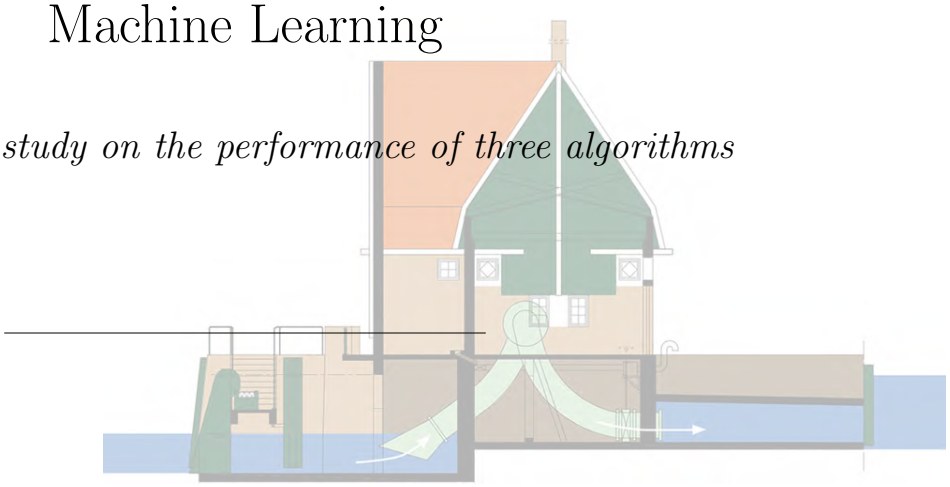
Please note that this is the anonimized version of the full report. Therefore several adjustments have been made. For instance instead of using the pumping stations name, it has been called e.g. 'PSa' (pumping station a). Also figures have been altered to display those new names. Finally all the Attachments and Code has been removed, this caused in the report that if anywhere it was linked to these removed Attachments to show up like '??' .

Predictive Maintenance for Sewer Systems using Machine Learning

A comparative study on the performance of three algorithms



A. Shanti Bruyn
st. nr. 2171627

Vrije Universiteit
August 20, 2018

Witteveen+Bos:
Main mentor:
M. Stam

Vrije Universiteit:
W.J. Fokkink

Second reader:
R. Bekker

Contents

1 Introduction **1**
- 1.1 Predictive maintenance . . . 1
- 1.2 Machine learning . . . 2
- 1.3 The sewer system . . . 2
- 1.4 Problem statement . . . 4
  - 1.4.1 Main goal . . . 5
  - 1.4.2 Secondary goals . . . 5
- 1.5 Overview . . . 5

2 Materials & Methodologies **6**
- 2.1 Data . . . 6
  - 2.1.1 Original database . . . 6
  - 2.1.2 Open source data . . . 7
- 2.2 ML techniques . . . 7
  - 2.2.1 Random Forest (RF) . . . 9
  - 2.2.2 X Gradient-Boosting (XGB) . . . 10
  - 2.2.3 Support Vector Machine (SVM) . . . 11
  - 2.2.4 Long-Short Term Memory (LSTM) . . . 13
    - 2.2.4.1 Challenges regarding the temporal model . . . 14
    - 2.2.4.2 The proposed model . . . 15
  - 2.2.5 Comparing techniques . . . 17
- 2.3 Software used . . . 21
- 2.4 Methodologies . . . 21
  - 2.4.1 Data preparation . . . 21
    - 2.4.1.1 Feature engineering . . . 21
    - 2.4.1.2 Non-temporal data . . . 21
    - 2.4.1.3 Temporal data . . . 22
    - 2.4.1.4 Empty variables . . . 23
    - 2.4.1.5 Outliers . . . 25
  - 2.4.2 Selecting the defects . . . 27
  - 2.4.3 Selecting the pumping stations . . . 29
  - 2.4.4 Selecting the final variables . . . 30

3 Results **31**
- 3.1 Data . . . 31
  - 3.1.1 Final dataset vs original dataset . . . 31
  - 3.1.2 Feature importance . . . 31
- 3.2 ML models . . . 35
  - 3.2.1 Models . . . 35
    - 3.2.1.1 RF . . . 35
    - 3.2.1.2 XGB . . . 40

3.2.1.3 SVM . . . 45
3.2.1.4 LSTM . . . 49
3.2.1.5 Before oversampling minority . . . 49
3.3 Prediction . . . 51
3.3.1 Insights . . . 55

4 Discussion **58**
4.1 Usefulness of the results . . . 58
4.2 Validity of results . . . 58
4.3 How transferable are the results? . . . 60

5 Conclusion & Recommendations **61**
5.1 Conclusion . . . 61
5.2 Recommendations . . . 61

Bibliography **63**

Abstract
In this research we tried to find if it was possible to use Machine Learning techniques in order to predict when a pumping station would experience a serious defect. In order to do this we looked into three algorithms (Random Forest, X-Gradient Boosting and Support Vector Machines) and compared their performance on 13 pumping stations. Of those three the X-Gradient Boosting had the highest F1-score most of the time. And in all cases -despite some data issues- the certainty goes up. The worst performing F1-score is on a pumping station where 7.95% of the instances experiences a defect, and the reached precision and recall are respectively 0.450 and 0.333. Therefore the conclusion of this research is that Machine Learning can definitely help with predicting defects in the sewer system. To find the algorithm and variables to do this best would be a new research.

Word of thanks

I would like to thank a few people, in the order in which they for the first time came into contact with me and my research. To start with Jaap, you had all faith in me from the start and mentored me to setup a good basis and method to look at all the data in the database. Secondly Wan, you mentored me throughout my complete research and were there with advice on how to tackle difficult situations and always ready to think with me whenever I encountered an issue. Thirdly Petra, you helped me understand the sewer system and interpret what the values in the database actually ment in real life and helped me make a good start with writing everything down. Then Mark, you were there for any question I had with regards to Machine Learning, you always tried to make time to fit me in your extremely busy time schedule. And finally Mattijs, you helped me to follow my own ideas, and gave me helpful feedback on my report.

Managers summary (en)

The main goal of this research paper is to find out whether Machine Learning can be used to predict a defect in a sewer system in the upcoming three days with the measurements already in place. This research found that it is -within limits- possible to do this. Of the algorithms this research tried, the X-Gradient Boosting seems the most promising. However, when some extra measurements would be added to the database (like energy-usage of the pumping station and turbidity), the predictions would probably be even better.

Managers summary (nl)

Het belangrijkste doel van dit onderzoek is om erachter te komen of Machine Learning gebruikt kan worden om een ernstige storing in het riool systeem in de komende drie dagen te kunnen voorspellen. Dit onderzoek wijst erop dat dit -hetzij met mate- mogelijk is. Het algorithme X-Gradient Boosting presteert het beste van de drie geteste modellen. Verder zouden de voorspellingen waarschijnlijk enorm verbeterd kunnen worden door een paar extra metingen op te nemen in het systeem (zoals het energieverbruik van het gemaal en de troebelheid van het water).

Chapter 1

Introduction

1.1 Predictive maintenance

Predictive maintenance is quite a broad term used in many different contexts. But it has always something to do with knowing ahead of time when a part will require maintenance, so 'unplanned downtime' can be prevented.

Sometimes predictive maintenance is called the 'fourth generation engineering'. The three predating generations of engineering are roughly (in order of occurrence): 'fix it when it breaks' [1], 'replace it regularly so it will not break' and 'replace it when the expected life span is nearly at its end / reached a certain state'. In this line predictive maintenance can be said to be: 'replace it when the data indicates it will break in the near future'.

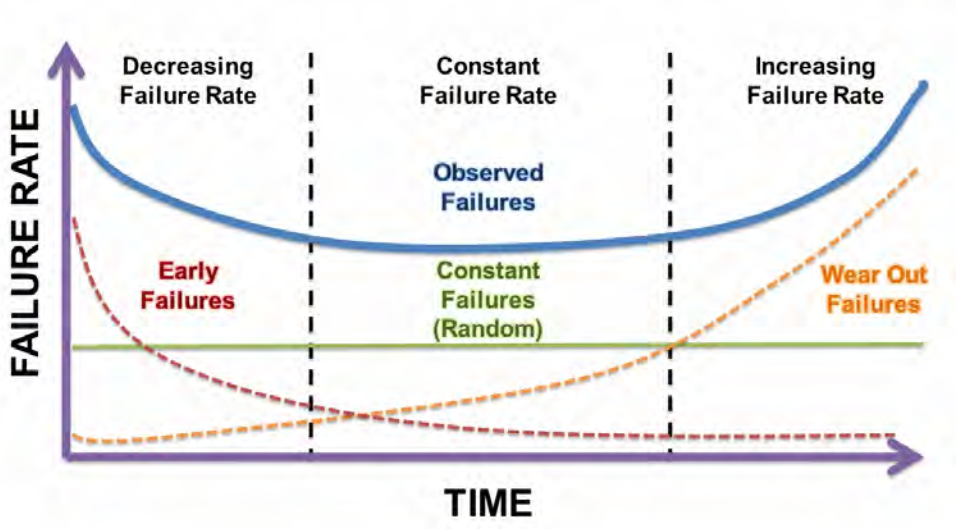


Figure 1.1: Type of failures of parts shown throughout time [2]

Predictive maintenance has the potential to decrease costs. Looking at Figure 1.1 -depending on how critical the system is- somewhere in the section 'Constant Failure Rate' the part will be replaced. Which will be for a relatively small percentage of the parts the optimal moment: just before they break down. However, if predictive maintenance is applied, ideally nearly all parts will be replaced at this optimal point, because it is predicted for each part separately when it will break down. This means that the downtime costs due to the early failures can be drastically reduced. And that the parts do not have to be replaced until their optimal replace moment arrives. This avoids additional costs due to replacing a still well-functioning part.

Currently 'predictive maintenance' is rising in popularity. Many sectors with assets are looking into it, hoping to reduce the (often huge) costs of asset management. The replacement costs alone of the sewer system in the Netherlands during the time 2010-2012 were 670 million euro [3].

1.2 Machine learning

The term 'Machine Learning' (ML) dates back to 1959 [4]. ML comes down to letting the computer 'learn' from historical data. Before ML existed, a 'rule set' had to be explicitly programmed in order for a computer to be able to draw conclusions based on data.

In general it can be stated with ML that more and better data leads to better results (informally sometimes it is said: 'garbage in garbage out').

There are two major classes within ML: the first is 'supervised' and the second is 'unsupervised'. Supervised ML means that the data is already 'labeled'. So the output is already known for a large part of the data. This is not the case for unsupervised ML, which is often used to find new patterns in the data. In this research it is already known whether a defect has taken place or not, therefore the data is 'labeled' and thus supervised learning will be applied.

To validate a supervised ML model, the dataset is divided into a training set and a test set. While the model is training, it only gets to see the training data. Once the model is done training, it will try to predict the outcome with the input variables from the test set. These outputs can then be compared to the real output to draw a conclusion on the viability of the model.

As stated in section 1.1, in order to do predictive maintenance, data are required to get an indication on when a part will break down. In order to learn what aspects of the data form indications for a nearing breakdown, ML is often used. Causing ML to be frequently used in regards to predictive maintenance.

1.3 The sewer system

First and foremost sewer systems came into existence to protect the human population from diseases. The oldest sewer pipe ever found dates back to 4000 B.C. [5]. However, the start of the realization of the true importance of a well-functioning sewer system had to wait till the cholera outbreak in London of 1858, when John Snow realized and showed that the outbreak was caused by a single feacal contaminated water well [6]. After this realization the sewer systems around the world have improved greatly. In The Netherlands there was already some primitive form of waste management. But this big transformation to make sure nearly everyone is attached to the waste management system only occurred around 1960 [7]. The modern day importance of the sewer system is indicated by the fact that the British Medical Journal voted in 2007 'the sanitary revolution' as the most important medical milestone since 1840 [8].

In The Netherlands the dominant type of sewer system is the combined system. This entails that domestic and industrial wastewater and storm water runoff from roofs and streets are transported together in one sewer pipe to a wastewater treatment plant (WWTP). Due to limited storage and pumping capacity the system can get overloaded during heavy rainfall. In order to make sure that no wastewater flows onto the streets or comes back up through toilets in people's homes, combined sewer overflows (CSOs) were introduced. At CSOs wastewater can

flow directly into the surface water if the water levels in the sewer system get too high. Alternative types of sewer systems exist, such as the (improved) separate sewer system, where polluted (domestic) wastewater and more clean storm water runoff is transported through separate pipes.

The sewer system is generally divided in multiple smaller 'units' which are called 'catchments'. Each catchment collects the water and due to gravity the water flows to the lowest point of that catchment. On this lowest point a pumping station is installed to pump the water into the next catchment. The final catchment discharges directly into the WWTP.

A pumping station can consist of multiple pumps. For instance, with the combined system there are usually at least two pumps to be energy sufficient. One (smaller) pump for the polluted (domestic) wastewater, also called dry weather flow (dwf). And one (bigger) pump for the storm water runoff, also called wet weather flow (wwf). Sometimes a spare of these pumps is added, which can take over immediately in case a pump breaks down.

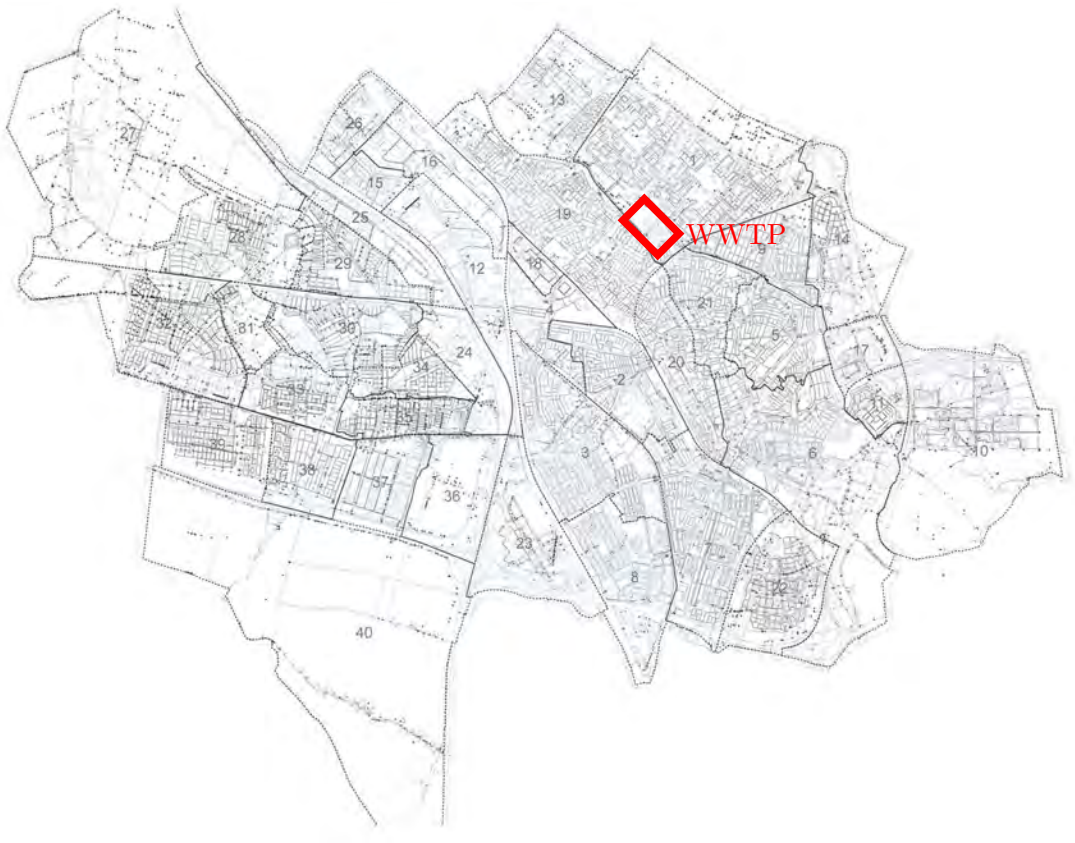


Figure 1.2: The municipality of Utrecht divided into catchments, with the WWTP location in red

This study focuses on the sewer system in the municipality of Utrecht. With almost 350,000 inhabitants [9] and a connected area of approximately 1.5 km^2[1], the municipality of Utrecht is a big sewer system in the Netherlands. This system is divided into 40 catchments as can be seen in Figure 1.2. Similar to other areas in The Netherlands, the dominant type of sewer system is the combined system.

[1]Estimate based on [10] page 27

1.4 Problem statement

Due to the -possibly- massive cost reduction, many municipalities try to find ways to change preventive (or sometimes even reactive) maintenance into predictive maintenance. This research will try to find an answer to the question whether ML can help to move -with regard to sewer systems- to predictive maintenance.

