat unlikely. It provides an interesting avenue for future research though.

The overall behavior of the models seemed to indicate that those in better health either remained stable or worsened, those who were in worse health either improved or dropped out.

We found the behavior of the EQ5D Index interesting, in which almost all of the pre-

dictive power came from the initial value. Similar behavior was present for the other indicators, but not to the same degree. The behavior seems to imply that those who have a low initial value revert to a better quality of life. This seems a little bit paradoxical to us, and that domain experts should be able to produce a better explanation for this behaviour.

As the indicators were highly ranked when considering both the feature importance and statistical analysis, they appear to work well as indicators of patient health. Though the ASES scores are due to be removed as they have not been statistically validated, we found them to contribute noticeably to the prediction results.

The BOA Annual report includes data on the difference between units, which we did not do. There appear to be significant differences in their results, but we simplified our work by not controlling for these, which may or may not bias our results.

While we are confident in the performance of the models we obtained, for future work, the performance should be able to be increased. As the treatment program is still ongoing, the amount of data continues to grow which could help our problem with too few patients in the minority classes. However, the learning curve indicate that simply adding more of the same data is unlikely to increase performance by much. To alleviate this there are several other sources of data that could also be used in unison to capture a better picture of the patient. Similar to the BOA-registry, there exist a registry for patients who have had hip surgery, which could be used to compare the two groups and find further patterns. Additionally, more general sources of electronic-health-records containing medicines and other patient conditions and could provide useful information for the predictions.

As far as this dataset is concerned, we feel that we did enough experimenting with feature representations, imputing, scaling, and model parameters that another approach is needed to achieve significantly improved performance on the tasks analyzed. There is however a lot of potential for more interesting work to be done on the dataset, for example attempting to isolate groupings and clusters of patients, examining correlations between features in greater detail, expanding the list of indicators to predict, and analyzing how the patient results change after one year.

We found the tools used to be straightforward and easy to use. Both Scikit-learn and Keras have online documentation available which was very comprehensive. While training the models, the neural networks in particular, did take time we were also happy to have fast graphics cards available. After having seen that the performance of the Gradient Boosting models was best among the ones we tested, we did consider to try XGBoost. XGBoost is another library, but which focuses on Gradient Boosting models. In the end, we opted not to include another library to prevent scope creep.

The statistical analysis more closely resembles a traditional approach to analyzing data, but it gave us valuable insight into how well the models were modeling reality. We found it interesting that the effect sizes were smaller than we expected for the different treatments, but as described by Nüesch et al. (2010). in their meta-study on the effect sizes of OA treatments, the effect sizes tend to relate to the study size, with smaller studies displaying larger effect sizes. As the amount of data we have used compared to the studies analyzed by them is much greater, we find that that would explain the smaller effect sizes. However, that does not explain why the effect sizes are smaller in the first place.

5.1 Conclusions

In this thesis we have constructed several machine learning models and evaluated them in order to predict patient outcomes in osteoarthritis dataset. In order to do that we began with two questions:

> **Is it possible to predict the outcome?** Yes, depending on the model we have shown an accuracy of 60%-70% on the whole dataset.
>
> **What factors contribute to the outcome?** We have found several, among the most important is initial health, age, weight and BMI.

In short we can conclude the following.

- We obtained the best results with a Gradient Boosting model; the others are not far behind however
- Neural Networks do not appear to outperform a Gradient Boosting model significantly and require more tuning
- The most important features are EQ5D Index, EQ5D VAS, PainJointFrequency and NRS, followed by Age and Weight/BMI
- For predictions we have shown an accuracy of 60%-70% on the whole dataset depending on the model
- Predicting EQ5D gets a 70% accuracy by simply saying that those with an index under 0.6 will improve, and the others are stable
- The models have a hard time predicting which patients worsen, in part due to that the amount of patients who do worsen is relatively low
- Patients that begin the program with worse values tend to improve the most, and this is the largest factor in how much the patients improve
- Patients with knee pain improve more than those with hip pain
- Exercise helps the results, while increased weight has an adverse influence on the result
- Predicting non-responders on the followup is more difficult, with many false-negatives
- The learning curves indicate that simply increasing training data is unlikely to improve performance

Bibliography

BOA-Registry. (2015). Annual report 2015. Retrieved from http://www.registercentrum.se/sites/default/files/dokument/boa_arsrapport_2015_utokad_version.pdf

Desmeules, F., Dionne, C. E., Belzile, É. L., Bourbonnais, R., Champagne, F. & Frémont, P. (2013). Determinants of pain, functional limitations and health-related quality of life six months after total knee arthroplasty: results from a prospective cohort study. *Sports Medicine, Arthroscopy, Rehabilitation, Therapy & Technology*, *5*(1), 2. doi:10.1186/2052-1847-5-2

Estabrooks, A., Jo, T. & Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, *20*(1), 18–36. doi:10.1111/j.0824-7935.2004.t01-1-00228.x

Freund, Y. & Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23–37). Springer.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, *20*(8), 832–844.

