Investigating the effect of weather for predicting the hospitalization of patients with end-stage renal disease

Axel Nyström
Investigating the effect of weather for predicting the hospitalization of patients with end-stage renal disease

(Whether Weather Makes VÖR® Better)

Axel Nyström
axel.nystrom@gmail.com

June 14, 2018

Master’s thesis work carried out at Lytics Health AB.

Supervisors: Mattias Sellin, mattias.sellin@lytics.ai
Jacek Malec, jacek.malec@cs.lth.se
Examiner: Elin Anna Topp, elin_anna.topp@cs.lth.se
Abstract

Patients with end-stage renal disease (ESRD) frequently suffer complications that lead to hospitalization, and predicting such hospitalizations ahead of time is potentially of great benefit. Lytics Health AB has developed an AI system that uses physiological measurements from the patients to train a Random Forest classifier for making such predictions.

However, physiological measurements alone do not paint a complete picture of the patient risk profile. It is believed that weather might contain one of the missing pieces, since research has shown it to correlate with various diseases that are common comorbidities for patients with ESRD.

In this thesis the effect of adding weather measurements to an already developed AI system was investigated. A test environment was created where different combinations of measurements like wind, precipitation, temperature, atmospheric pressure and humidity could be evaluated on different patient groups. In the end, no significant improvement due to weather was observed.

**Keywords**: machine learning, random forest, ESRD, dialysis, weather
Acknowledgements

I would like to thank my supervisors Jacek Malec at LTH and Mattias Sellin at Lytics for their help and feedback. I would also like to thank everybody at Lytics for the opportunity, for making me feel like a part of the team, and for all the discussions and luxurious lunch breaks. Thanks to Kasper Tall and Olle Nyström for proofreading the report and giving valuable suggestions and feedback. Thanks to mom for always believing in me. Finally, a special thanks to Venus for the reminder that the weather can always get worse.
# Contents

## 1 Introduction

1.1 Background ............................................. 8
   1.1.1 End-stage renal disease and dialysis ........ 8
   1.1.2 Centers for Dialysis Care ..................... 9
   1.1.3 LYTICS VÖR® ........................................ 10

1.2 Previous research on weather and disease ............ 11

1.3 Goal of the project ................................... 12
   1.3.1 Test environment ................................. 12
   1.3.2 Specifying the scope ............................ 12

1.4 A note from our benefactor .......................... 16

1.5 Outline of the report ................................. 16

## 2 Theory

2.1 Binary classification ................................ 17

2.2 Random Forests ....................................... 20
   2.2.1 From trees to forests ......................... 20
   2.2.2 Theoretical properties of the random forest . 21

2.3 Cross-validation strategies .......................... 22
   2.3.1 K-fold ........................................... 22
   2.3.2 Rolling forecast origin ....................... 24

2.4 Performance metrics ................................ 25
   2.4.1 Confusion matrix and related metrics .......... 25
   2.4.2 The receiver operating characteristic ....... 26

2.5 What counts as an improvement? ..................... 26
   2.5.1 The problem ..................................... 28
   2.5.2 Central Limit Theorem to the rescue .......... 28

## 3 Method

3.1 Baseline data ........................................ 29

3.2 Weather data ......................................... 31
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1</td>
<td>Collecting and cleaning the weather data</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Weather features</td>
<td>32</td>
</tr>
<tr>
<td>3.3</td>
<td>Test environment</td>
<td>34</td>
</tr>
<tr>
<td>3.4</td>
<td>Summary of the experiments</td>
<td>35</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Hyper-parameter optimization</td>
<td>36</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Cross-validation tests</td>
<td>37</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Filtering</td>
<td>37</td>
</tr>
<tr>
<td>4</td>
<td>Results and discussion</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Hyper-parameter optimization, first attempt</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>Tests of cross-validation strategies</td>
<td>42</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Results of k-fold cross-validation</td>
<td>42</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Kitchen sink varieties</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Weather</td>
<td>43</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Weather features versus baseline</td>
<td>43</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Weather features versus baseline on subset of patients</td>
<td>46</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Weather features versus subsets of baseline features</td>
<td>47</td>
</tr>
<tr>
<td>4.3.4</td>
<td>The placebo features</td>
<td>50</td>
</tr>
<tr>
<td>4.4</td>
<td>Revisiting hyper-parameter optimization</td>
<td>50</td>
</tr>
<tr>
<td>4.5</td>
<td>Revisiting weather</td>
<td>51</td>
</tr>
<tr>
<td>5</td>
<td>Discussion and summary</td>
<td>55</td>
</tr>
<tr>
<td>5.1</td>
<td>Discussion of the results</td>
<td>55</td>
</tr>
<tr>
<td>5.2</td>
<td>Future work</td>
<td>56</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Other weather sources</td>
<td>56</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Automatic feature extraction</td>
<td>56</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Patient clusterings</td>
<td>57</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Different classifiers</td>
<td>57</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Different target labels</td>
<td>58</td>
</tr>
<tr>
<td>5.3</td>
<td>Summary and conclusion</td>
<td>58</td>
</tr>
<tr>
<td>Bibliography</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>Appendix A Abbreviations</td>
<td>69</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

End stage renal disease (ESRD) is a medical condition in which kidney function is reduced to the extent that either regular dialysis treatment or a kidney transplant is necessary for survival. In the United States, over 670,000 people are affected, and of these over 460,000 depend on dialysis to survive [1]. The typical patient will receive dialysis treatment several times per week at specialized clinics, but because of their condition, they suffer increased risks of various complications that require hospitalization.

A software system has in recent years been developed, which uses physiological data collected during dialysis treatment in order to predict which patients are most at risk of complications that may lead to hospitalizations. This system is called LYRICS VÖR®, hereafter referred to as the System, and it is being developed by Lytics Health AB (Lytics) in cooperation with the Centers for Dialysis Care, Inc. (CDC), who provide the necessary data. The idea is that by predicting disease development, it may be possible to take preventive measures that ultimately change the course of the disease for the better.

It has long been known that seasonal and meteorological parameters can have a significant impact on both hospitalization rates and mortality due to a number of different diseases that commonly occur for individuals with ESRD. Examples of such conditions include cardiovascular and respiratory diseases [2, 3, 4, 5]. The hypothesis therefore is that weather might hold some predictive power over the hospitalization of at least some subset of ESRD patients, and it was the goal of this master’s thesis to investigate the feasibility of improving the System by incorporating weather measurements into the model. Improving the predictive power of the System can potentially help to prevent unnecessary hospitalizations, alleviate suffering and extend the lives of thousands of people, while also reducing the economic cost to society.

Following in this chapter is a more detailed background to ESRD and the way the System is currently used in practice. After that we present a brief overview of the literature on the topic of weather and disease. In the next section, the goal and scope of the master’s thesis is described along with a general outline of how the task was approached. After a
note on the confidentiality of certain aspects of the project, the chapter is wrapped up with an overview of the structure of the rest of the report.

1. Introduction

1.1 Background

This section will present the background information necessary to understand the context and purpose of this thesis. It begins with some general information about kidney disease, dialysis treatment and overall disease prevalence. Following is a quick introduction to the American health care system with a focus on the Centers for Dialysis Care Inc. (CDC), which is the organization in Cleveland, Ohio, that is cooperating with Lytics to improve the efficiency and accuracy of their preventive care. The section concludes with a description of the AI system developed by Lytics.

1.1.1 End-stage renal disease and dialysis

The kidneys are a pair of internal organs, each roughly the size of a fist, located on either side of the spine just under the rib cage. The kidneys are complicated organs involved in a number of life-critical tasks. These include the removal of drugs and waste products from the blood, the balancing of body fluids, the production of red blood cells as well as hormones that regulate other important organs [6]. If the kidneys are damaged, their ability to function properly can be permanently reduced. Any condition where this occurs is referred to as chronic kidney disease (CKD), which is further classified into five stages based on severity. The last stage of CKD is called end-stage renal disease (ESRD), or kidney failure, and it is characterized by requiring either a kidney transplant or regular dialysis treatment to survive. Common causes for CKD include diabetes, high blood pressure, autoimmune diseases and genetic diseases [6]. There is currently no cure for ESRD, but with proper treatment, life can be significantly extended.

About 30% of all American ESRD patients live with a kidney transplant. The remaining 70% require frequent dialysis, which is a process of filtering the blood to get rid of excess fluid, waste products and toxins [7]. There are multiple types of dialysis, the most common of which is known as hemodialysis, which is where the blood is pumped through an external machine that performs the dialysis. Such treatments typically take about four hours to complete, and must be performed several times per week, for some patients even daily. The treatment usually takes place in specialized clinics, although it is possible to receive the proper training and equipment to do the treatment at home instead [8].

Dialysis treatment replaces some, but not all the functions of a healthy kidney. It follows that if the kidneys fail, the risk of many other diseases, so called comorbidities, increases dramatically. Common comorbidities of ESRD include chronic obstructive pulmonary disease (COPD), septicemia, anemia, hyperkalemia, hyperphosphatemia, and various forms of heart disease [9,10]. These increased risks are also reflected in the hospitalization, readmission and mortality rates. In 2013 in the United States, the average number of hospitalizations for hemodialysis patients was 1.7 per patient year, with a 34.8% risk of readmission within 30 days of discharge. The mortality rate for hemodialysis patients was 172 per 1 000 patient-years in 2013 [1].
CKD is often referred to as a silent disease, because symptoms usually only present themselves at a rather late stage. Indeed, in the United States, approximately 15% of the adult population have some form of CKD. About half of those with a severely reduced kidney function are aware of their condition, and only 4% of those with a mildly reduced kidney function are aware of it [7]. The worldwide prevalence of CKD is similar, although difficult to estimate [11].

CKD, and especially ESRD, represents an enormous cost to society as well as a deep source of suffering. In total over 678,000 individuals were being treated for ESRD in the United States in 2014, with roughly two-thirds receiving dialysis. The total Medicare fee-for-service spending on ESRD patients was $32.8 billion in 2014 [12]. The hospitalizations alone account for more than a third of the costs of treating ESRD, and if the number of hospitalizations could be reduced, costs would decrease and patient outcomes would improve [13].

1.1.2 Centers for Dialysis Care

The Centers for Medicare and Medicaid Services (CMS) is a part of the Department of Health and Human Services in the United States, tasked with administering Medicare and related programs. Medicare and Medicaid are federal health insurance programs that cover different groups of people. In particular, Medicare is an insurance which covers the health care costs for people with ESRD [14].

CMS has partnered with a number of dialysis clinics, nephrologists and other health care providers to create ESRD Seamless Care Organizations (ESCO). The purpose of the ESCOs is to provide beneficiaries with person-centered, high-quality care, and to accomplish this, CMS puts certain requirements on the ESCOs. If the ESCO fails to fulfill any of the requirements, it will be held financially accountable, but if on the other hand it does manage to fulfill the requirements, the ESCO will receive any savings in cost for the affected patients [15].

The CDC is a non-profit provider of dialysis and related health services to people with ESRD. The CDC provides 18 facilities located in and around Cleveland, Ohio, and they are registered as an ESRD Seamless Care Organization [16]. As an ESCO, CMS will align a certain number of the patients at CDC that are eligible for the program, and it is then up to the CDC to make sure that they fulfill the requirements for those patients. Out of the roughly 2,000 ESRD patients under the care of CDC, about 600 are enrolled in the ESCO program.