In order to predict a defect in the sewer system of Utrecht, a lot of data is needed: the more the better. This data will be extracted from a database which was built in a pilot to measure as much as possible in a sewer system. Six municipalities together started the group 'Meten' ('Measuring' in Dutch), which had the goal to create a database in which all the municipalities collaborated. The municipality of Utrecht pragmatically started building a monitoring network with corresponding database in 2006 [11]. Since then the monitoring network has been elaborated, and settings have been altered[2]. For this reason only data since April 2013 will be taken into consideration.

In 2016 Arjen Kruit looked into this very same database, hoping to be able to predict the defects [12]. He indicated 8 major defects which cause a pumping station to completely stop working. He used an XG_boosting ML algorithm to predict the defects. Sadly his results were that no pattern had been found to reliably predict such a defect. Looking at this result we wanted to find the results when a few choices are made differently.

Shortly before this research was finished, another research got published which turned out to be very helpful. Marly van der Meij published her 'Predictief Onderhoud' [13]. This research was done on the same database. She looked into more detail at one specific pumping station. Her scope was more the mechanical engineering one. However she did get extremely good results on the pumping station she chose with the Random Forest algorithm. An effort will be made to use her two most telling features too. It is important to note that the researches have been set up very differently. She focussed on one pumping station in particular whereas this research looks at several pumping stations. Secondly her data is aggregated per 10 minutes, and this research aggregates the non-temporal data per day. Lastly regarding the defects, different choices have been made.

It is unclear which model suits the data from the sewer system best in order to be able to -most reliably- predict major defects in a pumping station. For this reason multiple models will be tried. To start with an algorithm which looks at the data as if it is temporal data (looking at the data as if it is a time series): the Long-Short Term Memory (LSTM)[3] (see subsection 2.2.4). In order to be able to compare results with Arjen Kruit [12] the X-Gradient Boasting (XGB) algorithm will be tried (see subsection 2.2.2). Similarly to be able to compare with Marly van der Meij [13] the Random Forest algorithm (RF) will be tried (see subsection 2.2.1). Last but not least a completely different classifier will be tried: the Support Vector Machine (SVM) (see subsection 2.2.3).

Once major defects in a pumping station can be predicted reliably three days in advance, unplanned down-time can be prevented. And hopefully mechanics will no longer have to be called in the middle of the night or during weekends to fix those defects. Another major advantage would be that with knowledge on when a part will fail costs for Asset Management can be reduced.

[2] For instance the predetermined starting level changes throughout time, an example can be seen in Figure 2.12.

[3] The researchers struggled so much with this model that in the end it was not run, see more on the struggles in subsubsection 2.2.4.1

To make the goals of this research paper explicit:

1.4.1 Main goal

Is it possible with "Machine learning" to predict a defect in a sewer system in the upcoming three days given the measurements in the system.

1.4.2 Secondary goals

1) *Which Machine Learning techniques are best suited for this goal?*

2) *Using experts advice and data analyses, what are the most important characteristics to predict a possible failure of a sewer system?*

1.5 Overview

The remainder of the report will start of with what Materials & Methodologies have been used. Giving some insights in the starting point and the thought process on data there is to work with (section 2.1), what ML techniques will be used (section 2.2), what software was used during this research paper (section 2.3) and finally the methods used to get from the raw data to the data used by the models (section 2.4). Then the results will be presented. Again, starting with the importance of the data (section 3.1), followed by the importance of the set parameters in the algorithms (section 3.2) to finish with the achieved predictions (section 3.3). Finally in the discussion (chapter 4) the results will be discussed. The conclusion (section 5.1) will answer the research questions and the recommendations (section 5.2) will give some recommendations and some possible future research. Completely at the end some personal struggles throughout the research paper will be talked about in the Peroration.

Chapter 2

Materials & Methodologies

This chapter will cover all the materials at disposal, like the received data (section 2.1) and the software used (section 2.3). Followed by the methods used on/with those materials to get to the results, including ML techniques (section 2.2) and data handling (section 2.4).

2.1 Data

2.1.1 Original database

As mentioned in the introduction data from 1 April 2013[1] up till and including 31 December 2017 will be used. This database collects data on all 40 catchments (see Figure 1.2), with a total of 525 pumping stations. There are nearly 380 million measurements in total in the database.

In this database the following variables are present:

- Flow rate[2]: the number of m^3 that passes through the sewer system per 5 minutes
- Groundwaterlevel[3]: in m above NAP[4]
- Precipitation[5]: in mm/5min
- Turbidity[6]: in FNU (= Formazine Nephelometric Unit)
  A measure for how see-through the water is.

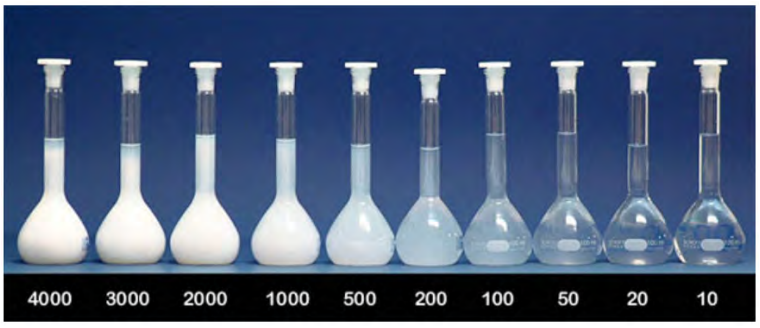


Figure 2.1: What does Turbidity look like? [14]

[1] In the orange box in Figure 2.12 it can be seen that the predetermined starting levels have been changed
[2] In Dutch: debiet
[3] In Dutch: grondwaterstand
[4] 'Normaal Amsterdams Peil' or freely translated 'normal waterlevel in Amsterdam'
[5] In Dutch: neerslag
[6] In Dutch: troebelheid

- Level measurement[7]: measured as difference in m from NAP in the sewer system itself
- Groundwaterlevel uncorrected: 'cm WK' meaning freely translated 'centimeter watercolumn', which is a measurement of pressure
- Groundwaterlevel manually: measured as difference in m from NAP in the surface water
- Airpressure[8]: 'cm WK'
- Temperature: in degree celcius
- Goundwater17: a variable with no unit or parameter. Concerning groundwater
- Waterlevel[9]: in 'gr.C'[10]
- ciw20: a variable with no unit or parameter or system

Sadly after some analysis in the database, most of those variables turned out to be unusable for several reasons. To start with, some were barely ever filled out. This was the case with for instance turbidity: sensors should have been up on five different locations. But there were only measurements for about two months on one location. This was quite a disappointment, for nearly all experts said this should have been a great indicating factor. Another issue was that many variables were not directly linkable to the location of a pump in the sewer system. And given the size of the system, this could have given a quite wrong impression.

In the end all the following research has to focus on the two variables Flow rate and Level measurement.

Also data on defects is collected in the database. In total there are nearly 3,000 different types of defects, of which 75 actually took place in the chosen time frame.

2.1.2 Open source data

To enrich the database, both daily and hourly weather data have been collected from the KNMI from 01-01-2012 up till and including 2017-12-31. The non-temporal methods will use the daily data (**??**). Since for the non-temporal methods the data has already been aggregated per day. LSTM will use the hourly data (**??**) despite the data for the LSTM being per minute. The KNMI simply does not provide such detailed data.

Other open source data have been considered. Such as whether the pumping station is situated in industry or living area. However due to time constraints this was not looked further into.

2.2 ML techniques

In search of a well-fitting model, four models will be tried and evaluated: Random Forest (RF), XG-Boosting (XGB), Support Vector Machines (SVM) and Long-Short Term Memory (LSTM). For all four models a grid search will be performed to make sure that the chosen parameters are good.

[7] In Dutch: peilmeting
[8] In Dutch: luchtdruk
[9] In Dutch: waterstand
[10] Even the experts have no idea what 'gr. C' could stand for with regards to the waterlevel. It is therefore assumed that this is a mistake in the documentation.

The four models exist of two different 'types' of algorithms, one works with so-called 'temporal' data and three work with 'non-temporal' data. Temporal data is data about a situation at a specific moment in time. Whereas non-temporal data can be aggregated.

Skewed data
The data is quite skewed, as can be seen in Table 2.5 and Table 2.6. So some attention should go out to that aspect as well.

Originally the idea was to let all the models, data sources and parameter settings run and determine via 5-fold cross-validation the average accuracy and the model with the highest score would be the best. However, since the data is so skewed, it was decided to oversample the minority. Doing this combined with cross-validation would result in a model over-sensitive to exactly those settings with a defect because those instances would be copied over and over. This situation is sketched in Figure 2.2.

Instead it has been decided to create a test set first with about $^1/_4$ of the total defect instances and total non-defect instances. And to only oversample the minority already in the training set. This situation is sketched in Figure 2.3. Sadly this resulted in not using 5-fold cross validation, as that would always result in situations similar to Figure 2.2.

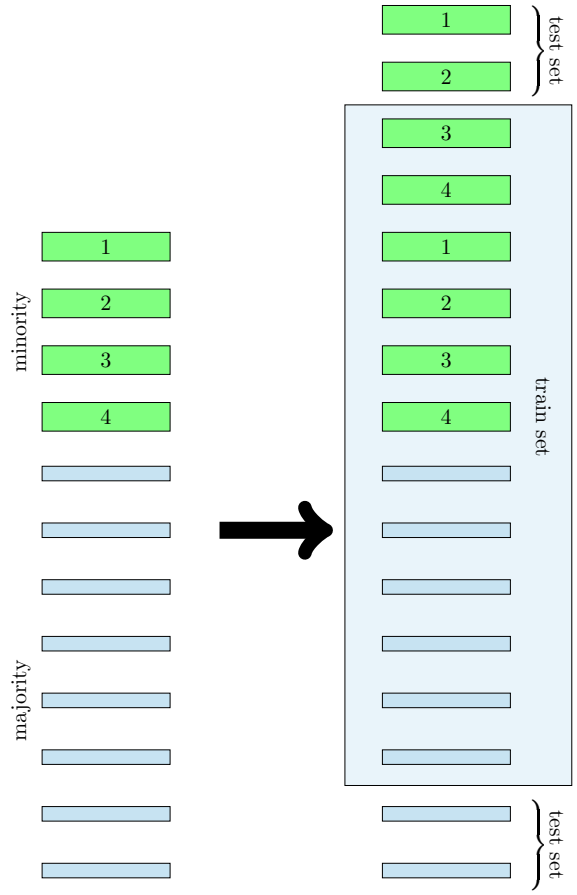


Figure 2.2: The test set has exactly the same instances of defects, as where the model trains on in the training set (1 & 2). Resulting in a less reliable validation score.

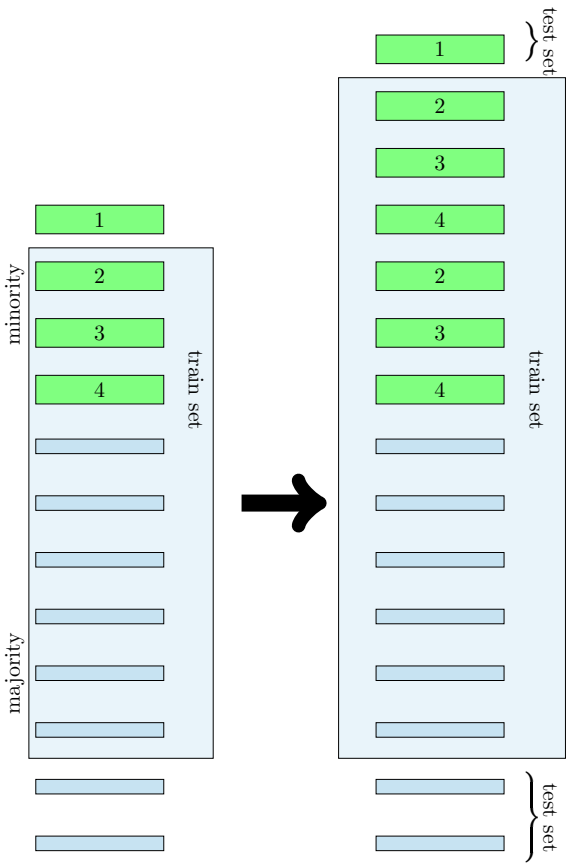


Figure 2.3: Now the test set contains no defects with exactly the same instance as the training set. Resulting in a more reliable validation score.

2.2.1 Random Forest (RF)

A Random Forest is a Bagging of the predictors of the Decision Tree. Breiman explains it as: 'Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor. The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote when predicting a class.' [15]. Resulting in 'What one loses, with the trees, is a simple and interpretable structure. What one gains is increased accuracy.' [15].

In other words a Random Forest is a 'forest' of Decision trees. Basically one creates a lot of these Decision trees with a slightly randomized aspect to ensure they are not all exactly the same. Then when a prediction needs to be made, all Decision trees are asked to 'vote' for their prediction. The prediction with the most votes will be the prediction given by the Random Forest.

A Decision tree is a model which is built up from questions. So for instance if one were to try and label a vehicle as either a car or a bicycle, an example might be:

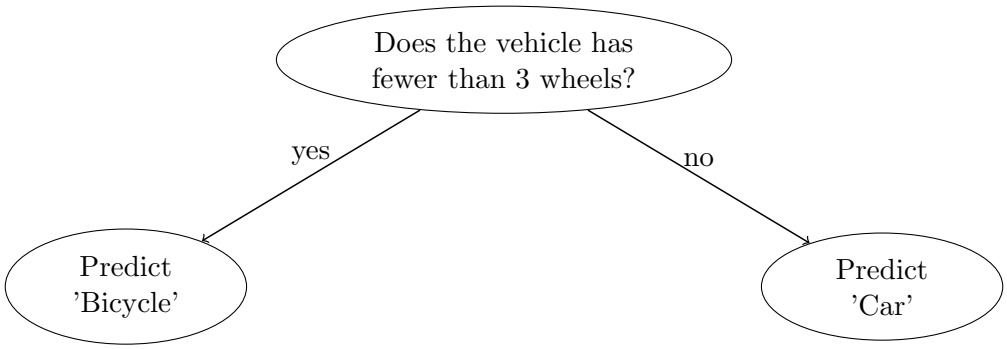


Figure 2.4: Example of a decision tree

Naturally in a model most of the time the Decision trees are more elaborate than the given example and have many more 'levels'. But the idea remains the same.

Interesting to note here is that Marly van der Meij [13] also uses a Random Forest in her research.

There are 17 parameters which can be set in a Random Forest. Those are all given in **??** with their explanation, defaults and range. In Table 2.1 all the values which will be tried in the parameter grid search in order to find the 'optimal values' are shown.

name	grid search
n_estimators	{ 5,50,500 }
criterion	{ 'gini', 'entropy' }
max_features	{ 'auto'; 0.25; 0.50; 0.75 }
max_depth	{ 'None' }
min_samples_split	{ 2; 0.1 }
min_samples_leaf	{ 1; 0.1 }
min_weight_fraction_leaf	{ 0 }
max_leaf_nodes	{ None }
min_impurity_decrease	{ 0; 0.25; 0.5 }
bootstrap	{ True; False }
oob_score	{ True; False }
n_jobs	{1, -1}
random_state	{ 0 }
verbose	{False}
warm_start	{ True; False}
class_weight	{'balanced'}

Table 2.1: All the variables and the suggested parameter grid search

These parameters have been chosen for several reasons, all will be discussed shortly: Because the number of Decision trees can influence the outcome a lot, it has been chosen to try a rather broad range. Varying the `n_estimators` from 5 trees to 500. Secondly the `criterion` will try all possible options. For the maximum number of features (`max_features`) to consider, the best split, again a broad range has been decided upon, including the default. No limitation has been set to the `max_depth` of the tree. The `min_samples_split` is the absolute minimum - and default - (2) has been chosen to be tried next to a 10% demand. In order to be able to look at what effect this has. The same thought process is applied to `min_samples_leaf` also with the minimum (1) and the 10% option. With regard to `min_weight_fraction_leaf` it has been chosen to set no demand on the weights of all the input samples required to be a leaf. Similarly no limit has been set to `max_leaf_nodes`. On the other hand it has been chosen to try a spread of different values for `min_impurity_decrease` to see if it matters: 0, 0.25 and 0.5. `Bootstraps` simply tried all possibilities: True and False. Just like `oob_score`. The `n_jobs` tries both what happens if the code is run on 1 core or on all cores. `Random_state` simply has an integer to place the seed. `Verbose` is False because during the parameter grid search no output is needed, but this has no effect on the predictions. Nearing the end: `warm_start` simply tries all options. And finally `class_weight` is set to 'balanced' because it is an unbalanced dataset.

2.2.2 X Gradient-Boosting (XGB)

Where a Random Forest is the bagging of several Decision Trees, XG-Boosting is the boosting of a Decision Tree. 'The term boosting refers to a family of algorithms that are able to convert weak learners to strong learners' [16]. '[XG-Boosting is] a scalable end-to-end tree boosting system (...), which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges.' [17].