Ioffe, S. & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. *CoRR*, *abs/1502.03167*. Retrieved from http://arxiv.org/abs/1502.03167

Lee, S.-K., Son, Y.-J., Kim, J., Kim, H.-G., Lee, J.-I., Kang, B.-Y., … Lee, S. (2014, April). Prediction model for health-related quality of life of elderly with chronic diseases using machine learning techniques. *Healthc Inform Res*, *20*(2), 125–134. Retrieved from https://synapse.koreamed.org/DOIx.php?id=10.4258/hir.2014.20.2.125

Miotto, R., Li, L., Kidd, B. A. & Dudley, J. T. (2016, May). Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Sci Rep*, *6*, 26094. 27185194[pmid]. doi:10.1038/srep26094

Nüesch, E., Trelle, S., Reichenbach, S., Rutjes, A. W. S., Tschannen, B., Altman, D. G., … Jüni, P. (2010). Small study effects in meta-analyses of osteoarthritis trials: meta-epidemiological study. *BMJ*, *341*. doi:10.1136/bmj.c3515. eprint: http://www.bmj.com/content/341/bmj.c3515.full.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perlich, C. (2011). Learning curves in machine learning. In *Encyclopedia of machine learning* (pp. 577–580). Springer.

Poole, A. R. (2012). Osteoarthritis as a whole joint disease. *HSS journal*, *8*(1), 4–6.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. doi:10.1023/A:1022643204877

Socialstyrelsen. (2012). *Nationella riktlinjer för rörelseorganens sjukdomar stöd för styrning och ledning*. Retrieved from http://www.socialstyrelsen.se/publikationer2012/2012-5-1

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Sullivan, G. M. & Feinn, R. (2012). Using effect size, or why the p value is not enough. *Journal of graduate medical education*, *4*(3), 279–282.

Turkiewicz, A. (2016). *Epidemiology of osteoarthritis in sweden. register and cohort studies on prevalence and mortality.* (Doctoral dissertation, Lund University).

Weng, S. F., Reps, J., Kai, J., Garibaldi, J. M. & Qureshi, N. (2017, April). Can machine-learning improve cardiovascular risk prediction using routine clinical data? *PLOS ONE*, *12*(4), 1–14. doi:10.1371/journal.pone.0174944

Wikimedia Commons. (2013). Artificial neural network with layer coloring. Retrieved from https://en.wikipedia.org/wiki/File:Colored_neural_network.svg

Wirth, R. (2000). Crisp-dm: towards a standard process model for data mining. In *Proceedings of the fourth international conference on the practical application of knowledge discovery and data mining*. Citeseer.

Appendices

Table 1: Dataset variables (Swedish)