One of the measures taken by CDC to fulfill the requirements from CMS is to appoint special nurse care managers with the task of preemptively finding and caring for those patients with the greatest risk of complications. At CDC, each care manager is responsible for 100 ESCO patients, and they spend somewhere between 30% and 50% of their time on data mining, looking through patient and hospitalization statistics in an attempt to figure out which patients are most likely to benefit from preventive care. The rest of the time is used to coordinate and delegate tasks, meeting with and caring for the patients. For example, the nurse care manager might find that a patient has a low hemoglobin count, and then asks a doctor to prescribe the proper medicine. Another patient’s lab results might show abnormal protein levels, and the proper reaction then might be to get the patient in...
1. Introduction

Figure 1.1: Example view of the front end of the System. The user can sort patients by various characteristics, including the predictions made by the machine learning classifier. The names and dates have been obfuscated in this image.

touch with the dietitian. The nurse care manager usually only has time to deal with 15 or fewer patients each week.

The idea behind LYTICS VÖR® is to increase the efficiency of the care managers by automatically ranking the patients based on an estimated likelihood of future complications. If such a ranking can be made accurate enough, it would significantly speed up the data mining process for the care managers, allowing them to care for a larger number of patients, while also focusing their attention on the patients that would benefit the most from it.

1.1.3 LYTICS VÖR®

LYTICS VÖR®[1], which we shall call the System, is a decision support system designed by Lytics to assist registered nurses (RN) and other health care professionals with the task of prioritizing their attention on patients that are more likely to be in need of short term medical attention. From the point of view of the RN, the system will sort patients by the likelihood that they will become hospitalized within 30 days. The likelihoods are estimated by a machine learning algorithm that is part of the System. The sorted patient list is presented to the RN through a graphical user interface (GUI) together with other useful statistics as illustrated in figure 1.1. The intention is that by paying closer attention to those estimated to be exposed to higher risks, the RN will save time while also increasing the accuracy and efficiency of the preventive measures that can be taken. Such improvements in efficiency can ultimately lead to reduced hospitalizations and health care costs, reduced suffering and increased quality of life for the patients [13].

The System uses physiological data collected from the patients during their dialysis treatments, together with various lab results and hospitalization statistics. The measurements include blood pressure, heart rate, amount of fluid removed, blood flow rate, patient

1Vör is a goddess in Norse mythology, wise and inquiring, so that nothing can be concealed from her.
1.2 Previous research on weather and disease

This section presents the findings of a brief survey of the literature on the topic of weather and its possible connections to hospitalizations and mortality.

The weather measurement most commonly associated with health complications appears to be temperature, and the second most common is humidity. Cold temperatures have for example been linked to fluid accumulation for hemodialysis patients [17]. Temperature and humidity have also been found to correlate with the blood pressure of hemodialysis patients in separate studies from both Poland and Japan [18, 19]. However, a Canadian study failed to find any seasonal variations for blood pressure among hemodialysis patients [20]. Cold temperature has also been correlated with high blood pressure in patients other than those with hemodialysis, as well as with mortality in the general population in Macedonia [21, 22]. Both low and high temperatures have been linked to the incidence of COPD admissions in Europe [4, 5]. There are also numerous studies indicating links between weather, especially cold temperatures, and heart disease and mortality [2, 3, 23, 24].

A particularly interesting article by Braga et al. investigated the effect of weather on mortality due to various respiratory and cardiovascular diseases. They used Poisson regression to estimate the delayed effect of weather, and found that for cold cities in the US, high temperatures were associated with increased cardiovascular mortality, but the effect was only observed on the day of death and the day before. The same study also found a correlation between temperature and deaths from COPD, but did not observe any effects from humidity [25].

As can be imagined, air pollution has also been linked to various health issues, including hospitalizations due to ischemic heart disease, and the emergency department visits due to COPD [26, 27]. In this thesis however, we have opted not to investigate the effects of pollution further, primarily due to time constraints.
1.3  Goal of the project

The goal of this master’s thesis, which was conducted at Lytics by Axel Nyström between November 2017 and May 2018, was to investigate the feasibility of improving the System with the help of weather measurements. To this end, weather information was collected, turned into features, and combined in various ways with features from the System. A new software environment was developed to simplify the process of systematically defining, testing and analyzing the effects of various such modifications.

In the remainder of this section, we will give a brief overview of what was tested, how it was tested and why we tested it.

1.3.1  Test environment

Most of the work in this project was focused on developing a test environment that could be used to explore various ideas. This test environment takes as input the training data and labels constructed within the System (denoted by \( X_b, y \)), as well as weather measurements downloaded from the National Center for Environmental Information (NCEI), to produce test results that can later be analyzed. The test environment and its connection to the System is illustrated schematically in figure 1.2.

The test environment can be roughly divided into three parts, illustrated by boxes in figure 1.5. The feature extraction box is concerned with converting the raw weather data into features \( X_w \), that will hopefully be useful to the classifier. The purpose of the filter box is to select some subset of the baseline features \( X_b \) and labels \( y \) from the System and combine them with some subset of the weather features \( X_w \) produced by the feature extraction box. The AI box is more complicated, and will be further broken down into smaller pieces that will be discussed in greater detail in chapter 2.

One test case for the test environment consists of specifying the exact behavior for each of the boxes. In this project, more than a hundred distinct test cases were defined, executed and analyzed, and the highlights of these tests will be explored in chapter 4.

1.3.2  Specifying the scope

There are many things that could be adjusted in an attempt to improve the system, and most of these things can be changed independently of the rest. This leads to a combinatorial explosion of different possible options, none of which can be theoretically discarded. To make progress, a strategy is necessary for determining what should be included in the scope of the project.

Referring again to the overview of the test environment in figure 1.5, we will experiment with each of the three boxes: the feature extraction, the filtering and the AI. The filtering is done in two primary ways: we select subsets of features from the baseline and from the weather features, and we select different groups or clusters of patients from the dataset. Within the AI-box, which is further broken down in figures 2.1 and 2.2, we will look at different cross-validation methods and consider how to aggregate the resulting scores. We will attempt to fine-tune the hyper-parameters of the classifier.
1.3 Goal of the project

Figure 1.2: Overview of the test environment that was used to explore the potential benefits of adding weather to the System. The features and target labels that are created by the System \((X_b, y)\) is fed as input to the test environment, together with weather data captured from NCEI. Whereas the System proceeds to use the features and labels to make predictions about the future that is subsequently presented to the CDC, the aim of the test environment is to thoroughly evaluate the performance of various combinations of physiological and meteorological measurements in predicting the specified labels.
Figure 1.3: The test environment (the contents of the dashed box) can be divided into three parts. The feature extraction part creates features from the weather data, the filter part combines weather with features from the System, and the AI part trains and evaluates a classifier that tries to predict the labels.
Weather features

The input weather data was limited to various measures of temperature, precipitation, humidity, atmospheric pressure and wind. This somewhat arbitrary limitation was primarily inherited from the data set which was collected from NCEI, but it was also a means of reducing the scope of the project. From the weather data, two additional types of features were constructed: features relating to apparent or felt temperature (heat index and wind chill), relative humidity, and time-series features derived by looking at past measurements. The time-series features were variations of rolling averages and Fourier coefficients.

Patient clustering

One hypothesis was that if weather can be used to predict hospitalizations, then the effect would at any rate be fairly small and statistical in nature. Surely weather does not affect everyone the same, or to the extent that it does, the effect itself is stochastic, such that we can only say that a person is more or less likely to suffer some particular complication as a result of a particular weather condition, rather than suggesting that any person exposed to such weather will necessarily see a specific outcome. Certain common comorbidities of ESRD, as we learned in section 1.2 have for example been shown to correlate with weather, but it is not known whether all ESRD patients are equally affected by such weather conditions.

It seemed plausible therefore, due to the diversity of the patient groups and the large overlap of different types of comorbidities, that we might be able to improve the AI system to a greater extent if we specifically limit the dataset so that it only uses some certain subgroup of patients. Since there are almost 10 000 patients in the data set it would of course be impossible to try all clusterings, but with some intuition and domain expertise, perhaps a suitable clustering might still be found. The clusterings that were tried focused exclusively on categories of symptoms such as shortness of breath, cardiovascular issues and similar.

The multiple straw hypothesis

The observation was made that although weather might have an effect on certain patients, adding weather as features to the classifier will not provide additional information if the things that are affected by weather are already measured directly. In light of this, one might hypothesize that perhaps adding all these new weather features is essentially the same as adding more straws to a glass of lemonade: Clearly, adding an extra straw to the same glass won’t give us more lemonade, but perhaps if we removed some of the existing straws, the weather straws might become useful.

As a way of testing this hypothesis, the baseline features were placed in various categories that were subsequently used in different combinations to see if adding weather features to them would make an improvement.

It was during these tests that it was discovered that although the classifier did actually enjoy a large improvement when using some groups of baseline features combined with weather, it was also equally improved when those same groups were combined with random noise. This slightly disturbing discovery was investigated in some detail, and we will cover the results in section 4.3.4.
1.4 A note from our benefactor

Due to the highly competitive nature of the field, some information that might otherwise have been natural to include in a report of this kind has been deemed sensitive by Lytics, and has for this reason been omitted or kept purposely vague. This includes details about the inner workings LYTICS VÖR®, specifics of how the features have been created, what the AUC ROC score of the model is and what the hyper-parameters are for the random forest algorithm. Also considered sensitive is specific information about the training data, in particular the number of patients included in the system and their actual hospitalization rates. Some plots in this report are therefore presented without a specified scale on the axes, and many of the results are simply offsets to the baseline (i.e., how much better or worse the performance became after some test, when compared to the baseline), rather than the absolute values.

Despite these limitations in what can be made publicly available, all interesting information with respect to the thesis remains completely unaffected, including the main results and conclusions.

1.5 Outline of the report

The rest of the report is structured as follows. In chapter 2 the theory necessary to understand the rest of the project is presented, including an introduction to binary classification, random forests, different cross-validation strategies and performance metrics. Chapter 3 introduces the test environment and some statistics about the data set that was used, as well as the different experiments that were constructed.

Chapter 4 presents the results of some of the experiments, roughly in the order of discovery. It also contains some discussion about the results as they are presented, motivating the direction in which the tests were taken. Chapter 5 contains some further discussion of the results and some ideas for future work that fell outside of the scope of this project. The chapter concludes with a summary of the project.
In this chapter we will cover some of the theory necessary to understand the most complicated box in figure 1.3, which is the AI box. We can enlarge this box and further divide the contents into smaller pieces, as is done in figure 2.1. The train, predict, evaluate-box is further expanded in figure 2.2. All the boxes in these two pictures will be explained in more detail in this chapter, starting with the heart of it all: the classifier.

Although the focus of this project was not to experiment with different classifiers, we will explain what the classification problem is exactly, and also give some details about the Random Forest classifier that has been used.

### 2.1 Binary classification

The goal of binary classification is to associate an observation of a $d$-dimensional random vector $X \in \mathcal{X} \subset \mathbb{X}^d$ with a binary label $Y \in \{0, 1\}$. It is assumed that $(X, Y)$ has a joint distribution with a probability measure $\mathbb{P}_{(X,Y)}$. A classifier is a function $c : \mathcal{X} \rightarrow \{0, 1\}$ that attempts to solve the classification problem. The loss, or probability of error of a classifier $c$ is defined by

$$L(c) = \mathbb{P}_{(X,Y)} \{c(X) \neq Y\} \tag{2.1}$$

The classifier that minimizes $L$ is known as the Bayes classifier and is given by $c^*(x) = \arg\max_{r \in \{0, 1\}} \mathbb{P}_{(X,Y)}(Y = r \mid X = x)$. The corresponding minimal loss is $L^* = L(c^*)$.

In a supervised learning context, we define the training set $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ of $n$ tuples, such that $X_k \in \mathcal{X}$ and $Y_k \in \{0, 1\}$ for $k = 1, \ldots, n$. The task is then to create an algorithm that uses the training set $D_n$ to construct a classifier $c_n$ that is able to correctly classify new observations of $X$.