Noteworthy is that Arjen Kruit [12] also used XG-Boosting in his research.

In order to be able to set the correct parameters, a grid search will be done. The possible variables are shown in **??**. In Table 2.2 an overview is given of the parameter settings which are taken into consideration in the grid search.

name	grid search
loss	{'deviance', 'exponential'}
learning_rate	{ 0.01; 0.1; 1 }
n_estimators	{ 10; 100 }
max_depth	{ 3; 10 }
criterion	{ 'friedman_mse'; 'mae' }
min_samples_split	{ 2; 0.01 }
min_samples_leaf	{ 1; 0.01 }
min_weight_fraction_leaf	{0}
subsample	{ 0.50; 0.75; 1.0 }
max_features	{ None; 'sqrt' }
max_leaf_nodes	{None}
min_impurity_decrease	{ 0.0 ; 0.5 }
init	{None}
verbose	{0}
warm_start	{ True; False }
random_state	{None}
presort	{'auto'}

Table 2.2: All the variables and the suggested parameter grid search

From first to last the reasoning behind the options chosen for the parameter grid search: starting with `loss`: simply all the options are being tried. For `learning_rate` a range has been chosen which will be able to give an indication on how important this variable is. Next `n_estimators` since this is a trade-off with learning_rate not too many values have been chosen. Only the default and a lower value. With regard to `max_depth` the default has been chosen and a larger value since it depends on the interaction of the input variables, which is in this case quite big (a pumping station will only start pumping -e.g. produce a Flow rate- when the Level measurements is high enough). Looking at `criterion` the choice has been made to try both 'types', so a mean squared error and an mean absolute error. Since it states that the 'friedman_mse' is usually better, these are the two options to try. Coming next `min_samples_split` the default (and absolute minimum) of 2 has been chosen together with trying a 1% demand. Following `min_samples_leaf` similarly to min_samples_split the default will be used, and one other option: 1%. Next with `min_weight_fraction_leaf` only the default value will be used. Halfway, since the data is very skewed, multiple options will be tried for `subsample`: the default and two smaller values, leading to a reduction of variance and an increase in bias. Following `max_features` will be tried basically 'boundless' and the option which is frequently given. Similarly `max_leaf_nodes` will not be bound at all. With regards to `min_impurity_decrease` both the default and quite a high demand will be given for splitting a node. No `init` will be used. The `verbose` will be off, this has no effect on the model what so ever. The `warm_start` will try both options. And similarly to verbose the `random_state` has no effect on the model. Finally `presort` is mostly a variable which might reduce runtime, this is set to 'auto' so the computer can speed things up as much as possible.

2.2.3 Support Vector Machine (SVM)

In 1963 Vladimir N. Vapnik and Alexey Ya. Chervonenkis published a paper with the basis for SVM [18]. The idea is to divide the space of possibilities with a vector, which separates two classes. Originally this could only be a linear vector. However, in 1992 a solution was proposed using a 'kernel trick' to be able to have vectors which are not solely linear [19].

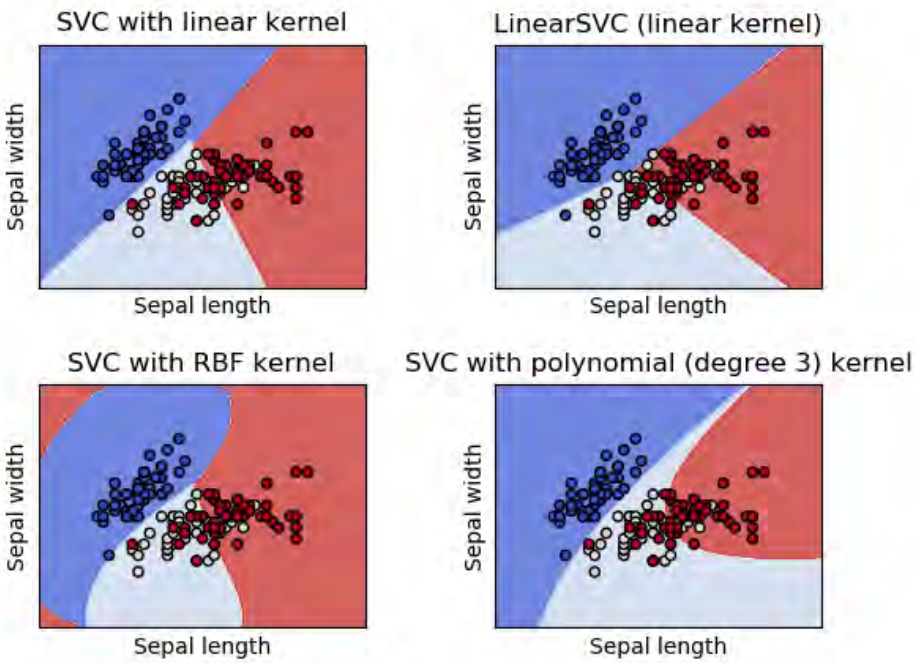


Figure 2.5: Visual examples of how SVM can be used as a classifier (SVC = Support Vector Classifier) [20]

SVM basically divides the realm of possibilities into 'area's' with for each possible prediction an area. Whenever an instance is in one of these area's, the prediction will be that belonging to that very area. In Figure 2.5 some visual examples are given.

SVM is used as one of the four algorithms to be tried and evaluated. This choice was made because it is a binary classifier. It is already implemented in a package in Python (scikit) [20]. And because of the 'penalty' variable it can be used in the case of skewed data [21]. On the other hand SVM is prone to over-fitting if the parameters are not chosen correctly [22] and SVM on itself does not give an indication of how 'certain' it is of its predictions. So to get probabilistic outputs, extra steps are necessary, which would increase the runtime.

In order to be able to set the correct parameters, a grid search will be done. The possible variables are shown in **??**. In Table 2.3 an overview is given of the parameter settings which are taken into consideration in the grid search.

name	grid search
C	{0.01; 1.0; 5.0; 10.0; 15.0}
cache_size	{200MB}
class_weight	{'balanced'}
coef0	{0; 1.0; 10.0}
decision_function_shape	{ 'ovo' }
degree	{ 1; 3; 5 }
gamma	{ 'auto'; 1.0 ; 0.1; 0.001 }
kernel	{ 'rbf', 'sigmoid', 'poly', 'linear' }
max_iter	{ -1 }
probability	{False}
random_state	{None}
shrinking	{ True }
tol	{ 0.0001; 0.001; 0.01 }
verbose	{ False }

Table 2.3: All the variables and the suggested parameter grid search

The rationale behind the chosen values for the parameter grid search is as follows, starting with `C`, the penalty parameter. Since the data is very skewed a high value is in order. Just to make sure a low value has been chosen as well together with the default and some values in between. Second comes `cache_size` which has been left at the default. Next the `class_weight` -again- since the data is very skewed this has been set to 'balanced'. Next the `coef0` only significant with kernel 'sigmoid', two values next to the default are being tried. For the `decision_function_shape` it has been decided to only try 'ovo' because there are only two classes, so one-vs-one should do the trick. Followed by `degree` which is only useful for the kernel setting 'poly', two values around the default and the default have been chosen to try. Now `gamma` is a kernel coefficient used by multiple kernel options, so this one might be important, for that reason four options are chosen to be tried. From the four possibilities regarding `kernel` all options will be tried. For `max_iter` nothing is set, the default will do. The `probability` is False, this does not influence the algorithm, only the time it takes to run. For `random_state` it has been decided to simply leave it at the default as this should not have too much effect. Leading on to `shrinking`, for which it was decided to go for the quickest solution with 'True'. The last important variable `tol` is the tolerance for the stopping criterion. This might make big differences, for that reason three options have been chosen: the default, the defaul /10 and the default $\times$ 10. Finally another non-important variable `verbose`: the model will not output anything while running.

2.2.4 Long-Short Term Memory (LSTM)

LSTM is the temporal model. It is a Recurring Neural Network. "The Recurrent Neural Network (RNN) is an extremely powerful sequence model that is often difficult to train. The Long Short-Term Memory (LSTM) is a specific RNN architecture whose design makes it much easier to train. While wildly successful in practice, the LSTM's architecture appears to be ad-hoc so it is not clear if it is optimal, and the significance of its individual components is unclear" [23].

The easiest way to explain the LSTM is by using an intuïtive example. Say in frame two a person is crying, this could be because he or she is sad or is hurt. However if now you get extra information and *know* in frame one another person was attacking. It is straight away much more likely that the person in frame two was crying because he/she was hurt.

This is what the LSTM does, it tries to learn (and remember) what is important information

from previous states in order to be able to better understand the current. Or in the case of this research, better predict the next.

In order to achieve this the LSTM looks at a *batch* of instances (in the small example above, this batch was just 2: the current and the former frame). It looks through all the data, draws conclusions. Where a normal Neural Network (NN) would now be done, the RNN 'feeds' these conclusions back into a new NN (a visualization of this in Figure 2.6). Hence the 'Recurrent' part of the RNN.

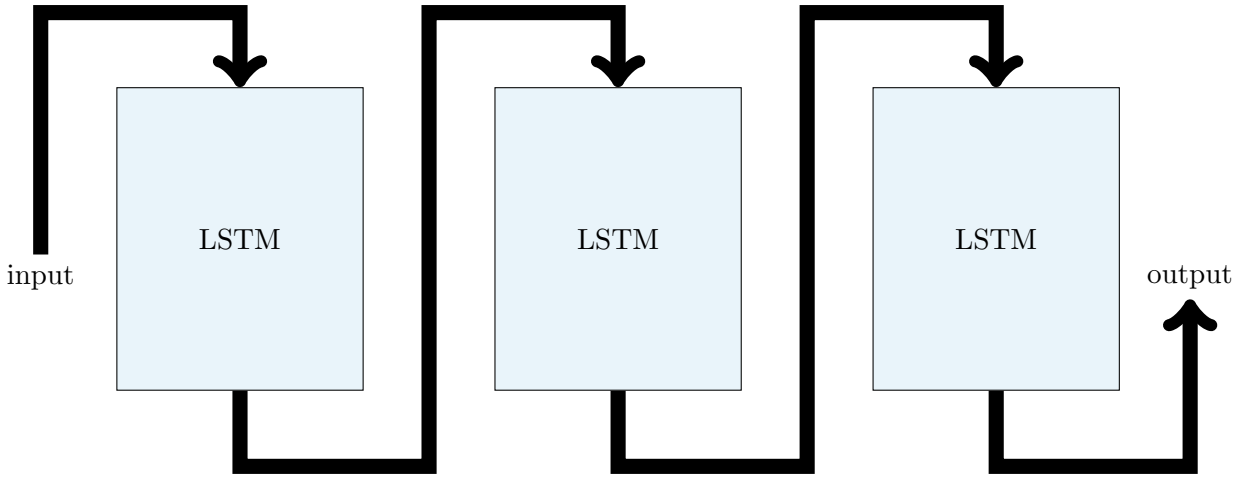


Figure 2.6: A visual example of an LSTM with 3 'layers'

2.2.4.1 Challenges regarding the temporal model

It turned out to take a lot more time (nearly 7 times as much[11]) to get it up and running compared to the other three models combined. This was due to a combination of several factors. First of all we had never worked with a temporal algorithm before. While we did work with non-temporal algorithms before. Having to research how to work with temporal data. The book 'Machine Learning for the Quantified Self: On the Art of Learning from Sensory Data' [24] turned out to be very helpful. It soon became clear that for temporal data a separate section of the database would be necessary. Since temporal data is so different from non-temporal data, all the engineered features had to be re-written. At this point it was decided that, because LSTM had already taken up much more time, those features would simply be left out of this model.

Secondly -as often the case with deep learning- it was impossible to figure out what exactly happens in the LSTM model. The uncertainty this brought was amplified by the unclear package documentation. In the end most information has been extracted from code-wise working examples on the internet

Once the LSTM model was working, it turned out to work very different from the other three models. The non-temporal models all have packages where the parameter inputs in one function determine how the model is set up. But LSTM is a Recurrent Neural Network and for that reason has many architectures. This caused that instead of having one function on which to perform the parameter grid search, with LSTM this could theoretically be an infinite number of functions. Resulting in an infinite number of models to search through for the parameter grid search. Since we want to find the best fitting model looking from a data point of view, we

[11] 20 days vs 3 days

looked for a setup of the LSTM in which a parameter grid search was still possible. To achieve this one major compromise had to be made: each 'layer' will be -parameter wise- exactly the same. Because of this choice, the number of layers can be seen as another parameter which can be tuned.

2.2.4.2 The proposed model

For a graphical representation of the workings of the LSTM model see Figure 2.7.

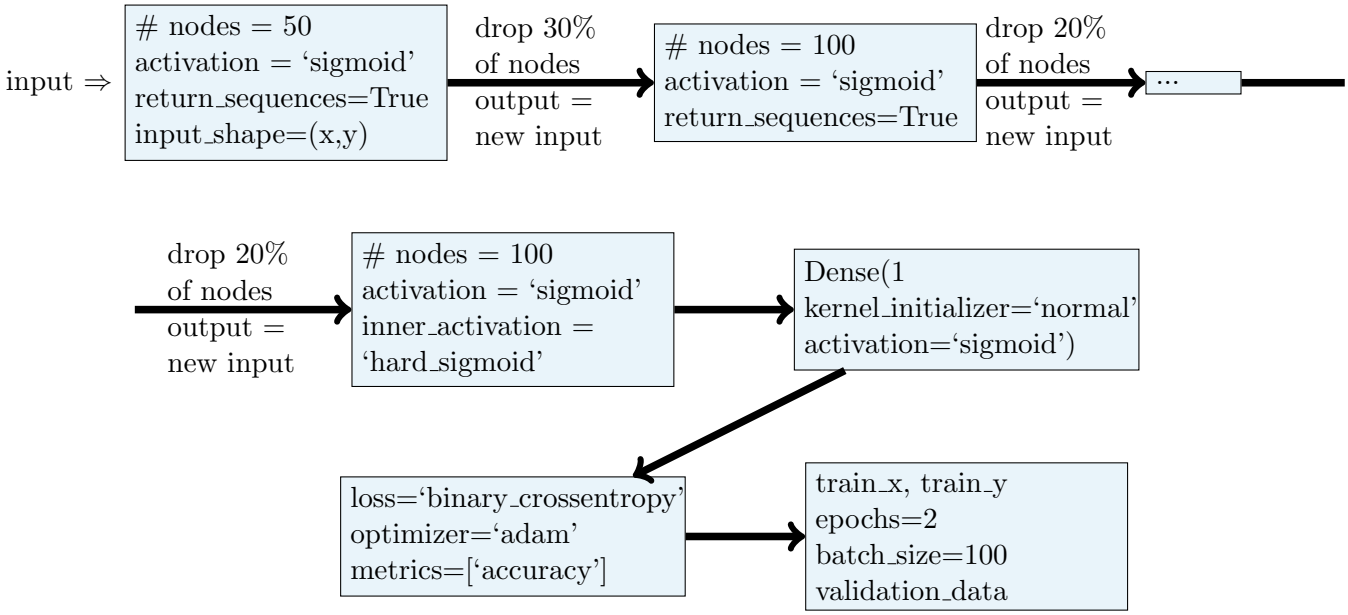


Figure 2.7: Inner workings of the LSTM combined with bits of the code

Only after all of these steps (of which each 'step' is a single line in the code) can a prediction be made.

It is unclear which variables the LSTM exactly has. With the use of the documentation, it was hard to get examples working. And in working examples, parameters are needed which are not documented (e.g. 'inner_activation'). But bring forth working code.

For this reason and all those mentioned above, a slightly alternative setup has been chosen for the LSTM.

Proposal for parameter grid search LSTM
An unlimited number of recurring steps can be chosen. All with either similar or different settings. With a % of nodes dropped to prevent over fitting or not. This results in an endless number of possibilities. That is unable to be caught in a parameter grid search. In order to be able to still perform a grid search, a number of assumptions will be made. The most important being: all layers will get exactly the same set of values for the parameters. This creates the option to have the number of layers as an additional parameter. Similarly what happens between the layers is the same between each layer. We are aware that this setup might exclude the optimal model.

The variables proposed to take into consideration with the grid search-LSTM are shown and explained in **??**. In Table 2.4 the values which will be tried in the grid search are shown.

name	grid search
n_layers	{ 1, 2, 3}
steps_to_look_back	{ 10, 60, 1440 }
activation	{ 'sigmoid' }
inner_activation	{ 'hard_sigmoid' }
units	{ 14, 81 }
kernel_initializer	{ 'random_uniform' }
loss	{ 'binary_crossentropy' }
optimizer	{ 'rmsprop' }
epochs	{ 2, 3 }
batch_size	{ 10, 60, 1440 }
unit_fotget_bias	{ True }
dropout	{ 0.2; 0.3}
threshold	{ 0.01; 0.1; 0.2 }

Table 2.4: All the variables settings which will be tested with regards to the LSTM model

The parameter `n_layers` does not actually exist. But it was deemed important to be able to take this parameter into consideration with the parameter grid search. For that reason it has been programmed. In order to program it, some assumptions have been made: the nodes (first - middle - final) will be created automatically, all with the same settings which have been determined by the other parameters. `N_layers` should actually not need more than 2 [25], so for good measure 1, 2 and 3 are tried.