Occasion	Variable	Definition (Swedish)
FV, M3, Y1	Pat_Date	Datum för besök
FV	Pat_Age	Ålder
FV	Pat_Sex	Kön
FV, Y1	Pat_Weight	Ange din vikt (kg)
FV	Pat_Height	Ange din längd (cm)
FV	Pat_BMI	BMI
Y1	Y1Pat_SurgHipKnee	Har du fått en ny höft- eller knäled inopererad sedan du gick artrosskolan
M3, Y1	Pat_StillPain	Har du fortfarande besvär från dina leder.(Höft,Knä,Hand,Axel)
FV, M3, Y1	Pat_WorstJoint	Vilken led har Du mest besvär från
FV, M3, Y1	Pat_Hip	Höft
FV, M3, Y1	Pat_Knee	Knä
FV, M3, Y1	Pat_Foot	Fot
FV, M3, Y1	Pat_Hand	Hand
FV, M3, Y1	Pat_Elbow	Armbåge
FV, M3, Y1	Pat_Shoulder	Axel
FV, M3, Y1	Pat_Jaw	Käkled
FV, M3, Y1	Pat_Neck	Nacke
FV, M3, Y1	Pat_Spine	Ryggrad
FV, M3, Y1	Pat_WalkDifficulty	Har du gångsvårigheter till följd av dina ledbesvär
FV, M3, Y1	Pat_WalkOtherDifficulty	Har du gångsvårigheter av någon annan anledning än smärta eller nedsatt funktion i lederna
FV, M3, Y1	Pat_CharnleyScore	Charnley klassifikation
FV, M3, Y1	Pat_ArmProblem	Under den senaste veckan, i vilken utsträckning har besvär från arm, axel, eller hand stört ditt vanliga arbete eller andra dagliga aktiviteter
FV, M3, Y1	Pat_PainJointFreq	Hur ofta har du ont i någon led
FV, M3, Y1	Pat_FearMovement	Är du rädd att dina leder tar skada av fysisk träning/aktivitet
FV, M3, Y1	Pat_PainJointNRS	Markera den siffra som motsvarar din genomsnittliga smärta från din mest besvärande led den senaste veckan
FV, M3, Y1	Pat_DesireSurgJoint	Har du så mycket besvär från någon led att du vill bli opererad
FV, M3, Y1	Pat_EQ5DMobility5L	Rörlighet
FV, M3, Y1	Pat_EQ5DHygiene5L	Personlig vård
FV, M3, Y1	Pat_EQ5DActivity5L	Vanliga aktiviteter
FV, M3, Y1	Pat_EQ5DPain5L	Smärtor/besvär
FV, M3, Y1	Pat_EQ5DAnxiety5L	Oro/nedstämdhet
FV, M3, Y1	Pat_EQ5DIndex5L	EQ-5D 5

Occasion	Variable	Definition (Swedish)
FV, M3, Y1	Pat_EQ5DIndex	EQ-5D
FV, M3, Y1	Pat_EQ5DVAS	Ditt nuvarande hälsotillstånd
FV	Pat_Smoking	Rökvanor
FV	Pat_BirthPlace	Är du född i Sverige
FV	Pat_Citizen	Är du svensk medborgare
FV	Pat_MaritalStatus	Vilket av följande alternativ beskriver bäst ditt civilstatus
FV	Pat_Education	Vilken är den högsta utbildning du genomgått
FV	Pat_SickLeave	Har du varit sjukskriven under det senaste året på grund av dina ledbesvär
FV	Pat_SickLeaveDur	I så fall sammanlagt hur länge
FV, M3, Y1	Pat_Employment	Hur ser din arbetssituation ut idag? Välj det alternativ som bäst beskriver din situation
FV, M3, Y1	Pat_ExerciseWeekly	Hur mycket tid ägnar du en vanlig vecka åt fysisk träning som får dig att bli andfådd, till exempel löpning, motionsgymnastik eller bollsport
FV, M3, Y1	Pat_PhysActWeekly	Hur mycket tid ägnar du en vanlig vecka åt vardagsmotion (minst 10 minuter åt gången), till exempel promenader, cykling eller trädgårdsarbete? (räkna inte med träningstid från föregående fråga)
FV, M3	Pat_CompareActivity	Jämfört med andra i din ålder, anser du att du är
FV, M3, Y1	Pat_UCLA	Aktivitetsbedömning. Här ska du uppskatta din fysiska aktivitetsnivå den senaste månaden. Alternativen väljs med hänsyn till hur aktiviteten belastar dina leder
M3, Y1	Pat_Opinion	Vad tyckte du om artrosskolan
M3, Y1	Pat_InfoAppl	Hur ofta tillämpar du det du lärt dig på artrosskolan i din vardag
FV, M3, Y1	Pat_ASESPain1	1. Hur säker känner du dig på att du kan minska din smärta avsevärt
FV, M3, Y1	Pat_ASESPain2	2. Hur säker är du på att du kan fortsätta med dina dagliga aktiviteter
FV, M3, Y1	Pat_ASESPain3	3. Hur säker är du på att du kan undvika att din smärta stör din sömn
FV, M3, Y1	Pat_ASESPain4	4. Hur säker är du på att du kan åstadkomma en liten till måttlig minskning av din smärta genom andra metoder än ökad medicinering
FV, M3, Y1	Pat_ASESPain5	5. Hur säker är du på att du kan åstadkomma en avsevärd minskning av din smärta genom andra metoder än ökad medicinering
FV, M3, Y1	Pat_ASESPainScore	Smärta
FV, M3, Y1	Pat_ASESSymptom1	1. Hur säker är du på att du kan påverka din trötthet