Although it is typically not possible to calculate the loss $L$ directly, some algorithms can be proven to exhibit properties that relate to how close to optimal its losses will be. In this context, one says that a sequence of classifiers $\{c_n\}$ is consistent for a distribution of
Figure 2.1: The AI system can be further broken down into multiple parts. The cross-validation box takes the input data and splits it into $n$ parts, each of which contains training and testing data that is fed to a classifier. The classifier will be trained on the data, make some predictions and output a score, as is further illustrated in figure 2.2. This happens $n$ times, and the scores from each of the train-predict-evaluate-boxes are then aggregated and the results are saved for further analysis.
Figure 2.2: This illustration shows the train-predict-evaluate step of the test environment in more detail. A collection of features with correct labels is fed to a classifier that will attempt to learn the pattern in the training data. Then it will attempt to predict the correct labels for a different data set, known as the testing set, $X_{test}$. These predictions can then be evaluated against the correct labels, $y_{test}$, to produce a score that quantifies how well the classifier performed.
(X, Y) if \( \mathcal{L}(c_n) \xrightarrow{p} \mathcal{L}^* \) as \( n \to \infty \). In other words, some algorithms produce classifiers that (for some types of classification problems) can become arbitrarily close to optimal given that the training set \( D_n \) is large enough.

2.2 Random Forests

There are many algorithms for performing binary classification, but in this thesis we have focused exclusively on the random forest algorithm developed by Leo Breiman, since this is the algorithm currently used in the System. The idea behind the algorithm is to create a large number of classifiers known as decision trees, where each decision tree is constructed from a random subset of the training data. Each decision tree is weak on its own, but the ensemble or forest of trees is strong [28].

In this section we will give an overview of how the random forest is created and present some of its theoretical properties.

2.2.1 From trees to forests

To understand the random forest, we must first understand its primary component, which is the decision tree. A decision tree is a type of classifier that makes a prediction by performing a sequence of decisions, or splits, on the input features. Each node in the tree corresponds to a particular feature and a threshold for that feature. Given a decision tree, the prediction for a set of features can be obtained as follows. Beginning at the root of the tree, look at the corresponding feature and its threshold. If the value of that feature is less than the threshold value, proceed to the left branch, otherwise proceed to the right branch. From the new node, consider its corresponding feature and threshold and proceed left or right depending on its value. This process repeats until an end node, also called leaf, has been reached. The value of the leaf corresponds to the output of the classifier [29].

A decision tree can be constructed or “grown” in a machine learning context by training on a data set \( D_n \). This can be done in several ways. One popular approach, known as Classification And Regression Trees (CART), introduced by Breiman et al. in 1984, works by recursively splitting the training set into smaller and smaller pieces. Each split divides the training set into two subsets and is chosen so that the labels in each subset are maximally separated, or “pure”.

There are multiple ways of defining purity in this context, but in the original formulation a measure known as Gini impurity is used. The Gini impurity \( GI(A) \) of a subset of labels A is defined (for binary classification) as \( GI(A) = 2pq \), where \( p \) and \( q \) are the probabilities that a randomly chosen label in A is positive and negative, respectively. If we were to guess the label of a point in A by choosing label 1 with probability \( p \) and label 0 with probability \( q \), then the Gini impurity \( GI(A) \) is the probability that this guess is wrong [30].

The process of dividing the training sets into smaller and smaller pieces continues until either one of a number of possible stopping conditions is met, or each leaf of the tree is completely pure. Besides the stopping conditions and the choice of purity measure, the behavior of the algorithm can also be adjusted by modifying the rule with which the feature to use for splitting at each node is selected [31].
2.2 Random Forests

A decision tree can theoretically represent any function that can be expressed in propositional logic, however, some functions, such as the majority function, require exponentially large trees to represent [29]. Decision trees are also typically unstable in that small perturbations or changes to the training data may result in a completely different classifier. In this sense they have a high variance, which is undesirable. But a high variance classifier can still have low bias, and this is in fact the case (under some circumstances) for the CART trees. The bias of a classifier is, intuitively speaking, how far away one would guess the classifier to be from optimal if the training data was unknown [32].

Random forest is one way of trying to reduce the high variance of the CART trees without sacrificing the low bias. This is done is by growing a large number of trees, and then to perform new classifications by aggregating the output of each individual tree. Typically, the majority vote is used as the prediction, or one can use the proportion of votes to rank order predictions. In Breiman’s random forest algorithm, the training data for each tree is obtained by sampling the training set with replacement so that a new set is obtained that contains an equal number of samples, but where some of the samples may be repeated several times, and others might not be included at all. This is an idea called bootstrap-aggregating, or bagging for short. In addition to bagging, when growing the trees of the random forest, a subset of features is chosen at random before every split. From these features, the one that minimizes the Gini Impurity is chosen for the split. In the next split, a new subset is chosen at random [28, 30].

To summarize, if we let the training data $D_n$ contain $n$ samples and $d$ features, then each tree in the random forest is constructed with the following three steps:

1. Select a new training set $\hat{D}_n$ by randomly sampling, with replacement, $n$ times from $D_n$.

2. Select $k < d$ features randomly. Out of these, split the node using the feature that minimizes the Gini Impurity.

3. Repeat step two until each node is pure.

In this way, the random forest algorithm incorporates randomness at two levels: both for the training data that is available for each tree, and for the features that are considered at each split.

2.2.2 Theoretical properties of the random forest

Breiman’s original random forest classifier relies on complex data-dependent mechanisms and is therefore quite difficult to analyze, and many of its fundamental properties are not well understood [30]. Nevertheless, some interesting results are still available. Breiman proved that the loss of the random forest classifier, as defined by equation 2.1, converges to a limit as the number of trees in the forest increases, which means that the classifier does not overfit with respect to the number of trees. Another important result, also due to Breiman, gives an upper bound for the loss function that is based on the strength of each individual tree in the forest and the extent to which they are uncorrelated [28].
2. Theory

consequence is that the better the individual trees, the better the forest, and the closer to independent each tree is, the better the forest will become.

Consistency, as discussed in the previous section, is the property that a classifier would become optimal if only it could train on enough data. In this sense, it has been shown that data sets exist such that Breiman’s random forest classifier is not consistent \cite{33}. It is however consistent with some slight modifications to the sub-sampling and splitting rules. For example, Wager et al. developed a variation that they call Causal Forests, which they show to be consistent \cite{34}. This means that while the original formulation of the random forest classifier can not be guaranteed to converge to the optimal classifier on every set of training data, some minor adjustments are possible that allow us to make this guarantee. Of course such results are welcome, but a convergence to something that is optimal in principle might not always translate to an algorithm that works well in practice.

2.3 Cross-validation strategies

We are using the random forest algorithm which, if we feed it data, will produce a binary classifier. Assume now that we have a data set $D_n$ containing a feature matrix $X$ of dimension $n \times d$ and a vector $y$ of dimension $n$ with target labels. In order to evaluate the performance of the classifier, it is customary to partition the available data $D_n$ into a training set $D_{\text{train}}$ and a testing set $D_{\text{test}}$, such that $D_n = D_{\text{train}} \cup D_{\text{test}}$ and $D_{\text{train}} \cap D_{\text{test}} = \emptyset$ (i.e. a proper partition), and then train the classifier on $D_{\text{train}}$ and evaluate it on $D_{\text{test}}$. In this context cross-validation refers to some strategy of choosing multiple such partitions and then averaging the results of the classifier on each of the test sets \cite{29}.

In our case, each observation in $D_n$ consists of features related to a particular patient, identified by a patient ID, at a particular point in time. The question then becomes how exactly one should partition $D_n$. In this thesis, two strategies were implemented and tested: k-fold and rolling forecast origin.

2.3.1 K-fold

In k-fold cross-validation, the data is partitioned into $k$ sets $D_k$ of equal size. The classifier is then trained on $k - 1$ of the sets, $\cup_{k \neq i} D_k$ and evaluated on the remaining set $D_i$. This is repeated $k$ times for $i = 1, \ldots, k$. The idea is illustrated in figure\ref{figure:2.3}. For our purposes, the main advantage of the k-fold cross-validation strategy is that it is quick for small values of $k$: the classifier is only trained $k$ times. The disadvantages relate to how exactly one should perform the partitioning. Three alternatives were implemented: splitting on patient ID, splitting on date, and splitting on rows. The potential drawbacks of these options are discussed in the following sections.

Splitting on patient ID

Splitting on patient ID means randomly placing each patient ID in one of $k$ groups, such that each group contains the same number of patients. If the number of patients is not divisible by $k$, the remainder is dropped from the data set. The primary disadvantage of
2.3 Cross-validation strategies

Figure 2.3: This image illustrates the k-fold cross-validation strategy for $k = 10$, where the split occurs on dates. Each line represents one split of the dataset where the black circles correspond to the training data and the gray boxes represent the test data. The data points are commonly shuffled before being divided into $k$ groups.

this splitting relates to time-sensitive features, especially weather, that are common across several patients.

In a realistic setting, only present and past feature values will be known to the classifier when making new predictions. But if the classifier is trained on the entire time span for a particular patient, as would be the case when splitting on patient ID, then all the weather features will be automatically known for all patients, because all patients have the same weather features for any given day. This means that when the classifier is later evaluated on the test set, it will know what the weather will be in the future. Such foresight could amount to cheating if the weather turns out to have a measurable effect on the risk for hospitalization. Even if we observe improvements due to weather features in this way, we could therefore not be certain that the improvements are not a result of cheating, as it were.

Splitting on date

Unfortunately, splitting the data set on dates is not necessarily an improvement over splitting on patient ID; in fact, the opposite is more likely to be the case. In particular, we must consider that some features contain historical data. Indeed, one of the features is simply the number of days since the previous hospitalization. But if we try to make predictions for day 1 about whether some patient will become hospitalized within 30 days, but our training data contains the information that the patient in question on day 40 was hospitalized 20 days earlier, then in effect the algorithm will know without looking at any other feature the correct label for day 1.
2. Theory

Figure 2.4: This image illustrates the rolling forecast origin cross-validation strategy. Each line represents one split of the dataset where the black circles correspond to the training data, the gray boxes represent the test data, and the light gray circles are data points not included in the split.

Splitting on rows

One row in the dataset consists of all the features belonging to a particular patient at a particular date. Splitting on rows in the k-fold strategy would be to randomly place each such row into one of \( k \) groups. This strategy does not seem to carry any particular advantage over splitting on date or patient ID, while maintaining all the disadvantages. However, the implementation overhead was small enough that we decided to test it anyway, for the sake of completeness.

2.3.2 Rolling forecast origin

To overcome the problems of the k-fold cross-validation strategy, which are mainly related to the temporal dependencies between our data points, one may instead utilize what is known as the rolling forecast origin. The idea is to select a number of dates in the past, and then for each date consider the data and correct labels that were available at that time, and use that as training set. For the test set, we use all the patients that we would have wanted to make predictions for at that date, which amounts to all living patients that have received dialysis within the last two weeks. In other words, we specify an origin from which we make a forecast, and then we let that origin “roll” forward, hence the name of the algorithm. The idea is illustrated in figure 2.4. A possible variation would be to not use all of the data available at a given date for training, but rather discarding data older than, say, one year.

The rolling forecast origin cross-validation strategy overcomes the problems associated with the k-fold strategy by evaluating the classifier in the exact same way as it would be used in reality. The main drawback of this is that the training and testing sets are very unbalanced: the training data contains several hundred thousand examples while the testing data on average contains only about 2,000 examples. For this reason it is important
2.4 Performance metrics

Table 2.1: The confusion matrix is a way of presenting how the predictions from a classifier relate to the true class.