The `steps_to_look_back` are chosen in such a manner that the look-back time will be respectively 10 minutes, an hour and a day.

Furthermore the `activation` will be set to 'sigmoid' because the end result should be between 0 and 1. For the same reason the `inner_activation` will be 'hard_sigmoid'.

As for the number of nodes in the hidden layers (here called: `units`) [25]:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

The output layer size is simple: one float between 0 and 1, thus output layer size = 1
However the input layer size is more complex. This depends on the number of steps we look back at. One timestep has two variables. Thus: input layer size = time steps × 2
Following the second rule, the formula should be something like:

$$\text{units} = \text{floor}\left(\text{steps\_to\_look\_back} \times 2 \times \frac{2}{3} + 1\right)$$

Meaning:

steps_to_look_back	size input layer	size output layer	units
10	20	1	14
60	120	1	81
1440	2880	1	1921

However this turned out to have an extremely negative effect on the run time[12], for this reason the number of units 1921 has been left out.

`Kernel_initializer` will be set to 'random_uniform' between 0 and 1. Because a scaler has been used to set all the variables between 0 and 1.

For this same reason (results having to be between 0 and 1) the `loss` function will be set to 'binary_crossentropy'.

"This optimizer[red. 'rmsprop'] is usually a good choice for recurrent neural networks." [26] For this reason the `optimizer` will be set to 'rmsprop'.

To see whether `epochs` matter a lot to start with, the options 2 and 10 were chosen. However during the running of the models it turned out that this slowed down the model too much[13], so therefore the options 2 and 3 have now been chosen: still there could be a difference, but the time needed should be cut down by $^{7}/_{12}$[14].

With `batch_size` a similar reasoning as with steps_to_look_back has been followed: 10 minutes, an hour and a day are tried.

According to [23] "The LSTM with the large forget bias outperformed both the LSTM and the GRU on almost all atsks[sic]". For that reason the parameter `unit_forget_bias` is set to True.

The regularizers will not be used because the `dropout` will be used. To see if and what type of effect this variable has, the number 0.2 and 0.3 will be tried.

Finally `threshold`: this variable does not exist for the LSTM model either, but has been created for the grid search. Since there are very few defects, it is expected that a lower threshold works better. For good measure some higher values will be tried too.

2.2.5 Comparing techniques

To be able to compare all the models and draw any conclusions from them, a validation method has to be chosen. There are four optional validation metrics: accuracy, recall, precision and F1-score.

The following abbreviations will be used:
TP = True Positive
TN = True Negative
FP = False Positive
FN = False Negative
Well-known from the confusion matrix[15].

Accuracy
As it is important to predict correctly both when a part will go into defect and when it will

[12] This was tested on the first 10,000 instances of pumping station PSa. While all other variables were kept the same and only the units where altered. Using 14 or 81 units took about 2 or 3 seconds per epoch, while using 1921 units it took 4124 seconds per epoch.
[13] After 36 hours only 128 out of the 649 possible parameter combinations had been tried for one data source
[14] $\frac{\text{original nr} - \text{new number}}{\text{original nr}} = \frac{2+10-(2+3)}{2+10} = \frac{7}{12}$
[15] https://en.wikipedia.org/wiki/Confusion_matrix

not, the 'accuracy' has been chosen as one of the measurement to compare the models by.

$$\text{accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FN} + \text{TP} + \text{FP}}$$

Informally: 'what percentage has been predicted correctly?'

Important to state here is that this metric makes it slightly more difficult to compare the pumping stations among each other. Since the percentage of defects is not similar. Ranging from only 1.56% to 15.78%, as can be seen in Table 2.6.

putnr	# defects	# days with starting defects	total time in defect	part of the time defect	% of days with starting defect
PSa	45	23	3 days 4:22:14	0.18	1.33
PSb	163	85	129 days 8:43:42[16]	7.46	4.90
PSc	239	103	3 days 15:27:31	0.21	5.94
PSd	71	36	4 days 12:21:38	0.26	2.07
PSe	68	27	0 days 15:45:38	0.04	1.56
PSf	22	12	0 days 2:24:41	0.01	0.69
PSg	68	24	1 days 1:29:4	0.06	1.38
PSh	69	22	3 days 11:32:48	0.20	1.27
PSi	28	10	15 days 16:26:24	0.90	0.58
PSj	69	47	0 days 14:54:48	0.04	2.71
PSk	38	28	1 days 2:26:36	0.06	1.61
PSl	77	27	3 days 10:56:10	0.20	1.56
PSm	28	13	0 days 23:15:13	0.06	0.75

Table 2.5: General information on the number of defects per pumping station before data preparation. The total number of days is 1,735.

[16] During the research it was found that the duration of the defects differed a lot in range. Most of them seemed to take less than a day, so first the assumption was proposed that such a defect could take no longer than 24 hours and if it did, it probably meant the sign signing the defect off was missed. However after consulting an expert this assumption was dropped, and no further assumptions were made with regards to the duration of a defect. Resulting in one defect in pumping station PSb already taking 120 days.

putnr	after data preparation % of instances with defect in 3 days non-temporal data
PSa	60 / 1736 = 3.46 %
PSb	230 / 1736 = 13.25 %
PSc	274 / 1736 = 15.78 %
PSd	98 / 1736 = 5.65 %
PSe	77 / 1736 = 4.44 %
PSf	36 / 1736 = 2.07 %
PSg	66 / 1736 = 3.80 %
PSh	57 / 1736 = 3.28 %
PSi	27 / 1736 = 1.56 %
PSj	138 / 1736 = 7.95 %
PSk	77 / 1736 = 4.44 %
PSl	74 / 1736 = 4.26 %
PSm	39 / 1736 = 2.25 %

Table 2.6: General information on the number of defects per pumping station after data preparation

For this reason an accuracy of 90% for pumping station PSc would be an improvement, whereas the same result for pumping station PSi would be worse than simply always predicting no-defect.

Recall
This metric returns how many of the defects are labeled correctly. However it does not measure in any way if the non-defects are labeled correctly.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

If this metric is high it means that when the mechanic gets no signal saying there is a defect, there will indeed not be a defect. But when he gets the signal that the pumping station might break down, it could still only be a very low change on an actual defect.

Precision
This metric returns how many of the 'labeled defects' are actual defects.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

When the precision is high, it means that when a mechanic gets the signal that the pumping station might go into defect, the pumping station probably will go into defect. However when the model gives no sign, the pumping station could still be going into defect.

F1-score
As recall and precision have their downfalls, a new metric was brought to life, one which combines the strengths of both: the F1-score.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

With a high F1-score the mechanic can feel at ease when he gets no alarm, and when he gets the message a defect is drawing near, the defect is probably really drawing near.

Conclusion
As a final aim of this research is to no longer having to do unplanned work in the weekend or

during the night due to an unforeseen defect, recall is definitely important. However if the model predicts every instance as a defect - therefore having the highest possible recall- the model is not trustworthy either. So some of the precision has to be taken into account as well. For this reason it has been decided to focus on the `F1-score` as a comparative means between the models.

F1-score has a tendency to 'follow' first and for most whichever has the lowest score of precision and recall (further down in the report some figures show this quite clearly, for instance Figure 3.2 and Figure 3.46). To find a somewhat meaningful 'under bound' for the F1-score, the following steps will be taken for each pumping station separately:

1. Assume - because of the skewed data - that precision is going to be the biggest issue. With this assumption and that a lower bound is being looked for, recall will be set to 1 for this calculation.

2. To get the value for precision, look at what value it would have if everything was predicted to be a defect, these values can be found in Table 2.6.

3. Calculate the F1-score belonging to this situation. If the F1-score is better than this lower bound, the model is returning a better precision.[17]

Following these steps results in the following:

putnr	calculation	Under bound F1-score
PSa	$2 \times \frac{0.0346 + 1}{0.0346 \times 1}$	= 0.0669
PSb	$2 \times \frac{0.1325 + 1}{0.1325 \times 1}$	= 0.2340
PSc	$2 \times \frac{0.1578 + 1}{0.1578 \times 1}$	= 0.2726
PSd	$2 \times \frac{0.0565 + 1}{0.0565 \times 1}$	= 0.1070
PSe	$2 \times \frac{0.0444 + 1}{0.0444 \times 1}$	= 0.0850
PSf	$2 \times \frac{0.0207 + 1}{0.0207 \times 1}$	= 0.0406
PSg	$2 \times \frac{0.0380 + 1}{0.0380 \times 1}$	= 0.0732
PSh	$2 \times \frac{0.0328 + 1}{0.0328 \times 1}$	= 0.0635
PSi	$2 \times \frac{0.0156 + 1}{0.0156 \times 1}$	= 0.0307
PSj	$2 \times \frac{0.0795 + 1}{0.0795 \times 1}$	= 0.1473
PSk	$2 \times \frac{0.0444 + 1}{0.0444 \times 1}$	= 0.0850
PSl	$2 \times \frac{0.0426 + 1}{0.0426 \times 1}$	= 0.0817
PSm	$2 \times \frac{0.0225 + 1}{0.0225 \times 1}$	= 0.0440

Table 2.7: F1-score under bound per pumping station

The highest lower bound -presented by pumping station PSc- is 0.2726. This will be the threshold used later on the check how well the models perform.

[17] Since a recall or precision score of 1 is very exceptional, the model is most probably at this very F1-score already doing better.

An often used method of getting a lower bound is by using a 'random prediction' (50% will be said to be defect at random). With pumping station PSc this resulted in a precision of 0.152 and a recall of 0.485, creating an F1-score of 0.232. As this is even lower than the above mentioned method returned, the threshold achieved by the first method will be used.

2.3 Software used

For completeness an overview of the software used with this research.

First of all PgAdmin[18] (and with that psql) was used for all the database queries. Once all the data was gathered, Python[19] was used for most of the feature engineering, data processing, running the models and creating the figures. During this process IDE gitbash[20] was used. For the report LaTeX[21] was used in TeXstudio[22]. For some final touches (like removing the Python-title from figures) paint was use. And finally the empty values (Table 2.8), the important features (Table 3.1), the average emptiness of important features (Table 3.2, Table 3.3) and the maximal F1-scores (Figure 3.51) were all manually inserted into excel to be able to play around easily on the search for insights.

2.4 Methodologies

2.4.1 Data preparation

2.4.1.1 Feature engineering

'Feature Engineering Is The Key' [27] as Pedro Domingos puts it. 'Often, the raw data is not in a form that is amenable to learning, but you can construct features from it that are.' [27]. For this very reason the researchers had originally planned to take out a month to find all sort of creative and possible meaningful features. However the process of getting the LSTM to work (more on this see section 2.2.4.1) took a lot longer than anticipated. And had the time reserved for feature engineering drastically shortened. For the same reason it has been decided to not create any of these features for the LSTM, as that would probably again take a lot of time. For the non-temporal models some features were engineered.

2.4.1.2 Non-temporal data

To start with, all the sensor data within the catchment of the current pumping station has been aggregated into a minimum, maximum, average and the sum of all the values per day per sensor.

Secondly from the dates it has been extracted which weekday it was (Monday = 0, Sunday = 6). And a distinction has been made between workdays and weekend days. Thirdly the season got extracted (winter =1, spring=2, summer=3, autumn = 4).

To add a bit more information on the flow measurements, two additional features have been created, to start with 'maximum_5_minute_difference_flow_measurement'[23] which keeps the maximum 5 minute difference of the flow measurements on that given day. The next being 'number_flow_measurement_turned_on' [24] which keeps track of the number of times the pumping

[18] https://www.pgadmin.org/
[19] https://www.anaconda.com/download/
[20] https://git-scm.com/downloads
[21] https://miktex.org/download
[22] https://www.texstudio.org/
[23] In Dutch: 'debiet_max_5_min'
[24] In Dutch: 'debiet_aantal_keer_aan'

station had to switch on.

As mentioned in the introduction, the two most telling features from the research from Marly van der Meij [13] have been added: defects_ratio[25] which is the average % of time the pumpingstation was in defect the past 3 days. A noteworthy major difference is that her research aggregates the data per 10 minutes while the non-temporal data in this research is aggregated per day. Her second important feature is 'pumped'[26] which entails the amount of water pumped by the pumping station divided by the total time the pumping station was not in defect. It is important to note here that Marly's research also made different assumptions with regards to the defects.

Finally the feature 'dagen_sinds_leegpomp' ('days_since_emptying' in Dutch) has been added. The reason for this can be clearly seen in Figure 2.12. This figure is created with the outlier detection code provided by [24]. The values in the red box (which has been manually added) only happen when the pumping station is manually 'emptied'. This happens by sucking up the remaining water and dirt. Instead of seeing these as outliers, it has been decided that this 'cleansing' of the pumping station could have something to do with its defects. Being it for better or worse. Therefore the number of days passed since the last cleansing has been added as a feature.

2.4.1.3 Temporal data

In the case of the temporal data, only the measurements exactly on the given pumping station are taken into consideration. This is done because temporal data is looked at as a time series. And in this case there are already two time series effecting each other.

[25] In Dutch: 'storingsratio'
[26] In Dutch: 'verpompt'

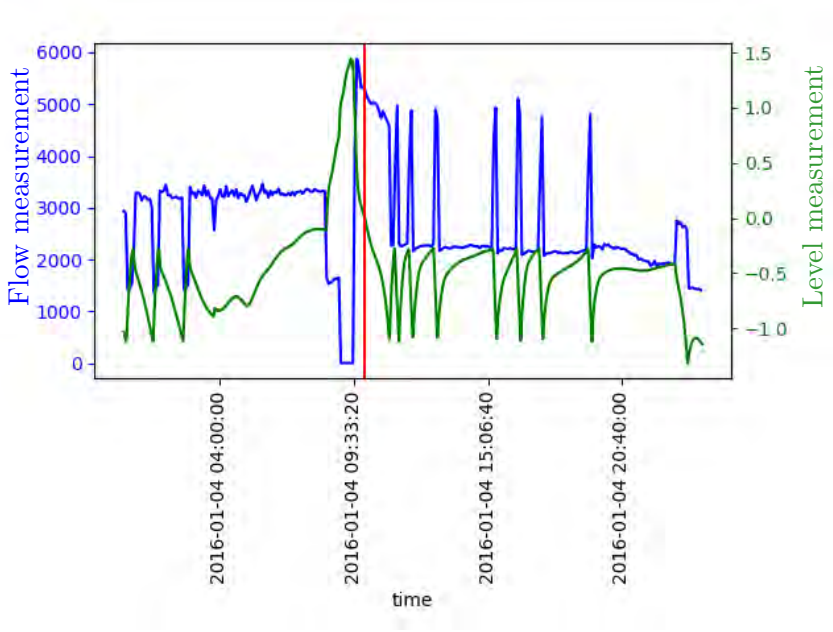


Figure 2.8: Visual representation of the temporal time series of pumping station PSb. The red vertical line represents a defect. It can be seen that as soon as the level measurement reaches a certain level, the first pump starts pumping and the flow measurement jumps up. In the middle it appears to be raining and a second pump starts to help pumping. In this particular time interval a defect has even been 'caught'.

2.4.1.4 Empty variables

The empty values have to be treated differently for the temporal and the non-temporal methods. This is because the non-temporal methods can deal with 'gaps in time' whereas the temporal methods cannot deal with such gaps.

Non-temporal data
Since it turned out that quite a lot of data was missing for the non-temporal data (see Table 2.8) the original plan of dropping each row with a missing variable did no longer seem like a good solution. It was decided that when more than 25% of the data is missing, the column would be all together dropped: having more than $^1/_4$ 'make-belief' seems too much while still keeping a decent number of variables. Secondly -to fill out the missing values- the choice has been made to want to preserve as much as possible from the original 'distribution' while having a logic reason to choose that number. In the case of flow rate it was chosen to set this to 0, meaning: 'the pumping station is currently not pumping'. For the level measurement this was a little bit more tricky. But eventually it was decided to use the average of that particular level measurement to fill the empty values with.

In Figure 2.9 and Figure 2.10 an example is given of how a level measurement was influenced by these decisions. The boxplots show that the quantiles are barely altered. However the histogram shows that despite best efforts choosing only one value to input has its drawbacks.