Occasion	Variable	Definition (Swedish)
FV, M3, Y1	Pat_ASESSymptom2	2. Hur säker är du på att du kan anpassa dina aktiviteter så att du kan vara aktiv utan att förvärra dina besvär
FV, M3, Y1	Pat_ASESSymptom3	3. Hur säker är du på att du kan göra något för att muntra upp dig om du känner dig nere
FV, M3, Y1	Pat_ASESSymptom4	4. Jämfört med andra personer med besvär som liknar dina, hur säker är du på att du kan hantera din smärta under dagliga aktiviteter
FV, M3, Y1	Pat_ASESSymptom5	5. Hur säker är du på att du kan hantera dina symptom så att du kan göra saker du tycker om att göra
FV, M3, Y1	Pat_ASESSymptom6	6. Hur säker är du på att du kan hantera den besvikelse/vanmakt som en sjukdom medför
FV, M3, Y1	Pat_ASESSymptomScore	Andra symptom
FV, M3	PT_Unit	Enhet
FV, M3	PT_Unit	Landstingstillhörighet
FV, M3	PT_Unit	Vårdnivå
FV	PT_VisitDate	Datum för första besök
FV	PT_WorstJoint	Patienten har mest besvär från
FV	PT_WorstSide	... och sida
M3	PT_IndEstimPT	Individuell bedömning vid första besök av fysioterapeut
M3	PT_IndEstimOT	Individuell bedömning vid första besök av arbetsterapeut
M3	PT_Theory	Artrosskola - teori
M3	PT_OACommunicator	Tillfälle med artrosinformatör
M3	PT_IndExerPT	Individuell träningsgenomgång, fysioterapeut
M3	PT_IndExerOT	Individuell träningsgenomgång, arbetsterapeut
M3	PT_Interpreter	Har någon del av artrosskolan genomförts med tolk
M3	PT_SupervisedEx	Gruppträning:

Parameter	Value	
Optimizer		Adamax with defaults
Class weights:		
Worsened	1	
Stable	1	
Improved	1	
Batch size	128	
Class distribution oversample:		
Worsened	3	
Stable	1	
Improved	1	
Callbacks:		
Early stopping		monitor='val_fbeta_score', min_delta=0, patience=24, verbose=1, mode='max'
ModelCheckpoint		monitor='val_loss', verbose=0, save_best_only=True, save_weights_only=False, mode='min', period=1
ReduceLROnPlateau		monitor='val_acc', factor=0.1, patience=10, verbose=0, mode='auto', epsilon=0.0001, cooldown=0, min_lr=0

Table 2: Final NN parameters

Package	Version	Package	Version	Package	Version
appdirs	1.4.2	matplotlib	2.0.0	sip	4.18
bleach	1.5.0	mistune	0.7.3	six	1.10.0
cffi	1.9.1	mkl	2017.0.1	sqlalchemy	1.1.6
cycler	0.10.0	nbconvert	5.1.1	sqlite	3.13.0
dbus	1.10.10	nbformat	4.3.0	tensorflow-gpu	1.0.0
decorator	4.0.11	notebook	4.3.1	terminado	0.6
entrypoints	0.2.2	numpy	1.12.0	testpath	0.3
expat	2.1.0	olefile	0.44	Theano	0.8.2
fontconfig	2.12.1	openssl	1.0.2k	tk	8.5.18
freetype	2.5.5	packaging	16.8	tornado	4.4.2
glib	2.50.2	pandas	0.19.2	traitlets	4.3.2
gst-plugins-base	1.8.0	pandocfilters	1.4.1	wcwidth	0.1.7
gstreamer	1.8.0	path.py	10.1	wheel	0.29.0
h5py	2.7.0	pcre	8.39	widgetsnbextension	1.2.6
hdf5	1.8.17	pexpect	4.2.1	xz	5.2.2
html5lib	0.999	pickleshare	0.7.4	zeromq	4.1.5
icu	54.1	pillow	4.0.0	zlib	1.2.8
ipykernel	4.5.2	pip	9.0.1		
ipython	5.3.0	prompt_toolkit	1.0.9	CUDA	8.0.61
ipython_genutils	0.1.0	protobuf	3.2.0	cuDNN	v5.1
ipywidgets	5.2.2	ptyprocess	0.5.1		
jbig	2.1	pycparser	2.17		
jinja2	2.9.5	pydot-ng	1.0.0		
jpeg	9b	pygments	2.2.0		
jsonschema	2.5.1	pyparsing	2.1.4		
jupyter	1.0.0	pyparsing	2.1.10		
jupyter_client	5.0.0	pyqt	5.6.0		
jupyter_console	5.1.0	python	3.5.3		
jupyter_core	4.3.0	python-dateutil	2.6.0		
Keras	1.2.2	pytz	2016.10		
libffi	3.2.1	PyYAML	3.12		
libgcc	5.2.0	pyzmq	16.0.2		
libgfortran	3.0.0	qt	5.6.2		
libiconv	1.14	qtconsole	4.2.1		
libpng	1.6.27	readline	6.2		
libsodium	1.0.10	scikit-learn	0.18.1		
libtiff	4.0.6	scipy	0.18.1		
libxcb	1.12	setuptools	34.3.1		
libxml2	2.9.4	setuptools	27.2.0		
markupsafe	0.23	simplegeneric	0.8.1		