<table>
<thead>
<tr>
<th>True class</th>
<th>Predicted class</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>Positive</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Negative</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

to repeat the process a sufficiently large number of times for different dates in order to obtain an estimate of the average performance. The problem with that however is that the algorithm is re-trained for each example, and training the classifier is compute-intensive. At the end of the day, a compromise must be made between realism, time and accuracy of the performance estimates. On this spectrum, most of the tests done in this work were of the slow, accurate and realistic type. We will call them “kitchen sink tests”, in reference to the kitchen sink realism genre of cinema [35].

2.4 Performance metrics

Suppose now that we have a binary classifier \( c_n \) that is trained on \( D_{\text{train}} \) and makes predictions for \( D_{\text{test}} \). How can we evaluate the performance of these predictions? Perhaps the most obvious metric is that of accuracy, which is the percentage of our predictions that were correct. But accuracy does not paint a complete picture. Imagine for example a very unbalanced data set, where only one in a thousand examples is labeled with a one, and the rest are zeros. Then the trivial classifier that always predicts a zero will have an accuracy of 99.9%, even though it is clearly uninformative.

2.4.1 Confusion matrix and related metrics

For binary classification, there are in fact two ways for a prediction to be correct and two ways for it to be wrong: a positive label can be correctly predicted as positive (true positive or TP) or incorrectly predicted as negative (false negative or FN), and a negative label can be correctly predicted as negative (true negative or TN) or incorrectly predicted as positive (false positive or FP). The proportion of predictions that fall into each of these categories are typically presented in what is called a confusion matrix, as in table 2.1. From each of these four categories one may now derive a number of different performance metrics.

True positive rate, also known as sensitivity, hit rate or recall, is the proportion of positive labels that are correctly identified: \( \text{TPR} = \frac{TP}{P} \). True negative rate, also known as specificity, is the proportion of negative labels that are correctly identified: \( \text{TNR} = \frac{TN}{N} \). Other common metrics include, but are not limited to, precision, negative predictive value, false negative rate, false positive rate, false discovery rate, false omission rate, accuracy, F1-score, Matthews correlation coefficient, informedness and markedness. These are all functions of FN, TN, FP and TP, however. What one chooses to focus on typically depends on how one values the different types of predictions. For example, in a legal system it is usually considered worse to imprison an innocent than it is to let go of someone guilty, so
when passing a sentence we would expect to try to reduce the false positives at the expense of the false negatives. As an example of the opposite situation, a doctor might rather err on the side of caution when ordering further tests of what is probably not a tumor, implicitly worrying more about the false negatives than the false positives.

In this work, a related but somewhat different approach was chosen, and it will be described next.

2.4.2 The receiver operating characteristic

All the previously mentioned metrics rely on having already determined a decision threshold for the classifier. But with a random forest classifier in particular, we obtain for each prediction the proportion of decision trees in the forest that vote for the positive class. In other words, the output of the classifier is a number $p \in [0, 1]$, and the predicted label can be chosen as 1 if $p > \theta$ and 0 otherwise. So the question then is what the best value for the threshold $\theta$ is: each choice will produce a different confusion matrix and therefore also different accuracy, recall, F1-scores and so on.

One way of analyzing the situation is to use what is known as the Receiver Operating Characteristic, ROC. This is the curve one obtains when plotting the true positive rate versus the false positive rate for all $\theta \in [0, 1]$, as can be seen in figure 2.5. The ideal classifier will be “hugging” the top left corner of the graph, and randomly guessing $p$ will correspond to a straight line from the origin to the top right corner.

It is possible to summarize the ROC curve into a scalar metric by integrating the curve. This is known as the area under curve for ROC, or AUC ROC for short. This will always be a number between zero and one, but in practice anything less than 0.5 can be turned into something better than 0.5 by simply choosing the opposite of what the classifier suggests, as in the “so bad it’s good” classifier in figure 2.5. An AUC ROC score of 0.5 is therefore what we expect by chance, and the perfect classifier yields a score of 1.0. Of course some information is lost when looking at the AUC ROC score alone, and it is worth noting that a score of for example 0.75 can be obtained by any number of ROC shapes, each with different properties with relation to the confusion matrix for some threshold.

The AUC ROC score can be interpreted as the probability that a randomly selected positive sample is given a higher score by the classifier than a randomly selected negative sample [36]. AUC ROC can also be shown to be invariant of class balance, which is not the case for other AUC type metrics such as the area under the precision versus recall curve [36]. This means that the shape of the curve is the same even if there happen to exist many more negative examples than positive in the data set.

2.5 What counts as an improvement?

Having chosen a classifier, a cross-validation method and an evaluation metric, the next issue to settle is how much the metric must improve from one set of features to another in order for us to confidently say that the new features improve the classifier. This may seem like a trivial point: why not just take the average test score from each test set in the cross-validation? It is important however to be careful when specifying how much of an
2.5 What counts as an improvement?

Figure 2.5: Illustration of the Receiver Operating Characteristic curve for four hypothetical classifiers. The perfect classifier would hug the top left corner, while randomly guessing is a straight line between from the origin to the top right corner. Anything above the diagonal line is at least decent, and anything underneath the diagonal could in theory be made useful by just doing the opposite of what that classifier suggests.
2. Theory

Improvement in the average is necessary for the test as a whole to be considered successful. This step constitutes the aggregation box in figure 2.1.

2.5.1 The problem

The situation is as follows: keeping the classifier, cross-validation, evaluation metric and everything else constant, we consider a baseline dataset $D_b = (X_b, y)$ and an experimental dataset $D_e = (X_b \cup X_w, y)$, where $X_w$ denotes some additional features, such as weather measurements. For the sake of brevity, we will call the baseline test $b$ and the test using extra features $e$. Now the question is whether the scores from test $e$ represent an improvement when compared to the scores of test $b$.

When using the “kitchen sink” testing, i.e., the rolling forecast origin cross-validation strategy, it turns out that in our case the average improvement is typically about an order of magnitude smaller than the standard deviation of the scores. At first sight it seems that this “improvement” is just a coincidence and that upon repeating the tests we would just as often obtain a decrease in the average scores. But this is not what happens. Instead, the scores vary quite a bit from day to day, but the difference between tests on any particular day is substantially smaller. This means that despite the relatively large variance of the average score itself, a much smaller increase in the average between two data sets can still be significant. This motivates the use of some additional theory which is covered next. The details of how the baseline fluctuates from day to day and how other tests compare to it will be further examined in section 3.1.

2.5.2 Central Limit Theorem to the rescue

Let $x_k = e_k - b_k$ be the difference between the scores for test $e$ and the baseline $b$, for time $k$. We now assume that $x_k$ are observations of independent and identically distributed random variables $X_k$. The distribution of $X_k$ is not known, but it has some mean and variance: $E[X_k] = \mu$ and $V[X_k] = \sigma^2$. We can now use a one-sample t-test to obtain an approximate confidence interval for the average difference $\mu$ between the scores, with the idea that if the interval does not contain zero, we have some confidence that the test $e$ was different (either better or worse) than the baseline $b$.

More specifically, we let the null hypothesis be $H_0 : \mu = 0$ and the alternative hypothesis be $H_1 : \mu \neq 0$, and use the statistic $t = \frac{\bar{x}s}{\sqrt{n}}$, where $\bar{x}$ is the sample mean and $s$ is the sample standard deviation. We then have that $t$ is an observation of the random variable $T = \frac{\bar{x}s}{\sqrt{n}}$ which is approximately $t$-distributed with $n - 1$ degrees of freedom, since $\bar{X}$ is approximately normal due to the central limit theorem. We can now discard the null-hypothesis with a confidence of approximately $\alpha\%$ if $t \in [-t_{\alpha/2}(n - 1), t_{\alpha/2}(n - 1)]$. Here, $t_{\alpha}(n - 1)$ is the $\alpha$-quantile of the $t$-distribution with $n - 1$ degrees of freedom, i.e., it is the number such that $P(T < t_{\alpha}(n - 1)) = \alpha$, and it can be obtained from a table of distributions [37].
Chapter 3

Method

The purpose of this chapter is to describe the method that was used to investigate the effect of adding weather to the System. We begin by examining the labels and features coming from the System, as well as the weather measurements that were collected and processed into additional features. After that, we take a closer look at the test environment that was developed to run the experiments, and we finish the chapter with a summary of the experiments, how they were constructed and what the ideas behind them were.

3.1 Baseline data

The baseline data consists of the features and labels $D_b = X_b, y$ as computed by the System at the end of November 2017. In this work, the baseline is considered fixed, and although we will construct tests where things are removed from or added to the baseline, neither the features themselves nor the labels will be altered.

The $X_b$ matrix is organized such that each row corresponds to the feature values of a single patient at a single date. The column-vector $y$ contains the labels, so that each row in $X_b$ can be associated with a single value in $y$. That value, or label, will be 1 if the patient became hospitalized within 30 days of that date, and 0 otherwise.

The baseline data contains approximately 10,000 patients, and the $X_b$ matrix has on the order of a million rows and several hundred columns, where each column corresponds to one feature. The features of the baseline are primarily constructed from physiological measurements from the patients. These can be divided into five rough categories: features collected during dialysis treatment, features from lab results, features from medications, features from hospitalization data, and others. All the patients considered in this way receive hemodialysis at clinics in the vicinity of Cleveland, Ohio.

The time period for the data stretches from 2011-01-02 to 2017-11-26. Each day has on average about 2,000 “active” patients, meaning living patients that have received dialysis within the past two weeks, and are not currently hospitalized. The average ratio of positive
3. Method

Figure 3.1: AUC ROC for the baseline evaluated with the rolling forecast origin cross-validation on 48 evenly spaced Sundays between 2013 and 2016.

to negative labels is around 20%, which means that if everything else was equal, an active patient would have approximately a 20% risk of becoming hospitalized within 30 days.

The default baseline settings were chosen as the complete set of features from the System, using the entire dataset and evaluating it using the kitchen sink cross-validation strategy as explained in section 2.3.2. The test period is chosen as 48 evenly spaced Sundays between 2013-04-01 and 2016-12-01, and has been kept the same for all kitchen sink tests in order to be able to more easily compare scores. The reason for not testing on earlier dates was to allow some training data to accumulate before trying to make any predictions. The data for the year 2017 was held-out to facilitate a final evaluation of any potential findings on unseen data that have not been polluted by other tests.

The fluctuation of the AUC ROC scores of the baseline is illustrated in figure 3.1. For each date in the time period, a new random forest is grown with the information known at that time, and predictions are made for the patients that were active at the time, meaning they were alive and on dialysis, but not hospitalized. These predictions are then summarized with the AUC ROC score, which is what is plotted in figure 3.1.

As we can see, the scores fluctuate quite a bit from day to day. It turns out that there is also some variance in the predictions made for a single day, due to the stochastic nature of the random forest algorithm. To measure this variance, the same test was repeated 48 times on the same date, chosen arbitrarily as 2016-04-10. From this, we can estimate the distribution of the scores on a single day by plotting the histogram, as shown in figure 3.2. As we can see, the distribution appears to be approximately normal within the same day. The standard deviation of the average score is approximately ten times larger between different days than it is within days.
3.2 Weather data

The following section will describe the weather data that was collected, where it came from and how it was processed into useful features for the classifier. The focus will be on the latter part, which is also known as feature extraction.

3.2.1 Collecting and cleaning the weather data

The National Centers for Environmental Information (NCEI) was chosen as the supplier of weather measurements, primarily because they release complete historical records of all the relevant measurements free of charge [38]. This is in contrast to many of the competitors, most of whom either lack the historical records needed for this project, or charge premium fees for such access.