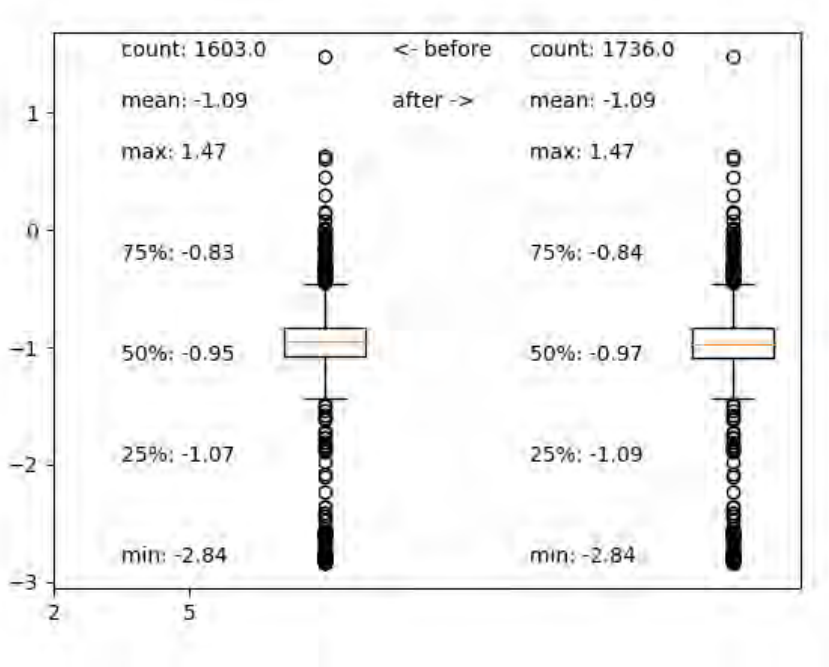


Figure 2.9: A comparing boxplot of a level measurement before and after the data preperation

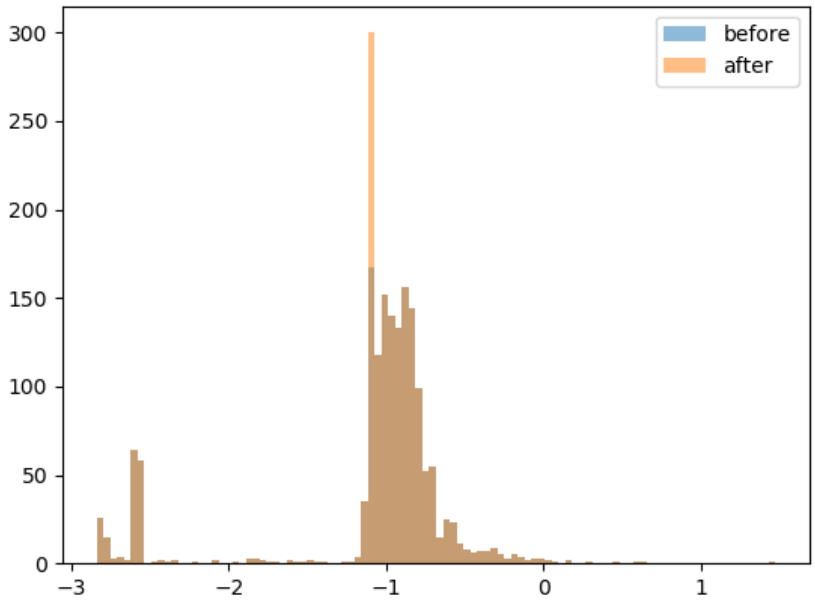


Figure 2.10: A comparing histogram of a level measurement before and after the data preperation

For each chosen pumping station an overview has been created to show how many variables are missing. This is only done for the non-temporal data. These can be found in the back. An short overview has been created in Table 2.8. In general it can be said that a lot of data is missing, which naturally leads to complications.

pumping station	average portion missing values	table with missing value data
PSa	0.48	??
PSb	0.40	??
PSc	0.38	??
PSd	0.36	??
PSe	0.28	??
PSf	0.28	??
PSg	0.39	??
PSh	0.48	??
PSi	0.23	??
PSj	0.07	??
PSk	0.12	??
PSl	0.12	??
PSm	0.50	??

Table 2.8: For each pumping station, the average portion of missing values and a link to the table with the missing values per variable for the non-temporal data of that pumping station is given

Temporal data
In case of the temporal method it is more difficult. Simply filling out a missing value is hard since this is minute data. Neither is it possible to deduce the missing variable from surrounding variables since most of the time if one variable is missing, all of them are missing. Only three options remain: (1) interpolating all the gaps. In which case valid questions may rise on what those values would actually mean. If minute data is linearly interpolated for multiple days. The next option is (2) 'cut' the data where variables are missing and choose the biggest 'chunk' to learn on. This would throw away a lot of potentially valuable data. The final option is (3) to fill-out 'weird' numbers for the missing data, for instance '-999'. Hoping the algorithm would learn that those numbers have no real meaning.

Originally option (3) was chosen. But early in the process this seemed to lead to very peculiar results[27]. As we were very much opposed to 'throwing away' valuable data option, (1) seemed the only one left. As early results indicated more reasonable results, the choice has been made to go with option (1): interpolating the data.

2.4.1.5 Outliers

The outlier detection provided by [28] is used.

[27] All test instances were predicted to have exactly the same chance on a defect

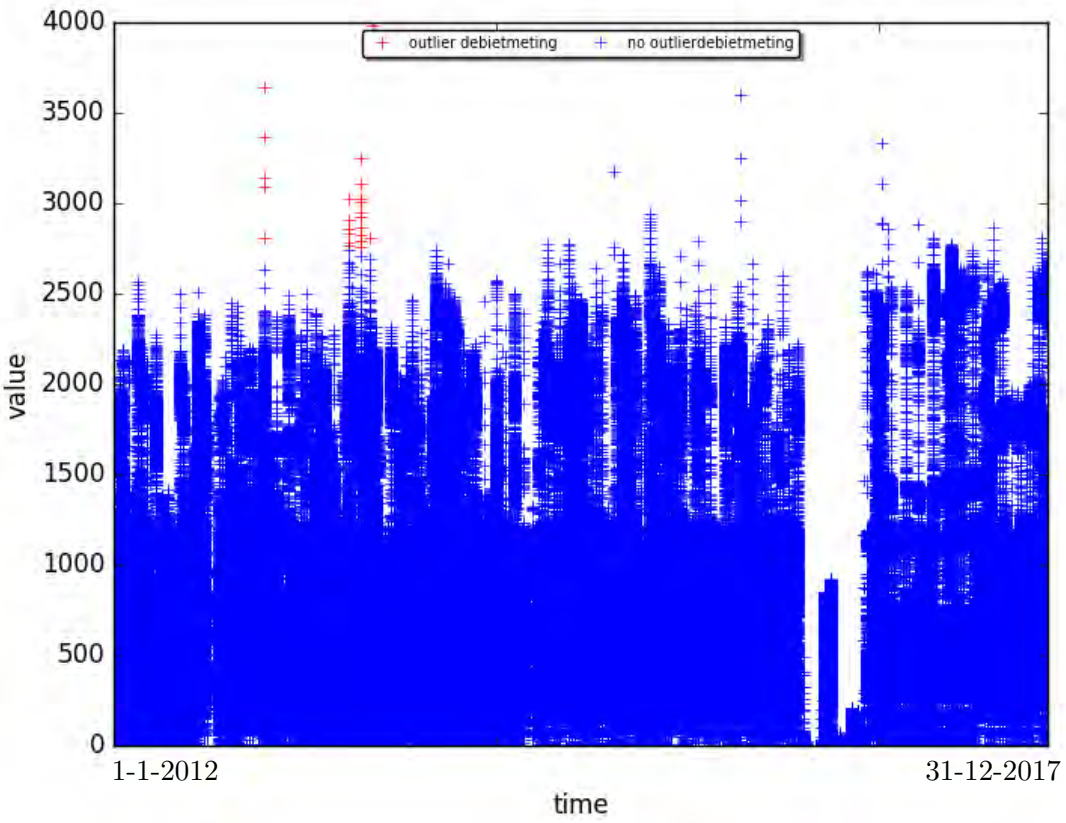


Figure 2.11: According to the code the outliers with regards to the flow measurements

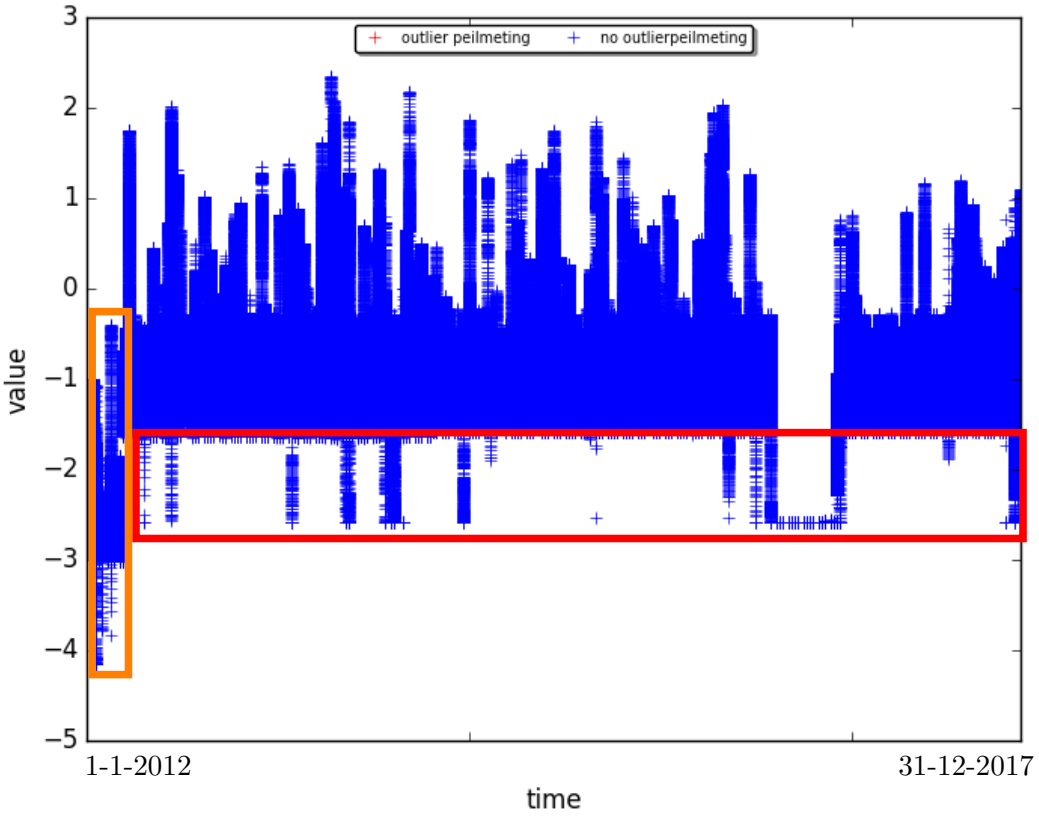


Figure 2.12: Outlier detection example of the level measurement of pumping station PSb. In red the 'outliers' caused by emptying the pump manually. And in orange a visual result of different settings of the predetermined starting level[28].

In Figure 2.11 and Figure 2.12 the outliers of pumping station PSb can be seen according to the code from [28]. It is interesting to note that the code and the domain experts differ profoundly on this matter. Starting with the flow measurements. The code marked some points as outliers. When looking more closely at the data together with a domain expert, she stated that those are perfectly reasonable values. For that reason those 'outliers' have been deemed as false-positives. And been treated as normal values.

As for the level measurement the opposite happened. The code marked no values as outliers. However as soon as the domain expert saw this figure, she immediately called all the values in the red box possible outliers. After some digging in the data it turned out those are points where the pumping station is emptied. As this might carry important information with regards to the defects, those values are left in. And in addition an extra feature is engineered: 'dagen_sinds_leegpomp' (see subsubsection 2.4.1.1).

2.4.2 Selecting the defects

Selecting the type of defects

The 75 defects in the database vary enormously in severity from a defect light bulb to a pumping station which stops pumping. In this research the choice has been made to only look at the severe cases where a pumping station stops pumping. Arjen Kruit [12] chose to only look at

[28] In Dutch: 'inslagpeil'

those severe defects as well. He narrowed it down to eight classes of defects, shown and explained in Table 2.9.

defect name	explanation
no pump available	the pump has been shut down, usually this is done manually by a mechanic
watchdog[29]	an energy event
thermal security	slide event[30]
mainstream security	the security on the main energy stream of the pump
flow rate measurement defect	an event in the flow rate measurements
shuttle alarm	the pump switches on and off too often and too quick in procession
power steering alarm	energy sent to a specific part of the machine, only a little energy is necessary
power failure	no more energy is supplied. This often happens when outside of the sewer system accidentally an energy supply line has been cut.

Table 2.9: The defects Arjen Kruit chose to predict

This set of defects is taken as the base set, and some minor differences are implemented. First of all because the energy events usually have their origin outside of the sewer system, the chances of being able to predict them with only sewer information are estimated to be low. For that reason 'watchdog' and 'power failure' are chosen to be left outside the scope, and even though they are major failures, no effort is made to predict those in this research.

Three other defects are major events too and have been included in this research. To start with the defect 'Opening and pump stopped'[31], meaning as much as that the pump stopped working and that the valve to keep the water from flowing back is open. So in this case the water can go into the opposing direction of which it should go. Secondly the defect 'Closing and pump works'[32], meaning that despite the pump pumping, this time the valve is closed and stopping the water from flowing correctly. Finally the defect 'Return flow notification'[33], meaning that the system detects water flowing in the wrong direction.

Creating the final list of defects to look at in Table 2.10.

defect name
no pump available
thermal security
mainstream security
flow rate measurement defect
shuttle alarm
power steering alarm
opening and pump stopped
closing and pump works
return flow notification

Table 2.10: The defects to be predicted

[29] In Dutch: 'netwachter'
[30] In Dutch: 'schuif event'
[31] In Dutch: 'openmelding en pomp uit bedrijf'
[32] In Dutch: 'dichtmelding en pomp in bedrijf'
[33] In Dutch: 'retourflowbewaking'

Cleaning the defects
Now the type of defects had been chosen, there was still some cleaning to be done. For instance the data sometimes showed that a mechanic would go to a pump and until the mechanic left again many defect messages would pass by: the mechanic manually restarted the pumps. Obviously those are not the type of defects this model wants to predict. For this reason any defect occurring when a mechanic is present is disregarded.

In very rare occasions it happened that no starting or no ending time was known regarding a certain defect. Since the time a defect took is needed for the feature engineering (see subsubsection 2.4.1.1), an effort was first made to look at a possible common length that defects took in order to fill that out. But this turned out not to be the case. And because it rarely happened (some pumping stations had no occurrence of it), it was decided to simply disregard those defects.

In the database, information on the priority of a defect is already included. Since the scope of this research is limited to major defects, only the defects with the highest priority will be predicted.

Furthermore, some of these defects were not only limited to the pumping stations alone. Since this research really narrows down to pumping stations, all the defects occurring at another location then where a pumping station is present were disregarded.

And finally, once the first notification of a defect was sent, it happened regularly that the same notification would be sent over and over again every time with a few seconds or minutes in between. The assumption has been made that those notifications were all the cause of one and the same defect. For this reason only the first time a specific defect occurs on any given day will be seen as a 'real' defect.

Comparing the defects in researches
To be able to better compare the results found in this research to the results found in the research of Marly van der Meij [13], a quick comparison between the decisions made with regards to the defects is in order. To start with, her research included more defects. Not only more types but also including those with one priority level lower than this research does.

2.4.3 Selecting the pumping stations

Throughout time different types and brands of pumps have been installed in the sewer system. Because of this not all pumps might have the same 'indicators' in their data to show an approaching defect. Therefore not all the pumps can be trained in the same model. In order to keep this research within the time limit, the choice has been made to pick around 10 pumping stations and look closer into those.

To select those 10 pumping stations, two demands have been made:

- This particular pumping station needs to go into defect at least 30 times (a bit over 1% of the time).
  We are aware that this causes a bias in the data. However it was deemed more important to be able to give the models some defects to learn from.

- This pumping station needs to have both the Flow rate measurements and the Level measurements.

This left a list of 13 candidates presented in Table 2.11.

pumping stations	
PSa	PSh
PSb	PSi
PSc	PSj
PSd	PSl
PSe	PSk
PSf	PSm
PSg	

Table 2.11: List of 13 pumping stations who met all the demands

Since there is no 'good' method to make a viable distinction between those pumping stations, the choice has been made to look into all 13 pumping stations.

Please note that during this part of the research, Marly van der Meij's research [13] was not yet published. After her research was published, some decisions on how to look at the defects have been altered. In order to be able to use some important features of hers (see subsubsection 2.4.1.1), the duration of the defects was required. At the start this was not taken into account. If a defect had no end time, it was deemed 'no defect'. For that reason it is possible that in Table 2.5 the pumping stations PSf, PSi and PSm have fewer than the here mentioned 30 defects.