Table 3: Keras enviroment

EXAMENSARBETE Predicting and Analyzing Osteoarthritis Patient Outcomes with Machine Learning
STUDENTER Per-Victor Persson, Hans Rietz
HANDLEDARE Pierre Nugues (LTH), Erik Rehn
EXAMINATOR Jacek Malec (LTH)

Förutspå patientutfall innan behandling med maskininlärning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Per-Victor Persson, Hans Rietz**

En fjärdedel av Sveriges befolkning över 45 lider av ledsjukdomen atros. Maskininlärning och AI ger potential för bättre vård genom att innan behandling förutspå utfallet.

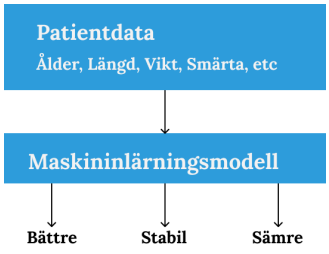
Få kan ha undgått att bilar har börjat köra sig själva, allt fler enheter både lyssnar och svarar på frågor, och dagens mobiler ritar träffsäkert in både moustacher och frisyrer på deras ägare. Dessa exempel och många fler är ett resultat av de otroliga framtsteg inom maskininlärning och artificiel intelligens som skett de senaste åren. Fler och fler uppmärksammar denna potiental och vill undersöka hur ML kan lyfta den egna verksamheten, vilket även sträcker sig till den medicinska världen.

Det vi har gjort är, att genom att träna maskininlärningsmodeller på patientdata för ett behandlingsprogram för artros, se om vi redan innan programmets start kan säga vilka patienter som blir bättre.

Artros är en ledsjukdom som i Sverige drabbar en fjärdedel av befolkningen över 45. Symtomen är ömma eller smärtsamma leder, men smärtan är i många fall tillräcklig för att göra operation aktuellt. Det har dock visat sig att träning kan ge lika bra resultat för många, så för att minska antalet onödiga operationer startades artrosskolan och BOA initativet. I artrosskolan får patienterna både träning och lektioner om artros, samtidigt som patienternas resultat samlas in for att utvärdera programmet.

Denna insamlade data har nu vuxit till att inkludera resultat för cirka 75 tusen patienter, vilket är mycket data där ingen tidigare djupgående analys gjorts. Just att ha bra och helst väldigt mycket data är viktigt för maskininlärning, ju mer data desto fler exempel har modellerna att träna sig själva på.

I slutändan upnådde vi en träffsäkerhet på cirka 65% på hur patienterna mådde efter Atrosskolan (bättre, samma, eller sämre). Svårigheten låg mycket i att andelen som blev sämre var underrepresenterade i datan, så modellerna hade färre exempel att träna på.



Vi kan också dra slutsatsen av resultatet att modellerna upptäckte mönster i vilka personer som faktiskt blir bättre efter programmet och vilka som ligger kvar på samma smärtnivåer. Den slutsatsen är spännande, för kan man identifiera grupperingar av patienter och sedan anpassa behandligen utefter gruppen betyder det att man kan ge en mer individualiserad vård där fler kan bli bättre, något som vi i framtiden tror kommer bli allt vanligare.