The datasets of most obvious utility for this thesis were found within what NCEI calls Quality Controlled Local Climatological Data (QCLCD), which contain hour-by-hour updates on dozens of different parameters from thousands of ground based weather stations across the globe, some of which have made recordings since the beginning of the 20th century [39]. The time span of interest to us however was the period 2011 to 2018, and because all the patients in the baseline receive their dialysis treatment in Cleveland, Ohio, we were further limited to only a handful of weather stations. These were Cuyahoga County Airport, Cleveland Burke Lakefront Airport and Cleveland-Hopkins International Airport. Further inspection revealed the data to be incomplete for Cuyahoga County Airport for the desired time period, and the Cleveland Burke Lakefront data was later found not to mea-

![Figure 3.2: Histogram of 48 runs of the baseline test on the same day, 2016-04-10. The distribution of AUC ROC scores appears to be approximately normal.](image)

Distribution of AUC ROC scores for 48 repeats on 2016-04-10

[Distribution of AUC ROC scores for 48 repeats on 2016-04-10]
sure snow. The data captured at Cleveland-Hopkins International Airport weather station appeared to be the most complete, and the others were therefore discarded.

The data captured by the weather station includes 23 different weather measurements together with special codes for each measurement indicating missing data or that the station is in need of service. Also included was a coded summary of the “weather obscurations”, including the presence of fog, rain, volcanic ash, dust storms and so on; neither the special codes nor the weather obscurations were used in this work.

The weather measurements were further trimmed down to only those that seemed the most likely to be able to affect patient health; measurements such as sun rise and wind direction were discarded. This brought the total number of basic weather measurements down to the following ten: daily maximum, minimum and average dry-bulb temperature, wet-bulb temperature, dew point, snowfall, snow depth, total precipitation, atmospheric pressure and daily average wind speed.

The first three temperatures are all so-called dry-bulb temperatures, which means that the thermometer has been shielded from radiation and moisture. Further analysis of the weather measurements revealed a handful of outliers as well as some missing data. The missing data was presumably due to the weather station occasionally requiring service or repairs. Since there were never any significant stretches of missing information, we used linear interpolation to fill in the blanks. The average temperature was found to almost always be exactly the average of the maximum and minimum temperatures, and so a small and simple dimensionality reduction was possible by replacing maximum and minimum temperatures with the daily deviation from the average.

Climatological measurements such as pollutants and similar were not included in the QCLCD, and due to time constraints, no further efforts were made to obtain such data.

### 3.2.2 Weather features

After having gathered and preprocessed some weather data, the next step before feeding the data to the classifier is called feature extraction or engineering. It is the process of trying to modify, combine or transform the raw data in various ways so that they become more useful to the classification algorithm. For the weather, we have limited our efforts to two major types of feature extraction: features derived from domain expertise (or intuition, in our case) and features constructed by viewing the measurements as time-series.

#### Apparent temperatures

Instructed by the literature from section 1.2, the measurements that most often appear to be correlated with disease are temperature and humidity. These can be combined, together with wind speed, into what is known as apparent or felt temperatures. These are measures attempting to model the fact that some weather conditions might feel warmer or colder than the actual dry-bulb air temperature, as a result of wind speed and humidity. During cold weather, high wind speeds will make the temperature feel even colder, because of increased convection [40]. On the other hand, when the temperature is high, a higher humidity will make the temperature feel even higher, because a high relative humidity will reduce the rate at which the body can lose heat through perspiration [41]. To capture these effects, wind chill and heat index were added as features.
There are several competing formulae for wind chill, but in this work we used the one derived by Osczevski et al. in 2001, which is also used by the National Weather Service in the United States [42, 43]. The formula is given by equation (3.1) where \( WCT \) is the wind chill temperature, \( V \) is the wind speed in mph and \( T \) is the air temperature in °F [43].

\[
WCT = 35.74 + 0.6215T - 35.75V^{0.16} + 0.4275TV^{0.16}
\]  

(3.1)

In order to calculate the heat index, we first need the relative humidity. Relative humidity is defined as the ratio of the partial pressure of water vapor in the atmosphere to the equilibrium vapor pressure of water above a flat surface of water at a given temperature. These pressures can be approximated by the August-Roche-Magnus formula since we know the temperature and dew point temperature, which is the temperature to which air must be cooled to become fully saturated with water [44]. The final formula for calculating the relative humidity is then given by equation (3.2) where \( \psi \) is the relative humidity, \( T \) and \( T_d \) is temperature and dew point temperature respectively, both in °C.

\[
\psi = 100 \exp \left\{ \frac{17.625T_d}{243.04 + T_d} - \frac{17.625T}{243.04 + T} \right\}
\]  

(3.2)

The heat index, similar to wind chill, is a result of extensive biometeorological research, and it is a model that actually incorporates numerous variables, such as the surface area of the skin of a person, the clothing coverage, the clothing resistance to heat transfer, and several others. In order to simplify the model, however, all values except for relative humidity and temperature can be given some assumed magnitudes. The resulting formula is then given by equation (3.3) where \( HI \) is the heat index or apparent temperature, \( T \) is the dry-bulb temperature and \( \psi \) is the relative humidity as an integer between 0 and 100. This time all the temperatures are in °F [45, 46].

\[
HI = a + bT + c\psi + dT\psi + eT^2 + f\psi^2 + g\psi T^2 + hT\psi^2 + iT^2\psi^2
\]  

(3.3)

The coefficients \( a - i \) are taken to be \( a = -42.379, b = 2.04901523, c = 10.14333127, d = -0.22475541, e = -0.00683783, f = -0.05481717, g = 0.00122874, h = 0.00085282 \) and finally \( i = -0.00000199 \).

**Time-series features**

Another approach for engineering new features from the weather measurements is to consider the temporal aspect and view the measurements as time-series. The idea is that past values might have some important influence on the classification problem at hand. For example, it might be that the exact temperature today is of little relevance for the hospitalization of ESRD patients, but that a long streak of cold weather over the past two months is rather important. Alternatively some sudden change in humidity, let’s say, is more significant than the value of the humidity itself. There are literally thousands of different documented methods with which to turn a time-series into features, but in the interest of time, only two were implemented and tested [47]. These two were rolling averages and discrete Fourier coefficients.

The rolling averages are straight-forward to implement. For any given measurement, we simply take the average over the past \( x \) measurements, and this average then becomes a
new feature. Of course the immediate question then is what the value of \( x \) should be, and which feature to average over. Although several variations were in fact tried, we settled in this thesis on using the values 7, 14, 28, 56, 112, 224 and 365 for \( x \), and to evaluate these averages for all the other 12 weather features, for a total of 84 new features. Using so many different variations might seem excessive, and we do introduce many highly correlated features, which is of course not optimal. But figuring out exactly which combinations are best is a notoriously difficult problem, while the random forest algorithm appeared to be quite insensitive to the inclusion of additional features. It should also be noted that a correlation between features does not imply that any of the features are redundant [48].

The Discrete Fourier Transform (DFT) is a transform \( \mathcal{F} \) of a sequence \( \{x_n\} \) of \( N \) numbers into the frequency plane \( \{X_k\} \), and it is defined by equation 3.4.

\[
X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-2\pi i kn} \tag{3.4}
\]

By looking at the DFT of a time-series, we get insight into its spectral properties and along which frequencies the time-series fluctuates the most. In our case, we want to turn a variable length sequence into a fixed (short) length sequence, because each number in the new sequence will be a feature on its own. Relatively advanced methods for selecting only the \( k \) most representative DFT coefficients have been developed, but as a first approximation, taking the first few coefficients is often pretty good [49, 50]. The DFT coefficients are complex valued, so to work around that, we used the absolute value of the coefficients.

For the DFT coefficients we also need to decide which features to calculate the DFT for, and how many of the coefficients we want to keep. With a similar line of reasoning as with the rolling averages, the first 10 coefficients were used for each of the 12 non-time-series features that were used, for a total of 120 new features.

### 3.3 Test environment

A substantial amount of time was spent designing and implementing a test environment that could be used to evaluate the performance of the System when adjusted in different ways. The scope of this endeavor, initially born out of necessity in order to establish a baseline, came to grow into a more elaborate system in which various aspects of the classification problem could be easily tweaked, tested and the results compared. Because this was a major part of the actual work involved in this thesis, and perhaps also the foundation of a test environment that can be used to evaluate other types of features and settings, this system will be described in more detail.

The original idea for the design of the test environment came from the realization that in order to thoroughly test certain ideas, many test cases would be required. With many tests, however, it would be difficult to keep track of exactly what was tested and what the results really were, especially since we might not know in advance what aspects of some test will be interesting to compare with future tests. A poorly organized system would necessitate substantial bookkeeping efforts, so some alternative is clearly to be preferred.

For these reasons, the approach was taken to let the code, as far as possible, document the tests. Each test is therefore an instance of a data structure that essentially lists the
3.4 Summary of the experiments

The experiments or tests that were performed in this work can be roughly divided into three categories: hyper-parameter optimization, cross-validation tests, and filtering tests. The first two kinds of tests are concerned with the AI box in figure 3.3. During hyper-parameter optimization we try to find good settings for the random forest classifier, and from the cross-validation tests we try to determine the best method of evaluating the classifier. The filtering tests are those adjusting the filter box in figure 3.3 and control exactly which baseline and weather features to use as well as which patient clusters to test on.

In this section we present some of the tests from each of the three categories and explain why we are running these tests, and in chapter 4 the results of the tests are examined and discussed.


3.4.1 Hyper-parameter optimization

Most classifiers provide settings, called hyper-parameters, that affect some details of the workings of the algorithm, and random forest is no different. Hyper-parameters can typically be divided into two categories: those that affect the classifier scores, and those that affect the run time of the algorithm. In this thesis, we have used the python package scikit-learn and its implementation of the random forest algorithm. It provides 17 different hyper-parameters, 12 of which can have any plausible effect on the classification performance. This number can be further trimmed down with the application of some theory: for instance, it can be shown that increasing the number of decision trees in the forest does not reduce the classification performance of the algorithm, although it does make the training slower. Past a certain number of trees, nothing is gained in practice [28].

It is possible that as long as the hyper-parameters are held constant, any observed change in performance due to the addition or removal of certain features can be explained entirely by the hyper-parameters. For example, if the hyper-parameters were perfectly tuned to the baseline, then they are unlikely to remain perfect once we’ve added weather features to the baseline. Observing a reduction in the average scores in such a scenario doesn’t necessarily mean that the new features are really to blame: It is possible that adjusting the hyper-parameters could lead to scores that are better than the baseline. In any case it is quite unlikely that any given setting is equally good or bad for any two sets of features, so it makes sense therefore that we at least gain some understanding of how the hyper-parameters affect our scores.

One tempting strategy is to optimize the hyper-parameters for each new subset of features that is tested. This is not feasible however, because actually finding the optimal hyper-parameters can be quite difficult. There are several approaches for how to find good parameters. The most straightforward is to manually try different combinations, but this can be both difficult and time-consuming, and typically requires a lot of experience working with the classifier.

Another popular approach is called grid-search, and it is basically a brute force search over the entire space of parameters, sampled at some predefined intervals. Grid-search has the advantage of being thorough, but also the crippling disadvantage of being computationally infeasible even for relatively small search grids.

A third approach to hyper parameter optimization is random search, which works by randomly sampling the space of possible parameters a certain number of times, and run the cross-validation on each sample. Random search can be quite good at finding approximately optimal solutions without having to try every possible combination. If given the same computational resources, random search is likely to outperform grid-search [51].

In this project, the random search strategy was used to attempt to optimize the hyper-parameters. In fact, two attempts were made. In the first attempt, the kitchen sink cross-evaluation strategy was used to evaluate each sampled parameter on 10 different dates. The sample space contained five different parameters, and it was sampled 100 times for the baseline dataset and 100 times for the baseline in combination with 12 weather features. The results are presented and discussed in section 4.1.