Noteworthy is that Marly van der Meij's research focusses on pumping station PSe. It will be interesting to compare the results.

2.4.4 Selecting the final variables

Since all variables turned out to have an extremely low correlation with going into defect[34], it was decided to look at the feature importance function of the decision tree (Figure 2.4). This is a feature which shows how important the decision tree deems which variable with regard to 'splitting' a node (in the example the split is done by the number of wheels of the vehicle). Only variables with a feature importance > 0 are kept and fed to the actual models.

[34]Somewhere along the lines of: no 'absolute correlation' being higher than 0.2

Chapter 3

Results

This chapter will display all the results. Starting with a comparison between the original data and the data used, and which features were important (section 3.1). Moving onto, for each algorithm, which parameters turned out to be most important (section 3.2). The predictions made by the algorithms (section 3.3) and finally some insights which can be gained from these predictions (subsection 3.3.1).

3.1 Data

3.1.1 Final dataset vs original dataset

Out of the 12 variables described in section 2.1 only two could be reliably used. Aggregating their minute-data to daily-data. And out of the 525 pumping stations only 13 where chosen in subsection 2.4.3 to look at, however all the data from the respective catchments could be used. Resulting in using from the original nearly 380 million measurements just slightly less than 145 million. Also out of the nearly 3000 types of defects only 9 types are taken into consideration as explained in subsection 2.4.2.

3.1.2 Feature importance

The figures **??** up till and including **??** show for each pumping station the 10 most important features. Often `defectsratio` is the most important feature. At the same time it never reaches over an importance of 0.25. And this is a scale from 0 to 1. So the result is still similar to the correlations: no variable truly gives a lot of information.

In Table 3.1 it is shown whether a feature was in the top 10 most important features for that pumping station. As each pumping station could have several level and flow measurements, those were all 'judged as one', to look more at whether level or flow measurements were important at all, or maybe always a certain aggregation form.

It can be seen that for each pumping station the `defectsratio` is important. This is straight away the only feature which is important to all pumping stations. In general the `level measurement` is used for each pumping station as well. But not all stations prefer the same aggregation method. The most used aggregation method is the minimum level measurement, which is important to all but one pumping station (PSl).

Another important feature, which is used by all the pumping stations but two (PSc and PSl), is the `days_since_emptying`.

Next are the `maximum_5_minute_difference_flow_measurement` and the `flow measurement`. Both are used by 8 pumping stations. As regards to the flow measurement the maximum seems the most important aggregated value because it is used in 7 of the 8 cases, where all the others are used less. The minimum was not important once.

The other values `pumped`, `weekday` and `season` are all used in less than half the cases. And season even only once.

pumpingstation	defectsratio	days_since_emptying	maximum_5_minute_difference_flow_measurement	pumped	weekday	season	level measurement		flow measurement	
PSa	X	X	X	X	X		max avg	min		
PSb	X	X	X	X	X		max avg	min		
PSc	X						max avg	min sum		
PSd	X	X	X				max avg	min sum	max	
PSe	X	X					avg	min sum	max avg	sum
PSf	X	X		X		X	max	min sum		
PSg	X	X	X	X			max avg	min sum	max	sum
PSh	X	X		X	X		max avg	min	max avg	sum
PSi	X	X		X			max avg	min sum	max	
PSj	X	X	X		X		max	min	max	
PSk	X	X	X				max avg	min sum		
PSl	X		X		X		max avg	sum	max	sum
PSm	X	X	X		X		avg	min sum	avg	

Table 3.1: Overview of the important features for each pumping station. When a field is empty it means that it is not one of the 10 most important features for that pumping station.

Interesting to note here is that the weather data imported from KNMI was not deemed impor-

tant for any pumping station.

In order to be able to compare the results, the following two tables Table 3.2 and Table 3.3 contain the average emptiness of the variables which were deemed important and with that only the features which were used. Two methods of averaging have been tried:

$$\text{average emptiness} = \frac{\text{sum of emptiness}}{\text{number of important features}}$$

$$\text{weighted average emptiness} = \frac{\sum \left(\text{emptiness} \times \text{importance}\right)}{\text{sum of importance}}$$

Furthermore the difference between these two tables is that Table 3.2 only takes the 10 most important features into account whereas Table 3.3 takes all features into consideration with an importance > 0.

putnr	number of important features	weighted average emptiness	average emptiness
PSa	10	0.079	0.090
PSb	10	0.055	0.057
PSc	10	0.057	0.077
PSd	10	0.082	0.093
PSe	10	0.053	0.064
PSf	10	0.042	0.063
PSg	10	0.057	0.065
PSh	10	0.065	0.093
PSi	10	0.055	0.079
PSj	10	0.040	0.054
PSk	10	0.058	0.066
PSl	10	0.046	0.066
PSm	10	0.102	0.114

Table 3.2: This table contains only the features and the average emptiness of the 10 most important features

putnr	number of important features	weighted average emptiness	average emptiness
PSa	17	0.079	0.087
PSb	39	0.080	0.100
PSc	64	0.071	0.084
PSd	43	0.099	0.113
PSe	26	0.059	0.069
PSf	17	0.045	0.065
PSg	15	0.056	0.059
PSh	16	0.067	0.095
PSi	14	0.056	0.080
PSj	30	0.052	0.065
PSk	39	0.072	0.080
PSl	32	0.057	0.072
PSm	25	0.113	0.122

Table 3.3: This table contains the average emptiness of all the features with an importance bigger than 0

3.2 ML models

3.2.1 Models

In order to find the effect of differently filled out parameters with these models, the best model (e.g. the model with the highest F1-score) with each algorithm-pumping station couple is chosen. If multiple models share the best score, the first model is elected. Then all the parameters are kept the same but one. This last parameter is changed into all possible values and the score is compared. This is done for all the parameters.

Important to note here is that not all models could be taken into account. Some parameter settings were not viable -resulting in no score- and in the cases where either the TN, FN or the FP where 0, the F1-score returned a 'nan' value and could thus not be evaluated using this method. In this particular case not a lot is lost due to this, because in these cases it means that at least either the recall or the precision is 0, and thus there would be a low F1-score.

3.2.1.1 RF

Some parameters simply have no effect whatsoever. For instance the `n_jobs` and the `warm_start`. Which resulted in figures like Figure 3.1. Also the `oob_score` had no result. However in 4 pumping stations any other value than the currently best value in combination with the other parameters resulted in a nan-value for the F1-score.

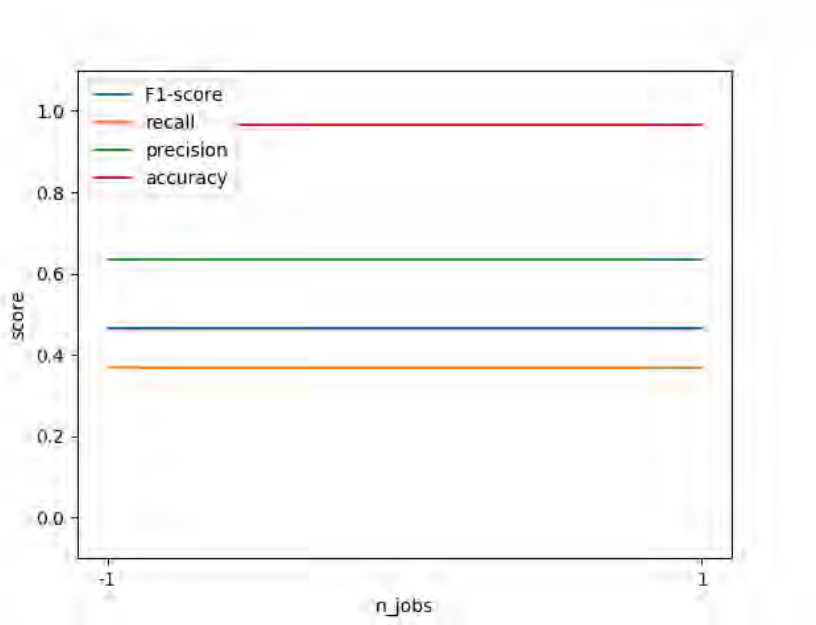


Figure 3.1: The F1-score of -in this case- pumping station PSd. When the variable n_jobs is changed, nothing happens.

n_estimators

This value seems to change a lot mostly between the values 5 and 50 and less so moving towards 500. It seems as if the lines in general seem to move towards each other at some point. Resulting in the idea in some cases that the optimal value is just outside the chosen parameters (e.g. Figure 3.2). But sometimes it seems to be exactly within (Figure 3.3).

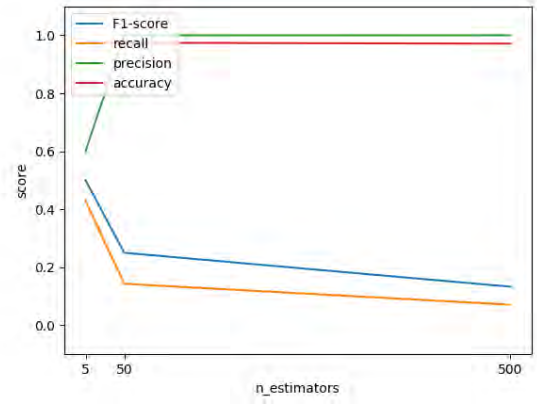


Figure 3.2: The effect of n_estimators on the metrics for pumping station PSg

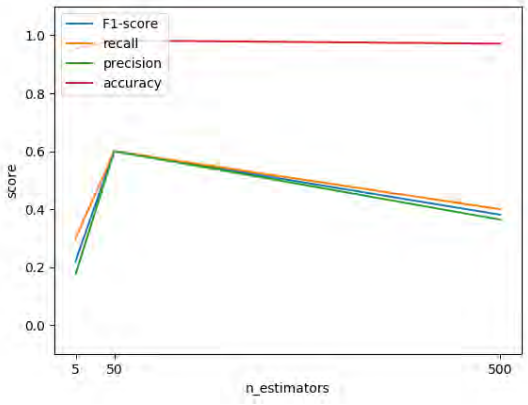


Figure 3.3: The effect of n_estimators on the metrics for pumping station PSi

criterion

Except for pumping stations PSa and PSm (Figure 3.4), entropy seems to be the better choice in general. With all other pumping stations resulting in figures like Figure 3.5.

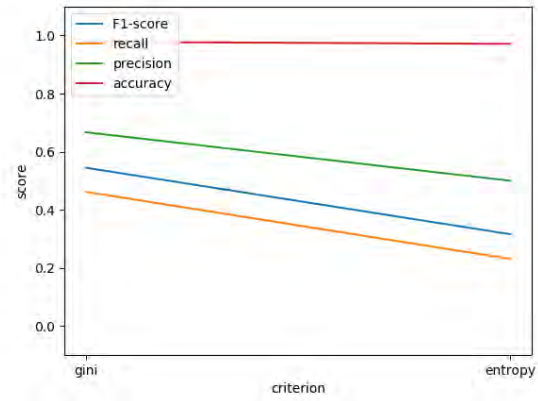


Figure 3.4: The effect of criterion on the metrics for pumping station PSm

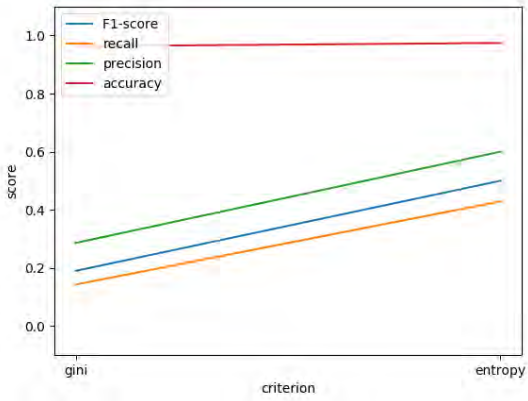


Figure 3.5: The effect of criterion on the metrics for pumping station PSg

max_features

Pumping station PSm only allowed for 0.5 and 0.75 to be tried, the other options resulted in nan-values (Figure 3.6). In general this parameter seems to have a bigger influence on the precision and recall (and with that the F1-score) than on the accuracy. Further there is not much to be said about it in general. It goes from quite rigid (Figure 3.7), to smooth and moving in sync (Figure 3.8), to complementing each other (Figure 3.9).

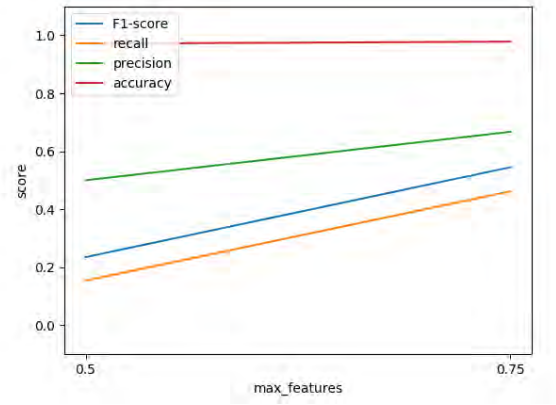


Figure 3.6: The effect of max_features on the metrics for pumping station PSm

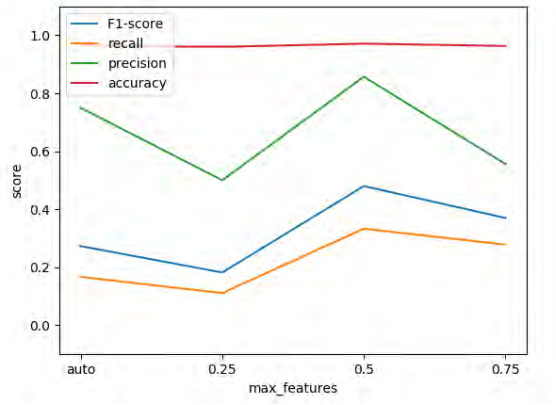


Figure 3.7: The effect of max_features on the metrics for pumping station PSl

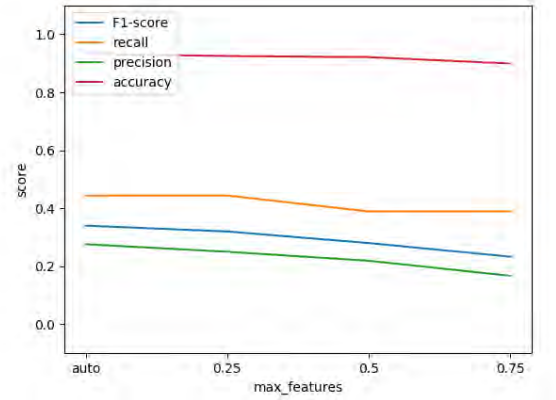


Figure 3.8: The effect of max_features on the metrics for pumping station PSe

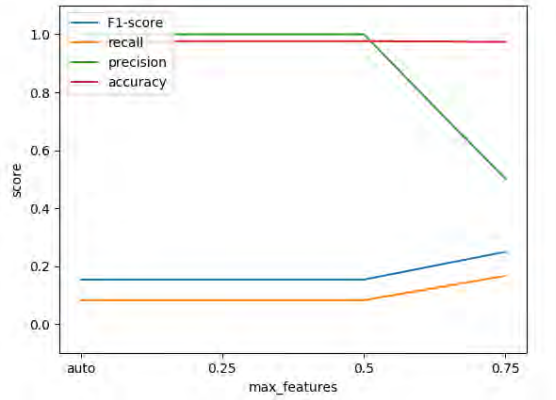


Figure 3.9: The effect of max_features on the metrics for pumping station PSf

Interesting to note here is that apparently the catchment does not have a lot of influence in this matter, as Figure 3.8 and Figure 3.9 are both from catchment 08.

min_samples_split

Only for two pumping stations (PSe and PSi) (Figure 3.10) is the value 0.1 best. In all other cases 2.0 is the best. Import to note here is that 0.1 means 10% of the samples. Which is a lot higher than 2. So in that sense the graph can easily be wrongly interpreted. Looking back it would have been great to see what would happen if more values had been tried[1]. It seems like possibly the point where recall and precision meet results in the best solution, Figure 3.11 is one of the clearest examples.

[1]Running all algorithms with their settings already took little over a week (day and night). So no more values were tried to not let the running take even more time.