In the second attempt, the parameter space was reduced to four parameters, and the cross-validation was changed to use a look-back of only 90 days. The cross-validation also used the “standard” 48 dates, which served to reduce the variance of the quality of
3.4 Summary of the experiments

Each test. In total, 400 samples of the parameter space were evaluated: 200 for the baseline dataset and 200 for the baseline in combination with all 216 weather features. The use of a look-back of 90 meant that even though the hyper-parameter optimization test consisted of almost 10 times more evaluations, it still ran 10 times faster than the first attempt. This turned out to have made a big difference, as we will see in section 4.4.

As a result of the hyper-parameter optimization, two new parameter settings were defined and evaluated. To keep track of this, we let $P_0$ denote the original settings used by the System, while $P_1$ and $P_2$ denote the parameters derived from the second hyper-parameter tests. $P_1$ corresponds to the parameters that appeared to be most suitable for the baseline data, and $P_2$ corresponds to the parameters most suited to the addition of the full complement of weather features. Due to the competitive nature of the field, the details of the parameter settings will not be disclosed in this report.

3.4.2 Cross-validation tests

Throughout most of the project, the rolling forecast origin cross-validation strategy, or the kitchen sink tests as we have called it, has been employed. The main reasoning behind this decision was that it seemed to be the most realistic way of evaluating the performance of the classifier: we examine the average behavior of the classifier as it would have performed on a number of different historical dates. Due to the variance that was observed in section 3.1, it appeared necessary to run the kitchen sink tests a fairly large number of times: all such tests were therefore run on the same 48 evenly spaced Sundays between April 2013 and December 2016. The data for the first 40 Sundays of 2017 was held-out for validation tests, and was used in section 4.5.

In the name of thoroughness, several alternative cross-validation techniques were also investigated, as was described in section 2.3. The k-fold cross-validation strategy was tested for $k = 5, 10$ and $20$, and for splits made on patient ID, dates and rows of the $X$ matrix. We also tested the rolling forecast origin strategy with a twist: rather than using all the data that is available for training at a particular date, we can restrict ourselves to using only data going back a certain number of days. We dubbed this modification “look-back”, and it was tested on the baseline data for values of 7, 15, 30, 45, 90, 180, 365, and 730 days.

3.4.3 Filtering

Four different ways of filtering the data will be considered in this report, and each can be adjusted independently of the others. We can select subsets of weather features, baseline features, random features and patient clusters. Together with the parameter settings, we then have five dimensions along which we can move, and we can think of a single test case as occupying a point in this five-dimensional space. In order to more easily describe the particular choices for each dimension, these coordinates has been encoded as a quintuple $W : B : R : C : P$, where each letter is followed by a number that corresponds to a particular setting for that dimension. Each dimension will have a 0-value that corresponds to what the setting is for the baseline along that dimension, and in order to compress the syntax, we say that the 0-value for a dimension is used when the dimension is not
3. Method

mentioned. The special case in which every dimension uses the 0-value will simply be referred to as the baseline.

For the weather features, we have defined six groups of features: \(W_0, W_1, W_2, W_3, W_4\) and \(W_5\), where \(W_0\) by convention corresponds to using no weather features at all. The baseline features were also divided into five groups: \(B_0, B_1, B_2, B_3\) and \(B_4\), where in this case \(B_0\) means that all the baseline features are used. Furthermore, the patients were divided into 9 clusters: \(C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7\) and \(C_8\), where \(C_0\) means all the patients are used. The random features, the use of which will be further motivated in section 4.3.4, consist of adding a number of features where the value at each day is a random number between 0 and 1. Rather than enumerating all possibilities for the random features, we simply denote that dimension by \(Rx\), where \(x\) is any positive integer or 0, in which case no random features are used. With this notation, there are 810 possible tests for each choice of random features. Of course the number of tests grows geometrically as more and more groups are added to each dimension, and this is indeed the primary reason why we limited our efforts to these fairly modest numbers.

To summarize the notation, one test can for example be described as \(W_1: B_2: C_0: R_0: P_1\), which would be shortened to \(W_1: B_2: P_1\), and which means that we test weather features \(W_1\) together with baseline features \(B_2\) using parameter settings \(P_1\), on all patients \((C_0)\) and using no random features \((R_0)\). The collection of tests where we use the subset \(B_3\) of baseline features with parameter settings \(P_0, P_1\) and \(P_2\) will be denoted by \(B_3: P_0\). Having described the notation, we will proceed to explain what each dimension entails in some more detail, except for the parameters, which we were, as previously mentioned, asked to keep secret.

**Weather groups**

The weather features were divided into four categories: basic, advanced, means and DFT. The basic weather features are the 9 raw measurements from which the rest of the features are also derived. These measurements are: daily average dry-bulb temperature, daily deviation of temperature from mean, wet-bulb temperature, dew point, snowfall, snow depth, total precipitation, atmospheric pressure and daily average wind speed.

The advanced weather features are wind chill, heat index and relative humidity. The means features are the 84 rolling averages of the 12 basic and advanced weather features, where the averages are taken for time windows 7, 14, 28, 56, 112, 224 and 365 days, as described in section 3.2.2. Finally, the DFT features are the 120 features obtained by taking the absolute value of the first ten DFT-coefficients of each of the 12 basic and advanced features, treating them as time-series.

The weather categories are summarized in table 3.1.

**Baseline groups**

The baseline features can be divided into four categories, and from these categories we can then select some combination of features to use. The categories are: dialysis, medicine, hospitalizations and other.

The dialysis features are derived from data collected during dialysis treatment. The medicine features are constructed from information about the type of medicine that the
Table 3.1: The weather features can be grouped into four categories. From these categories, any combination of features can be selected. This table defines six selections of weather features and which features should belong in each selection.

<table>
<thead>
<tr>
<th>Feature groups</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>W3</th>
<th>W4</th>
<th>W5</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Advanced</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Means</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFT</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The patient is using. The hospitalization features contain information about when the patient has been hospitalized in the past. Finally, the other features contain all the features that didn’t fit in any of the first three categories. These baseline groups are summarized in Table 3.2.

Table 3.2: The baseline features can be grouped into four sub-groups. From these sub-groups, any combination of features can be selected. This table defines five selections of baseline features and which features should belong in each selection.

<table>
<thead>
<tr>
<th>Feature groups</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialysis</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medicine</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hospitalizations</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Patient clusters

The patient clusters follow a slightly different formula than the baseline and weather feature groupings. The main idea behind the clusters, as was discussed in section 1.3.2, is that some groups of patients might be differently affected by weather than others. There were several ideas for how this could be done, but we ended up focusing on the idea of using comorbidities.

Unfortunately, the situation was made difficult by the fact that our dataset does not contain a perfect record of which patient is suffering from what. What was done instead was to look at the hospitalization records, which contained one field in particular known as “hospital admission diagnosis”. This field is a rough classification of the type of problem the patient is suffering from, although it may not be the underlying cause of the symptoms.

The hospital admission diagnosis can take one of eleven possible values, including a generic “Other” category. In addition to the hospital admission diagnosis, we also made use of a field called “primary complaint”, which describes, in free text, the symptoms that the patient presented with.
3. Method

The patient clusters were now constructed from the hospital admission diagnosis as well as the primary complaint in the following way. A patient will be considered to belong to cluster $X$ if he or she has, at any point in the past, been hospitalized with the admission diagnosis $X$, or if $X$ is present in the primary complaint. In this way, we looked at what seemed like the six most plausible complications that might have been influenced by weather, according to a registered nurse at Lytics. These complications were: shortness of breath (SOB), cardiovascular problems, respiratory problems, vascular access problems, sepsis and musculoskeletal problems. From these six complications, we can further isolate patients that at some point have suffered not just one, but several of these complications. In this way we arrived at eight different patient clusterings, which are summarized in table 3.3.

Table 3.3: A few possible complications have been gathered into six different groups. Patient clusters can now be defined by those who have experienced some combination of complications. Cluster $C^7$, for instance, contain patients with a history of cardiovascular and respiratory problems. Group $C^0$ contain all patients.

<table>
<thead>
<tr>
<th>Complications</th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SOB</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cardiovascular</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Respiratory</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Vascular Access</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sepsis</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Musculoskeletal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Chapter 4
Results and discussion

During this project, well over a hundred distinct test cases were defined and executed. In this chapter we present the highlights of those tests in roughly the order they were tested. This presentation begins with the results of an initial attempt at optimizing the hyper-parameters of the model, followed by various tests comparing the baseline features to the weather features. We then digress for a moment on the results of adding random features to the classifier, before optimizing the parameters once more and then re-running several of the tests. The final results are presented in section 4.5.

4.1 Hyper-parameter optimization, first attempt

In this thesis, we have employed the random search algorithm in our attempts of optimizing the hyper-parameters. This optimization was tried at a fairly early stage in the project, and the results were not particularly instructive. In short, out of the five different parameters that we attempted to optimize, none seemed to show any clear correlation with the test scores at all. Furthermore, the search, since it employed the kitchen sink cross-validation technique, was extremely slow. Each sample of the parameter space was tested on ten different Sundays, and in total 200 samples were tested, 100 for the baseline dataset and 100 for a dataset with some additional weather features. The entire experiment took almost a week to run.

Due to the apparent difficulty and large computational costs associated with fine-tuning the hyper-parameters, it was decided not to further pursue such optimizations and simply use the same settings as in the System. It turned out that this was a mistake, however, and at the end of the day, a bit more patience in this area might have saved a lot of time later down the line. We shall return to this topic in section 4.4.
4. Results and discussion

4.2 Tests of cross-validation strategies

Two types of cross-validation strategies were tested, each with a few different settings, as described in 3.4.2. In this section we present the results of these tests and discuss some of the implications.

4.2.1 Results of k-fold cross-validation

As we elaborated in section 2.3.1, it was our expectation that splitting on dates and on rows would be a sub-optimal strategy for cross-validation due to the temporal sensitivity of the data and the features used. This suspicion was empirically confirmed by the fact that both strategies gave very strong AUC ROC scores in excess of 0.97. Surely such results are far removed from the capabilities of any classifier on this data, so the only real explanation is that the algorithm is “cheating”: knowing what happens to a patient tomorrow will make it easier to predict the risk of hospitalization today. Drawing conclusions about the generalization error based on such tests seemed dubious at best.

More interesting was the k-fold cross-validation where the splits occur on patient ID. The algorithm might still be able to “cheat” by knowing about future weather, but all patients that it makes predictions for will be completely unseen during training. It turns out that the classifier performs significantly worse in this strategy, with an average AUC ROC reduction of about 0.02. Furthermore, the variance of the score appears to be increasing as the number of folds increases, which is not only counter-intuitive but also detrimental to the accuracy of the test. The main advantage of the k-fold strategy compared to the kitchen sink tests is that k-fold is fast for small values of $k$. However, if the variance is large, it must be combated by repeating the test many times, which defeats the purpose of using k-fold to begin with.

4.2.2 Kitchen sink varieties

A variation of the rolling forecast origin cross-validation strategy was implemented and tested with the intent of sacrificing some accuracy for the benefit of speed. We called this variation look-back, since we use only the data from the $n$ most recent dates exclusively, rather than training the classifier with all the available data at a particular date.

The look-back variation was tested on the baseline data for 7, 15, 30, 45, 90, 180, 365, and 730 days. When examining the training times of the classifier, plotted in figure 4.1, we can see that it scales linearly with the amount of training data. The baseline kitchen-sink test with the standard 48 test dates takes approximately one hour to complete, which can be compared to the 100 seconds of the 7 day look-back variation. The deviation of the look-back scores compared to baseline can be seen in figure 4.2. Somewhat surprisingly, the difference is quite small. In fact, it appears that using only the last three months of data is sufficient to obtain scores similar to the baseline, while still benefiting from an 18-fold speed-up. This will become important when we return to the hyper-parameter optimizations in section 4.4.