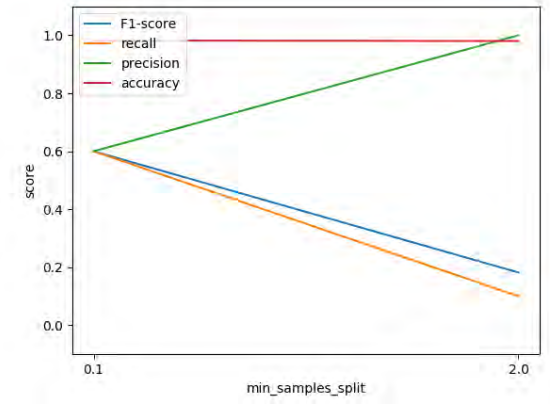


Figure 3.10: The effect of min_samples_split on the metrics for pumping station PSi

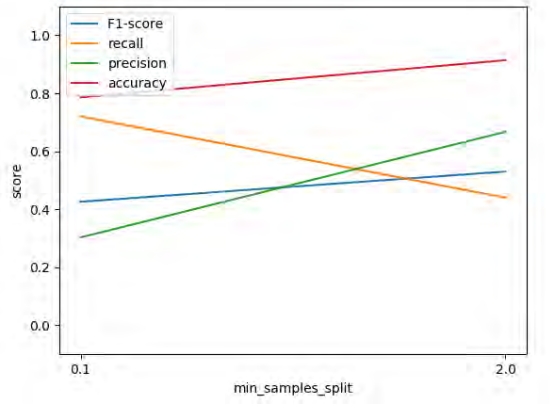


Figure 3.11: The effect of min_samples_split on the metrics for pumping station PSb

min_samples_leaf

In all cases 1.0 gives a better result than the 0.1, two examples are shown in Figure 3.12 and Figure 3.13. Similar as to min_samples_split it is slightly deceiving that the 0.1 is to the left of 1.0 while being larger. So it could very well be that more than 1 would be better, as long as it is less than about 10% of the data. Also a third value would have been very interesting.

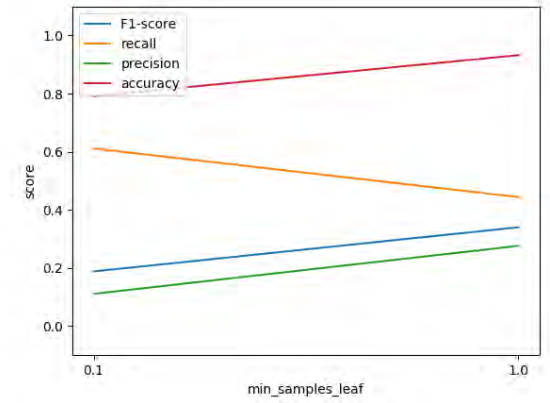


Figure 3.12: The effect of min_samples_leaf on the metrics for pumping station PSe

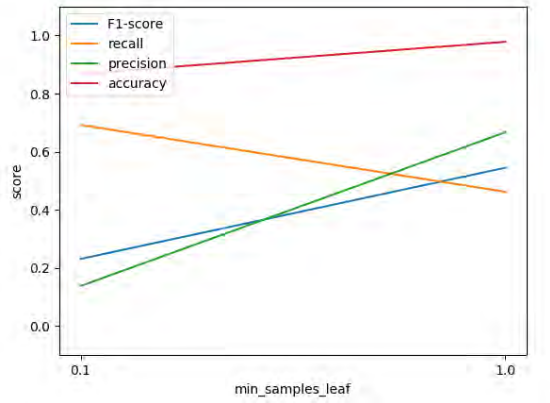


Figure 3.13: The effect of min_samples_leaf on the metrics for pumping station PSm

min_impurity_decrease

Only for seven pumping stations (PSc, PSd, PSf, PSg, PSh, PSi and PSk) did more than one value return a non-nan value, so only 7 pumping stations can be compared. Except for one pumping station (PSi - Figure 3.14) the recall is low at 0.0 and increases drastically around 0.25 to maintain more or less similar afterwards. This is complemented by precision and F1-score which start at their highest point on 0.0 and then drop in score to get similar results afterwards. With nearly all values accuracy seems to not be effected too much, however this one has a huge effect on accuracy. A more representative figure for the other 6 pumping stations is shown in Figure 3.15.

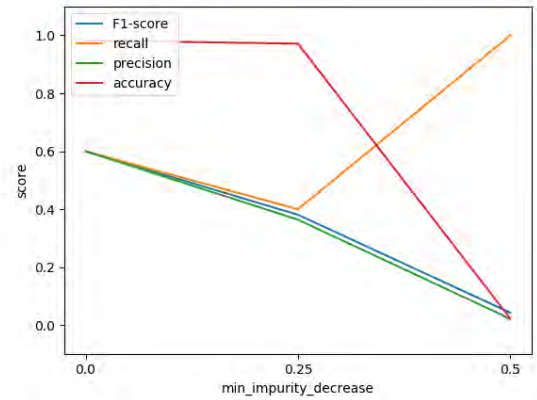


Figure 3.14: The effect of min_impurity_decrease on the metrics for pumping station PSi

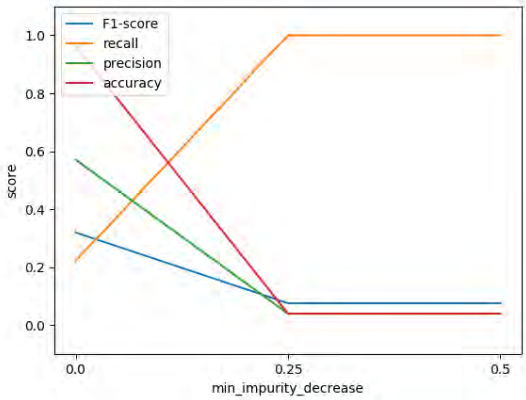


Figure 3.15: The effect of min_impurity_decrease on the metrics for pumping station PSk

bootstrap

Only 4 models resulted in figures (PSa, PSb, PSe and PSl). For those 4 models (except PSa where precision and recall are exactly alike and thus so is the F1-score - Figure 3.16), precision seems to prefer False, and recall seems to prefer True, resulting in a nearly even F1-score, an example is shown in Figure 3.17.

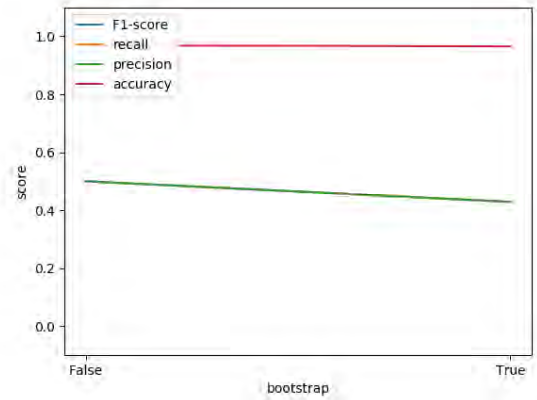


Figure 3.16: The effect of bootstrap on the metrics for pumping station PSa

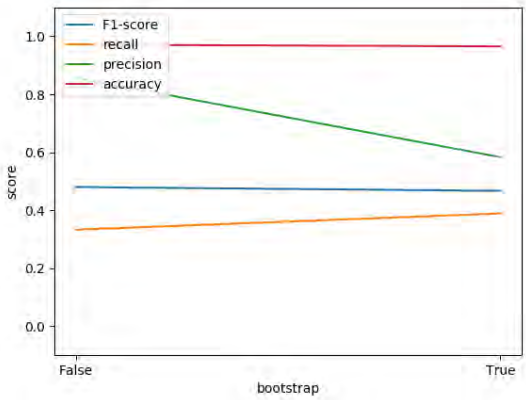


Figure 3.17: The effect of bootstrap on the metrics for pumping station PSl

Summary

So ,to summarize, the features `min_samples_split`, `min_samples_leaf` and `min_impurity_decrease` should be set not too big. In general entropy does the best work with regards to `criterion` and `n_estimators` and `max_features` should really be tuned every time separately. All the other features have (nearly) no effect, and can thus be disregarded.

3.2.1.2 XGB

Also in the XGB algorithm there turned out to be a parameter with no effect: `warm_start`. All other parameters at least had a little bit of effect on some pumping stations. This algorithm suffered the most of nan-values and with nearly every parameter there is at least one pumping station for which the values could therefore not be compared.

loss
In this case the pumping station PSi has nan-values. Furthermore this parameter has little to no effect on pumping stations PSh and PSj. On other pumping stations it does have effect, but no general preference, some prefer deviance (like Figure 3.18), and other prefer exponential (like Figure 3.19).

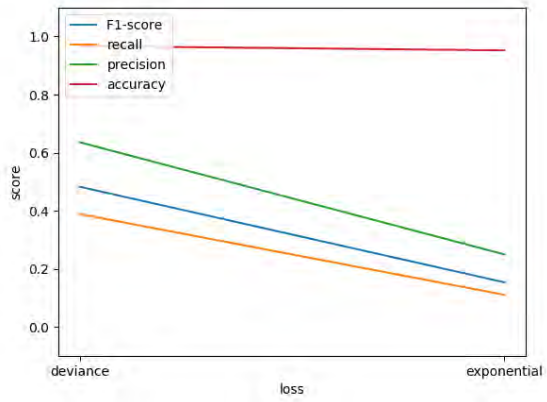


Figure 3.18: The effect of loss on the metrics for pumping station PSe

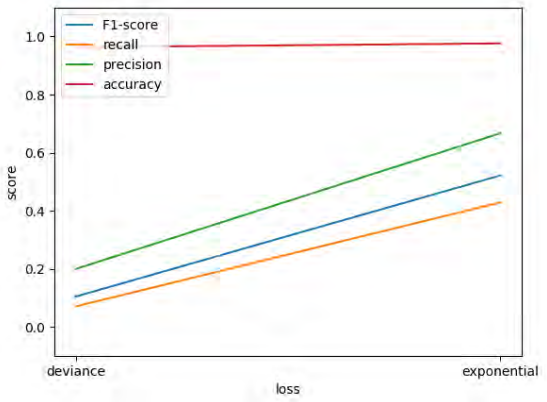


Figure 3.19: The effect of loss on the metrics for pumping station PSg

learning_rate
Three pumping stations (PSi - Figure 3.20, PSk and PSm) had nan-scores when the value was 0.01. With nearly all other pumping stations there is a sharp 'hook' at 0.1 with quite an effect towards 0.01, an example is shown in Figure 3.21

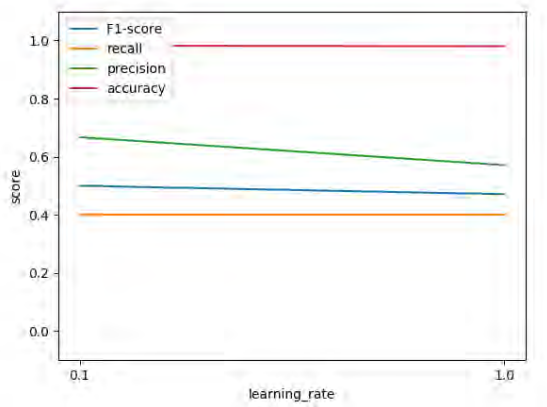


Figure 3.20: The effect of learning_rate on the metrics for pumping station PSi

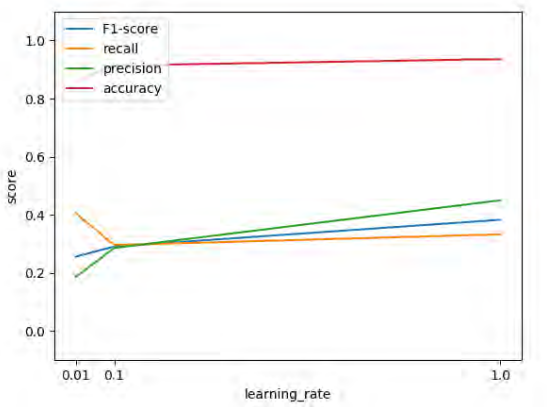


Figure 3.21: The effect of learning_rate on the metrics for pumping station PSj

n_estimators
Pumping station PSm had nan-values. Furthermore there is not really a comment 'preference'. In the case of pumping station PSa there is simply no effect. Three pumping stations (PSf, PSg and PSi - Figure 3.22) prefer 10 and the other eight pumping stations perform better at 100, an example shown in Figure 3.23.

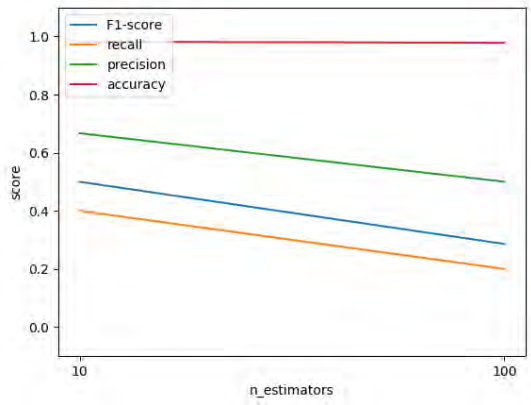


Figure 3.22: The effect of n_estimators on the metrics for pumping station PSi

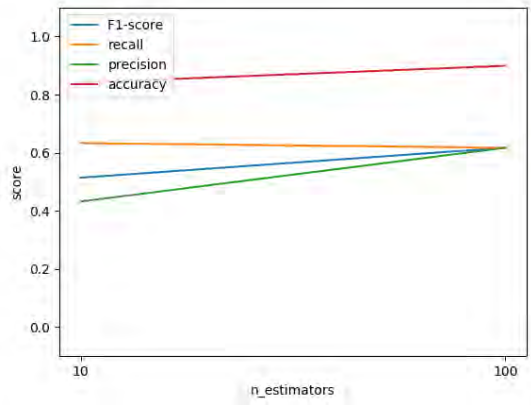


Figure 3.23: The effect of n_estimators on the metrics for pumping station PSc

max_depth:
Pumping station PSk had nan-values. With only two pumping stations preferring a max_depth 3 over 10 (PSe and PSf - Figure 3.24) it seems like in general 10 results in a better performance (Figure 3.25).

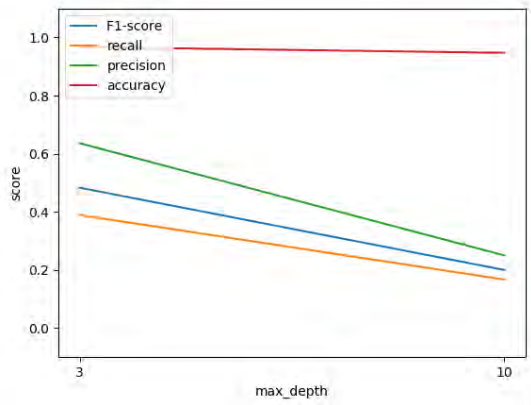


Figure 3.24: The effect of max_depth on the metrics for pumping station PSe

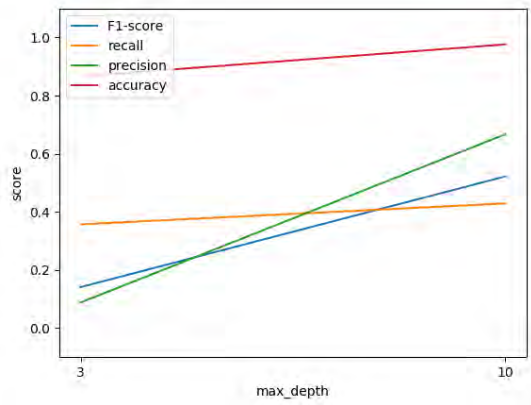


Figure 3.25: The effect of max_depth on the metrics for pumping station PSg

Interesting to note here is that both pumping stations which prefer 3 are from the same catchment (08).

criterion:
Both pumping station PSi and PSm returned nan-values. Except pumping station PSg (Fig-

ure 3.26) all pumping stations perform better with the friedman_mse criterion than with the mae criterion. Also interesting to note is that in general accuracy does not seem to be extremely influenced by this parameter, except with pumping station PSc (Figure 3.27).

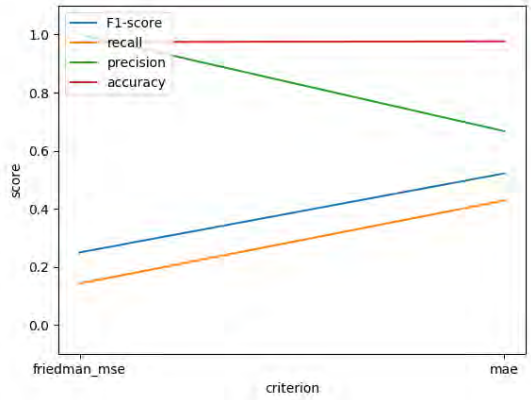


Figure 3.26: The effect of criterion on the metrics for pumping station PSg

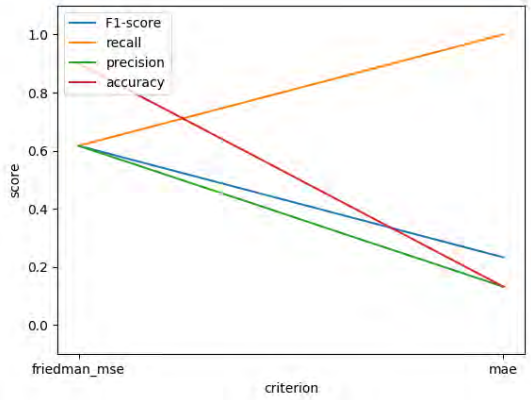


Figure 3.27: The effect of criterion on the metrics for pumping station PSc

min_samples_split

This parameter did not have any effect on seven pumping stations. There is only one pumping station (PSk - Figure 3.28) which prefers 2.0 over 0.01. The remaining five (PSa, PSd, PSg, PSi and PSm - Figure 3.29) all prefer 0.01 over 2.0. So in general little can be said, but it seems like it does not matter a lot. And if it matters then 2.0 is preferred over 0.01.