It should be noted that several of the features at any given date actually contain historical information that may have accumulated over an extended time period. This is clearly
4.3 Weather

We will now examine the results of some of the tests that included weather features. We begin by looking at how different versions of weather compare to the baseline, and then move on to investigate our various subsets of baseline features with and without weather features, as well as the effect of limiting the dataset to only certain patient clusters.

4.3.1 Weather features versus baseline

Plotting the deviation from baseline for the tests $W1 - 3 : B0 : C0$ on the 48 dates in the kitchen-sink cross-validation, as in figure 4.3, it appears as though all five collections of weather features give a small but significant improvement compared to the baseline. The rolling averages in $W3$ in particular look promising.

Figure 4.1: The (cumulative) training times of 48 iterations of the baseline test using rolling forecast origin with different values for the look-back parameter. The plot indicates that the training times scale linearly with the amount of training data.
4. Results and discussion

Figure 4.2: This plot shows approximate error bars for the mean of the average difference between the AUC ROC scores of baseline using all the training data compared to different values of look-back. So for instance, training only on data from the past month and a half (look-back=45 days), the average AUC ROC score is reduced by between 0.010 and 0.015 (approximately 95% confidence). The plot indicates that there is little need to use training data extending back more than about one year.
Figure 4.3: This plot shows approximate error bars for the mean of the average difference between the AUC ROC scores of baseline compared to the five versions of weather, \(W1 - 5\). All five tests indicate an increase relative to the baseline, which is significant with at least 95% approximate confidence. The best improvement appears to be the one from version \(W3\).
4. Results and discussion

Figure 4.4: This plot shows approximate error bars for the mean of the average difference between the AUC ROC scores of baseline compared to weather version $W_2$, when evaluated only on patient clusters $C_1$ – 8. The plot indicates a small but significant improvement on clusters $C_3$ and $C_4$, corresponding to respiratory and vascular access problems, respectively.

4.3.2 Weather features versus baseline on subset of patients

We will now look at the experimental effects of weather when compared to specific subsets of patients. For these tests, we used weather features $W_2$. Because the size of the training data is reduced when we look at subsets of patients, we can’t compare the results directly to baseline. Instead we compare weather version $W_2$ with $W_0$, i.e., with and without weather features. Specifically, we compare the tests $W_2 : B_0 : C_1$ – 8 with $W_0 : B_0 : C_1$ – 8. The results are shown in figure [4.4]

None of the patient clusterings appear to have been significantly improved by the addition of weather features, with the possible exception of clusters $C_3$ and $C_4$, which corresponds to patients with respiratory and vascular access problems, respectively. The improvement is quite small, however.
4.3 Weather

Figure 4.5: This plot shows approximate error bars for the mean of the average change in AUC ROC scores when weather version $W_2$ is added to various subsets of baseline features, $B_1 - 5$. The plot suggests that when using only hospitalization or medicine features ($B_2$ or $B_3$), there is a remarkably strong improvement in scores when weather is added to the mix.

4.3.3 Weather features versus subsets of baseline features

Turning now to subsets of baseline features, the $B$-groupings in table 3.2, we will compare the results of tests $W_2 : B_1 - 5 : C_0$ with those of $W_0 : B_1 - 5 : C_0$. The results of these tests are shown in figure 4.5.

Contrary to what one might expect, a substantial improvement in AUC ROC scores for groups $B_2$ and $B_3$ can be observed. Could this be evidence in favor of the “multiple straw hypothesis” from section 1.3.2? It would appear that whatever information is contained in the dialysis and remaining features are not improved by further adding weather, whereas medicine and hospitalization features are, so perhaps there is some informational overlap between weather, dialysis and remaining features. This hypothesis was swiftly debunked. Figure 4.6 illustrates what happens when random noise is added to the hospitalization features in place of the weather features. This corresponds to the tests $W_0 : B_3 : C_0 : R_0 - 640$.

The classifier becomes stronger, apparently, the more noise we add. The same thing happens when noise features are added to the entire set of baseline features, as can be seen in figure 4.7. The increase is an order of magnitude smaller, but it still appears to be significant.
Figure 4.6: This plot shows approximate error bars for the mean of the average difference between the AUC ROC scores of the classifier when trained on hospitalization features ($W_0 : B_3 : C_0$) compared to when training on hospitalization features in addition to different numbers of uninformative noise features. The plot indicates that the classifier performs better when we complement it with noise, and that these improvements are similar to those observed when adding weather.
Figure 4.7: This plot shows approximate error bars for the mean of the average difference between the AUC ROC scores of the baseline ($W_0 : B_0 : C_0$) compared to when adding different numbers of uninformative noise features. Although the improvement is substantially smaller than that observed for the hospitalization features, it is still significant, and it is comparable to the increase that has been measured due to weather features.
4. Results and discussion

4.3.4 The placebo features

In medicine, a patient may sometimes become better when treated with placebo; a substance, such as a sugar pill, that has no therapeutic effect. This is known as the placebo effect. In a similar way, it would appear that adding uninformative features can act as a kind of placebo for the classifier, improving it where no improvement ought to be found. It is not entirely certain what the reason behind this unexpected turn of events might be, but some hypotheses have been entertained as at least plausible.

The situation does have the taste of overfitting, which is, roughly speaking, when the model is more complicated than is warranted by the data, such that patterns are learned that do not generalize. This is conceptually similar to what happens when one recognizes faces in the clouds. It might be that the random forest classifier, with the baseline features, is seeing faces in the clouds, so to speak, and that by adding noise, the erroneous patterns are obfuscated and the faces disappear.

A related idea comes from examining the random forest algorithm itself. The algorithm fundamentally works by growing a large number of individually weak decision trees, and then averages the votes of each individual tree. If each decision tree is independent of the rest, the law of large numbers guarantees that the combined votes converge to the true mean. But in our case, if many of the features are highly correlated, the individual decision trees in the random forest will not be independent. The idea then is that by adding placebo features, we might be making the decision trees closer to independent at the cost of making their individual predictions less accurate. But if the increased independence is somehow worth more than the lost accuracy, the final results are improved. Supporting this view is the fact that the hospitalization features are rather highly correlated, and that’s where the largest improvement from placebo features was found.

In order to investigate the phenomenon further, an artificial data set was constructed and analyzed, and it was found that the placebo phenomenon could indeed be replicated on the artificial data set.

Regardless of the true reasons behind the observed placebo phenomenon, one important question is if the improvement from the placebo features can be obtained without deliberately feeding misinformation to the classifier. In particular, the situation begs the question if there are some hyper-parameter settings that might eliminate the placebo effect.

With the artificial data in place, this question was significantly easier to answer empirically, since optimizing the hyper-parameters on this small set requires orders of magnitude less computational power. Indeed, after a significantly large random search over the parameter space, the best settings were not affected by placebo features. So even if we still might not know exactly why the placebo effect was there to begin with, we now know at least one way of dealing with it in practice.

4.4 Revisiting hyper-parameter optimization

Having optimized away the placebo effect in the artificial data set, it was decided to revisit the hyper-parameter optimization. Because of the dramatic speed-up effect of the look-back parameter in the rolling forecast origin cross-validation that was observed in section 4.2.2, the number of parameter samples could be increased while still significantly
4.5 Revisiting weather

-decreasing the computational cost. The parameter space was also focused more narrowly on the parameters that seemed to have had the largest effect on the artificial data set.

Two optimization runs of 200 iterations were performed, each on the standard 48 dates but with a look-back of only 90 days. One of the runs used the baseline data set, and the other used the baseline in addition to \( W_5 \), which included all weather features. The tests took approximately 16 hours each to run, and this time, three of the four parameters that were sampled appeared to show a correlation with the average AUC ROC scores. This correlation is illustrated by the scatter plots in figure 4.8, which show the AUC ROC scores of each test plotted against the value of each of the four parameters.

4.5 Revisiting weather

From the two parameter optimization sessions in the previous section, two alternative parameter settings were chosen for evaluation. We label these new settings \( P_1 \) and \( P_2 \) and compare them with the default settings \( P_0 \). Because we at this point have trained the algorithm extensively (400 times in the parameter optimization sessions) on the same data set, there is always the risk that the patterns we might find are specific to this data and do not generalize well in reality, a phenomenon sometimes known as peeking [29]. To avoid this, the new parameter settings were evaluated on a held-out data set consisting of the first 40 Sundays of 2017.

A total of 21 new test cases were defined, focusing on determining the effects of weather versions \( W_2-4 \) when compared to \( B_0 \) and a small and large number of additional placebo features, for each of the three different parameter settings (the default settings from before and the two new settings found as a result of the parameter optimization). The test cases were \( W_0 : P_0 - 2 : R_0, 10, 216 \) and \( W_2 - 5 : P_0 - 2 : R_0 \). The reason for comparing with 216 random features in particular is that \( W_5 \) contains 216 different weather features.

Comparing the scores of the baseline for each of the three parameter settings using 0, 10 or 216 placebo features, i.e., comparing \( W_0 : P_0 : R_0 \), we can see that \( W_0 : P_1 : R_0 \) has the best average AUC ROC score. Furthermore, any addition of placebo features to \( P_1 \) or \( P_2 \) only seems to decrease the scores. This suggests that the parameter settings \( P_1 \) is the best one we’ve seen so far for the baseline, and also that it is unaffected or even adversely affected by the addition of placebo features. The average mean of the differences from each of these tests to \( W_0 : P_0 : R_0 \) is shown in figure 4.9.

Comparing the different weather versions for all three parameter settings against the optimized baseline \( W_0 : P_1 : R_0 \), as in figure 4.10, we see that none of the tests utilizing weather features constitute a significant improvement over \( W_0 : P_1 : R_0 \). In fact, most of the time, adding weather significantly reduces the average AUC ROC scores.

In conclusion, it would appear that whatever improvement was observed as a result of adding weather to the model was merely an artifact of poorly adjusted hyper-parameters for the classifier, and that once those parameters were tuned, the improvement due to weather disappeared completely. Of course any improvement of the system is potentially of great value, even though the underlying reason for the improvement was not what was originally expected. Metaphorically speaking, the picture on the TV might have improved when it was kicked in that “special” way, but once the antenna was adjusted, the picture became clearer still, and kicking the TV no longer made an impression.
Figure 4.8: Scatter plots showing the score of 200 randomly sampled hyper-parameters plotted against four different parameter values. The random search was performed twice with 200 samples each: once using the baseline features, shown on the left side, and once using baseline as well as weather features, shown on the right side. The plots indicate a correlation between parameters a, c, d and the AUC ROC score.
Figure 4.9: This plot shows the average deviation in AUC ROC scores from $W0 : P0 : R0$ for different parameter settings and different numbers of placebo features. We can see that the best results are obtained when using settings $P1$ with no placebo features. Also, for both $P1$ and $P2$, adding placebo features decreases the scores, whereas for $P0$, there is an increase in score when placebo features are added.
Figure 4.10: This plot shows the average deviation in AUC ROC scores from $W0 : P1 : R0$ for different parameter settings when using different weather features. The plot indicates that weather does not positively affect the scores of the classifier when the hyperparameters have been adjusted.
Chapter 5
Discussion and summary

In this chapter we begin with a brief discussion of the results and some of the potential explanations for what was observed. Following that is a section about some of the ideas for potential future work. The chapter is concluded with a short summary of the project.

5.1 Discussion of the results

The final results of this project were admittedly disappointing, particularly in light of some early promising tests. There are of course many potential explanations for why no improvement due to weather was found. Mistakes could have been made along any number of steps during the implementation, and perhaps the results would have been better with a different collection of weather measurements or more cleverly constructed features. A different combination of baseline and weather features might have been effective on some particular patient cluster that was not tested. A different set of target labels might have also made a difference, and it is also possible that a different classifier altogether would have been more suitable. Another possibility could be if the best possible improvement is in fact so minute that it can’t be measured, or that it can only be detected with more data, or data from locations other than Cleveland, Ohio. The data itself might also contain errors or outliers that renders it essentially useless, although this seems fairly unlikely.

A perhaps more plausible explanation is the hypothesis that the physiological measurements used in the baseline already capture everything that might be covered by weather. It might for example be the case that some patients are more likely to suffer a heart attack during cold weather, and it might also be that the reason for such an increased risk is entirely explained by the fact that the cold weather causes the skin to tighten which increases blood pressure. But if we already measure blood pressure, then learning about the air temperature provides us with no additional useful information.

Another potential reason for the negative result can be gleaned from Braga et al., who showed that the correlation between temperature and cardiovascular mortality had a rather
restrictive time window [25]. Indeed, the weather effect could be arbitrarily strong, but if it only affects the patients on the same day that the weather occurs, its use in a system like LYDICS VÖR® is severely limited, because the target predictions relate to events that might occur up to a month later. It might simply be the case that the temporal reach of past and present weather is not long enough to be useful. This point would be moot if one could accurately predict the necessary weather conditions far enough into the future, but of course weather is notoriously difficult to predict more than about a week into the future. It might still be interesting to utilize forecasts as features, though.

5.2 Future work

There are a number of different ideas that were considered during this project, but that were ultimately not implemented and tested, primarily due to time constraints. Some of these ideas are more or less simple variations of things that were in fact tested, and others would have been more ambitious expansions of the project scope, but would nevertheless have been rather interesting to pursue. Generally speaking, the shelved ideas include variations of the input weather data, the types of features that are extracted and how one can filter out the most important ones, other ways of evaluating the performance of the model, different ways of clustering patients, different target labels for the classifier, including regression models as well as different types of classifiers altogether. Following is a quick summary of ideas that would have been interesting to investigate further. It seems pretty clear that the final word on weather has not been spoken.

5.2.1 Other weather sources

Perhaps the most obvious next thing to try would be to incorporate other kinds of weather and climatological measurements into the model. Measurements of various air pollutants would be particularly interesting, because several studies have linked pollution with the frequency of hospitalizations due to COPD, for instance [27]. Other statistics such as cloud cover, fog, thunderstorms and so on might also be relevant, although the intuitive leap is perhaps a little longer to make such connections.

In addition to adding new weather measurements, a more spatially diverse patient group might have also been beneficial in that it would have allowed a cross-reference between patients living in different states, for example.

5.2.2 Automatic feature extraction

There is a tremendous amount of possibilities when it comes to engineering features from the raw data, and in this project we have barely even scratched the surface of this. But already with the small number of basic features that were tested during this project, for some relatively restrictive settings for the rolling averages and Fourier coefficients, we ended up with over 200 new features. Manually designing tests to explore the impact of various combinations only for these two methods for generating time-series features is quite time-consuming, and ideally one would use a systematic and automated approach
for this. There were initially some ambitions to do precisely this, but in the end we were forced to move these ideas to the category of possible future improvements.

Feature extraction is an incredibly difficult problem to approach systematically, and it is an active area of research. Generally speaking, the problem is this: there is an infinite number of possible transformations of the input data, doubly so if one considers time-series, and even if we were to construct a large number of features, most classifiers scale unfavorably (i.e. at least linearly) in complexity with the number of features. Therefore, it is necessary to filter the features to select only a small subset of the possible features, but this is also a very difficult problem, in particular when one considers multi-variate interactions between features.

Certain software packages for automating the process of both generating new features and subsequently filter them to select the best ones have in fact been developed and made publicly available. The tsfresh package in particular is capable of both generating a multitude of new time-series features, and filter them to select only the most important ones, but the filtering algorithm does not take multi-variate feature interactions into account [52]. The rebate package uses so-called relief-based algorithms to filter a large number of features effectively and has been shown to perform rather well on a large class of problems. Unfortunately the relief-methods all have a time and space complexity that is quadratic in the number of samples (although it is linear in the number of features), which effectively means that in order to use it for this project, only a small sample of the complete data set can realistically be used [53, 54]. In light of the results in section 4.2.2 where it was shown that even small values of the look-back parameter perform well, it might still be possible to use a relief-based method to filter from a large number of features generated by, for example, the tsfresh package.

5.2.3 Patient clusterings

Although we tried to group patients based on a rough classification of their past symptoms, the way in which these clusterings were done was not perfect. There are various ways in which this type of clustering might be improved, as well as additional types of clusters that could also be informative. Some of the ideas for clustering that were not tested include: age, sex, hospitalization frequency and difficulty (although it is not necessarily trivial to specify which patients are in fact unpredictable according to the classifier).

It might also be possible to approach the patient clustering from another perspective, and group them based on some of their feature values instead. If a suitable similarity measure were to be defined, in particular one that treats each patient as a multivariate time-series, then one of several clustering algorithms could be used to automatically identify the most similar patients. This approach is perhaps divorced from the idea of grouping patients by their actual symptoms, but that doesn’t necessarily mean it is any less effective. Regardless of method, patient clusters might also be of utility in an ensemble model where each patient cluster is trained by a separate model.

5.2.4 Different classifiers

It is not certain that the random forest classifier is the best choice for the problem domain at hand, particularly considering the temporal aspects of the data. Had the scope of the
5. Discussion and summary

project not been limited to using random forest, some variations of recurrent neural networks would have been high on the list of other classifiers to try. A number of different neural networks were in fact tested in a previous master’s thesis at Lytics, but the results were somewhat inconclusive, especially considering the difficulty in tuning such algorithms, but continuing the work in that direction would have been interesting [55]. Some promising approaches include Generative Adversarial Networks, Long Short-Term Memory (LSTM) networks, random survival forests and temporal difference learning. Another type of recurrent neural network, called Clockwork RNN, seemed particularly interesting [56].

5.2.5 Different target labels

It is not entirely clear that hospitalization within 30 days is the optimal choice of target labels for our kind of decision support system. Reformulating the problem might affect (positively or negatively) the usefulness of the system, when one considers that the ultimate goal is not necessarily to predict whether a patient becomes hospitalized, but rather to provide useful indications to a nurse as to which patients are most helped by their efforts. Of course changing the target labels of the classifier, or even turning away from the classification problem entirely, falls squarely outside the scope of this project, but some ideas along this line were nevertheless entertained in passing. Furthermore, if the target labels change, it is conceivable that weather might play a more significant role in predicting them.

Some ideas for altering the labels include changing the time-frame, dividing the time until hospitalization into multiple groups (and thereby changing to a multi-class classification problem), weighting the labels by the length of stay, filtering the hospitalizations such that the algorithm focuses more on preventable hospitalizations, trying to predict the reason for hospitalization rather than the hospitalization itself, and so on. Of course it might turn out that none of these ideas are practical upon further investigation, but it might still be worthwhile to think outside the box, as it were, and not limit oneself to a particular problem formulation merely because it was the first to be implemented.

5.3 Summary and conclusion

In this master’s thesis, weather measurements were used in an attempt to improve an existing random forest classifier designed to predict the hospitalizations of ESRD patients. The weather was collected from NCEI and consisted of daily summaries of temperature, wind speed, humidity, atmospheric pressure and precipitation. These measurements were used to extract weather features $X_w$ in three distinct ways: as combinations of measurements, averages of measurements for various time spans, and coefficients from the discrete Fourier transform. The effects of the classifier in terms of AUC ROC score were measured for different combinations of the weather features as well as different combinations of original baseline features from the existing system. A number of different cross-validation strategies were also utilized in the testing effort.

At first a small but statistically significant improvement was observed as a result of adding weather features to the model, but upon closer examination, it was found that a
similar improvement was also present when random noise was introduced instead of the weather features. By adjusting certain hyper-parameters of the random forest classifier it was possible to increase the performance of the model while also removing the effect of random noise. However, in so doing, the effect of the weather features also disappeared entirely.

Ultimately, no improvement to the model was observed that could not be better explained by things other than the new weather features. Nevertheless, we believe that the project was still of value to Lytics: the system was in fact improved, even though the improvement was due to adjusted parameters and not weather. Furthermore, during the thesis a new test environment was developed that makes it easier to experiment with new ideas. This environment has the potential to save a substantial amount of time for similar, future endeavors.


Appendices
Appendix A
Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>AUC</td>
<td>Area Under Curve</td>
</tr>
<tr>
<td>AUC ROC</td>
<td>Area Under Curve (for the) Receiver Operating Characteristic</td>
</tr>
<tr>
<td>CART</td>
<td>Classification And Regression Trees</td>
</tr>
<tr>
<td>CDC</td>
<td>Centers for Dialysis Care, Inc.</td>
</tr>
<tr>
<td>CKD</td>
<td>Chronic Kidney Disease</td>
</tr>
<tr>
<td>CMS</td>
<td>Centers for Medicare and Medicaid Services</td>
</tr>
<tr>
<td>COPD</td>
<td>Chronic Obstructive Pulmonary Disease</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>ESCO</td>
<td>ESRD Seamless Care Organization</td>
</tr>
<tr>
<td>ESRD</td>
<td>End Stage Renal Disease</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short Term Memory Neural Network</td>
</tr>
<tr>
<td>Lytics</td>
<td>Lytics Health AB</td>
</tr>
<tr>
<td>NCEI</td>
<td>National Centers for Environmental Information</td>
</tr>
<tr>
<td>QCLCD</td>
<td>Quality Controlled Local Climatological Data</td>
</tr>
<tr>
<td>RN</td>
<td>Registered Nurse</td>
</tr>
<tr>
<td>RNN</td>
<td>Recursive Neural Network</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristic</td>
</tr>
<tr>
<td>SOB</td>
<td>Shortness Of Breath</td>
</tr>
<tr>
<td>The System</td>
<td>LYTICS VÖR®</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
</tbody>
</table>
Väder som verktyg för prediktion av hospitalisering

Att väder påverka kroniskt sjuka patienter är väl dokumenterat. Det här arbetet har undersökt om sådana observationer kan användas i en maskininlärningskontext för att förbättra den preventiva sjukvården för patienter med kronisk njursvikt.


För att maskininlärning ska fungera bra så behövs stora mängder data. Dialyspatienter är relativt tacksamma i den bemärkelsen eftersom de får dialysbehandling flera gånger i veckan, och under denna behandling kan man samla in blodvärden och annan potentiellt relevant information. I mitt examensarbete har jag undersökt huruvida systemet som Lytics har utvecklat kan förbättras genom att lägga till väderdata i modellen.

För att undersöka detta på ett systematiskt sätt har en testmiljö utvecklats i syfte att undersöka om det är möjligt att förbättra arbetet med att lägga till och ta bort olika typer av meteorologiska mätningar och sedan jämföra hur detta påverkar algoritmens förmåga att förutsäga när patienter blir sjuka. Även effekten av att ta bort vissa fysiologiska datapunkter har undersömts, för att se om dessa kan ersättas av väderdata. Jag har även utforskat huruvida det fanns särskilda patientgrupper som blev lättare eller svårare att prognostisera när väderdata lades till.

Efter omfattande tester har emellertid ingen signifikant förbättring som följd av vädermätningar kunnat påvisas. Detta betyder dock inte att väder inte inverkar, men det är en indikation på att i den utsträckning vadret har mätbara effekter i det här sammanhanget så är de antingen svaga, eller så fängas de upp av de fysiologiska mätningar som redan ingår i modellen. Det kan också vara så att den effekten av väder inte är tillräckligt långvarig för att kunna vara till någon hjälp. Detta skulle exempelvis vara fallet om temperaturen idag har en stor inverkan för en patients välmående imorgon, men ingen talar om väderdata i området om en vecka eller en månad.