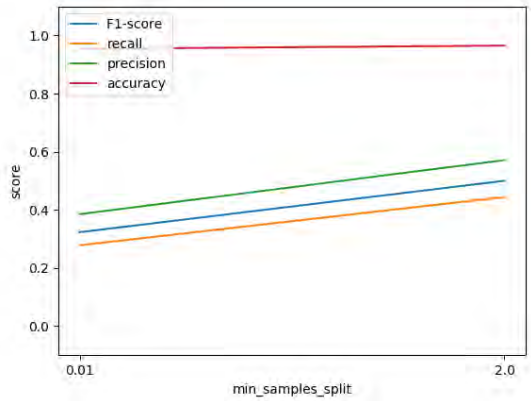


Figure 3.28: The effect of min_samples_split on the metrics for pumping station PSk

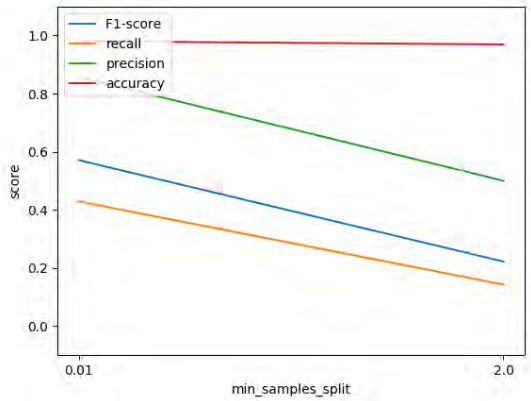


Figure 3.29: The effect of min_samples_split on the metrics for pumping station PSa

min_samples_leaf

Both pumping station PSi and PSm returned nan-values. With pumping station PSf this variable had no effect. And the remaining 10 pumping stations are perfectly divided: five pumping stations (PSa, PSd, PSg, PSh and PSk - Figure 3.30) perform better with 1.0 and the other five (PSb, PSc, PSe, PSj and PSl - Figure 3.31) perform better with 0.01.

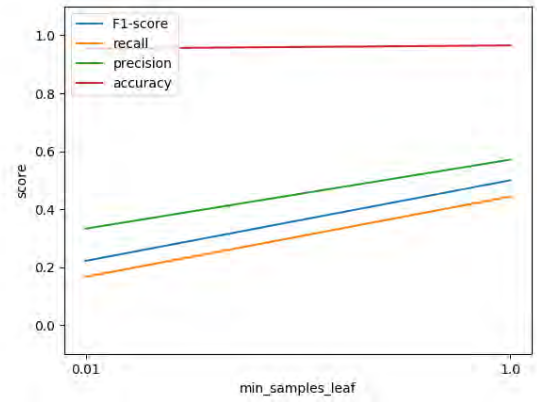


Figure 3.30: The effect of min_samples_leaf on the metrics for pumping station PSk

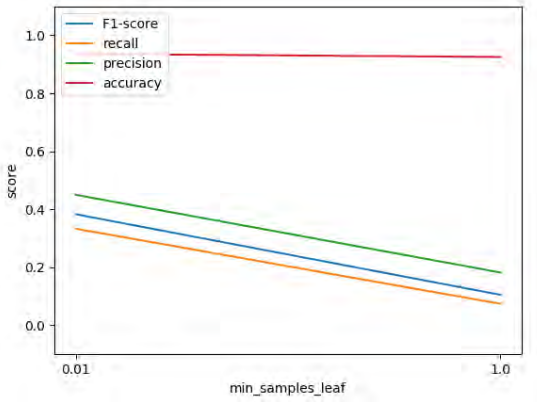


Figure 3.31: The effect of min_samples_leaf on the metrics for pumping station PSj

subsample

Pumping station PSg only has two values in the graph: 0.5 and 0.75, the 1.0 resulted in a nan-value. On which value performs better the opinions are divided. five pumping stations perform best with 0.5 (Figure 3.32). Three with 0.75 (Figure 3.33) and four with 1.0 (Figure 3.34). And then pumping station PSi performs best with both 0.75 and 1.0 (Figure 3.35).

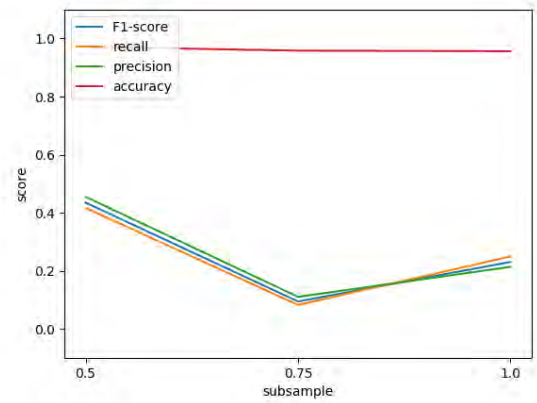


Figure 3.32: The effect of subsample on the metrics for pumping station PSf

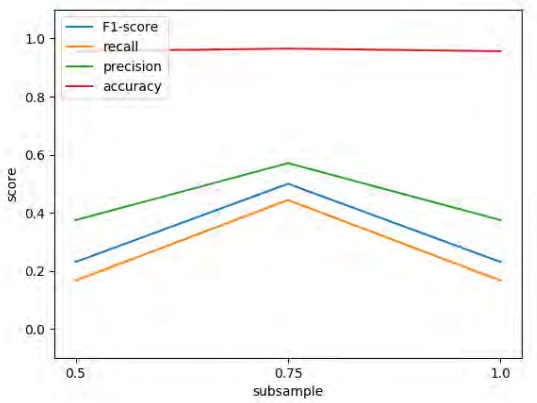


Figure 3.33: The effect of subsample on the metrics for pumping station PSk

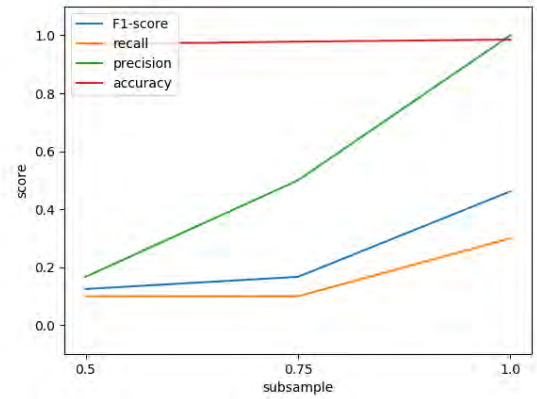


Figure 3.34: The effect of subsample on the metrics for pumping station PSm

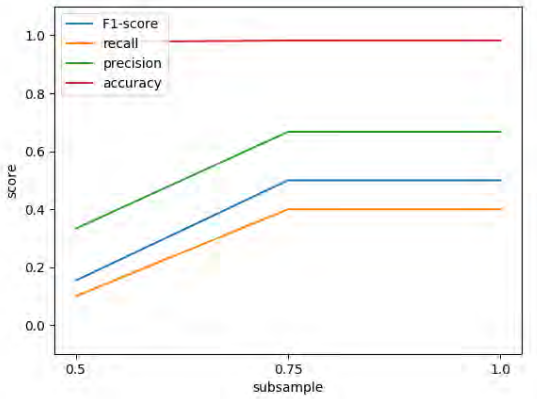


Figure 3.35: The effect of subsample on the metrics for pumping station PSi

max_features

Pumping station PSm returned nan-values. In general 'None' seems to outperform 'sqrt' as only three pumping stations perform better with the latter (Figure 3.36), and nine pumping stations perform better with 'None' (Figure 3.37).

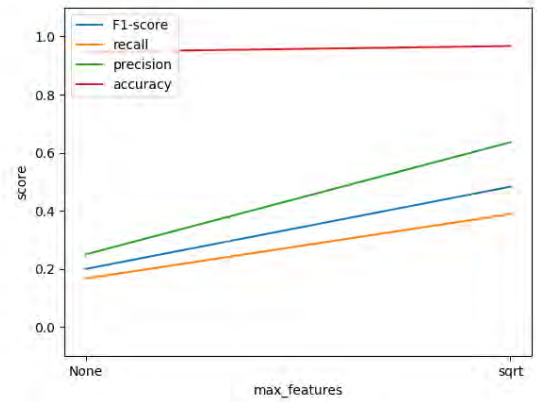


Figure 3.36: The effect of max_features on the metrics for pumping station PSe

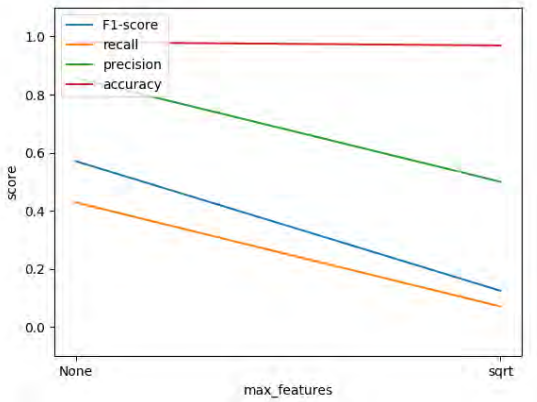


Figure 3.37: The effect of max_features on the metrics for pumping station PSa

min_impurity_decrease

Pumping station PSg returned nan-values. Generally speaking it can be said that 0.5 outperforms 0. Only three pumping stations have no improvement towards 0 (Figure 3.38). All nine other pumping stations perform better with 0 than with 0.5 (Figure 3.39).

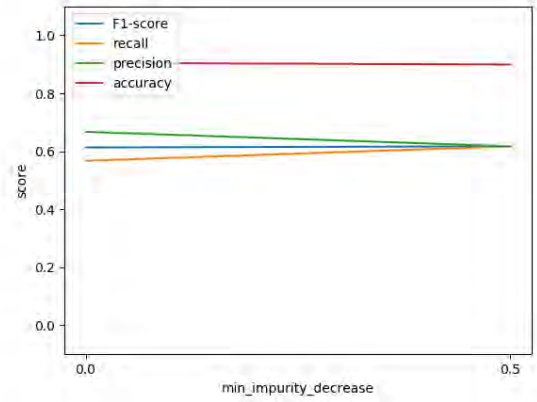


Figure 3.38: The effect of min_impurity_decrease on the metrics for pumping station PSc

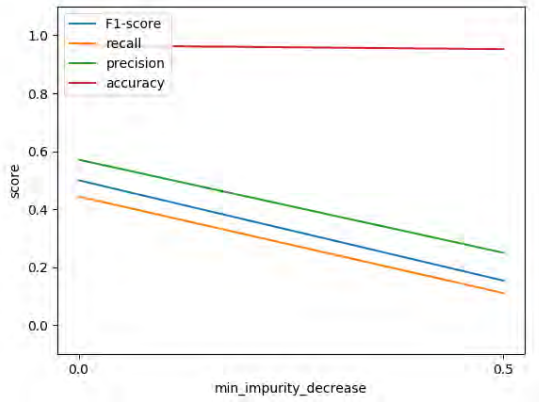


Figure 3.39: The effect of min_impurity_decrease on the metrics for pumping station PSk

Summary
These parameters in general differ a lot more than with the RF. For that reason it is advised to every time tune the parameters. Having said that, the variables `loss`, `learning_rate`, `n_estimators`, `min_samples_split` and `subsample` all do matter, but not one value outperforms the other options. So these variables will always have to be re-tuned. However `max_depth` seems to perform most of the time better with a depth of 10 over depth 3. It would be advised to try and find if values higher then 10 and between 3 and 10 maybe perform even better. Next `criterion` as the documentation already foretold, most of the time the option 'friedman_mse' performed best. In the case of `min_samples_split` and `min_impurity_decrease` there is usually not much effect, but if there is an effect it most of the time performs better with lower values. Regarding `max_features` it is generally best to choose 'None'.

3.2.1.3 SVM

With SVM all variables had at least some effect. Furthermore no nan-values were returned.

C
The 'penalty parameter', for some pumping stations there is quite some effect (Figure 3.40), whereas with other pumping stations there seems to be little effect (Figure 3.41). In any case there seems to be no clear tendency for one to perform better in all cases.

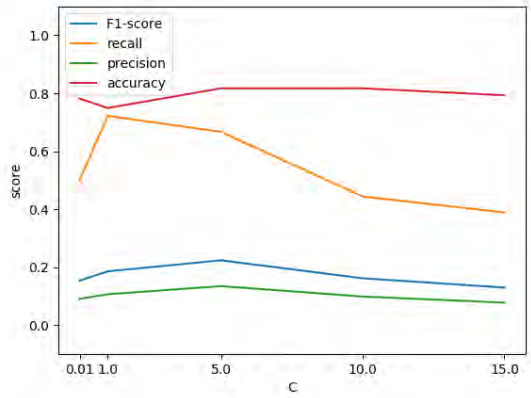


Figure 3.40: The effect of C on the metrics for pumping station PSe

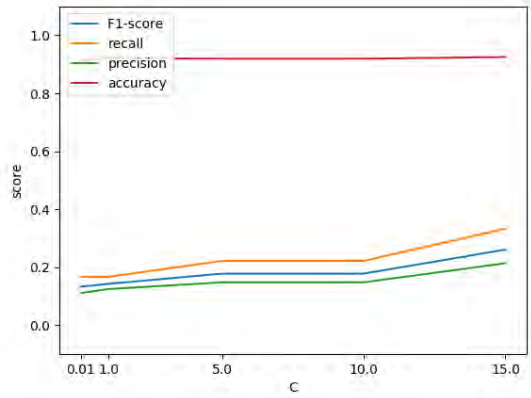


Figure 3.41: The effect of C on the metrics for pumping station PSl

coef0

Three pumping stations display no effect when this parameter is changed (PSc, PSf and PSh). The other pumping stations sometimes prefer higher coef0 (e.g. Figure 3.42) and sometimes prefer lower values (Figure 3.43).

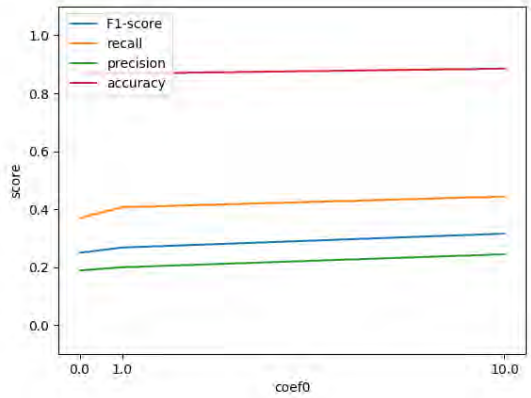


Figure 3.42: The effect of coef0 on the metrics for pumping station PSj

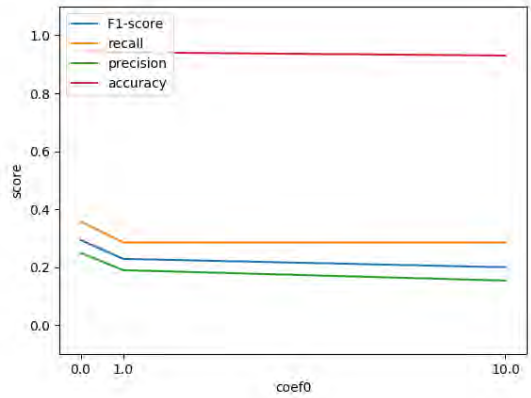


Figure 3.43: The effect of coef0 on the metrics for pumping station PSa

degree:

With exactly the same pumping stations[2] as with `coef0` there is no effect at all (PSc, PSf and PSh). In general there is very little that can be said, it seems to be all over the place, two examples are shown in Figure 3.44 and Figure 3.45.

[2] As degree is only used by the kernel 'poly' and coef0 only by 'poly' and 'sigmoid', it might have something to do with that

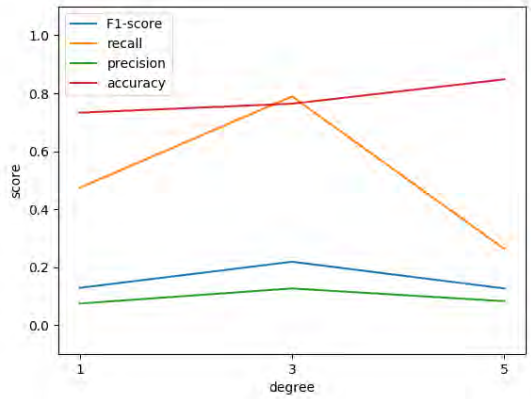


Figure 3.44: The effect of degree on the metrics for pumping station PSd

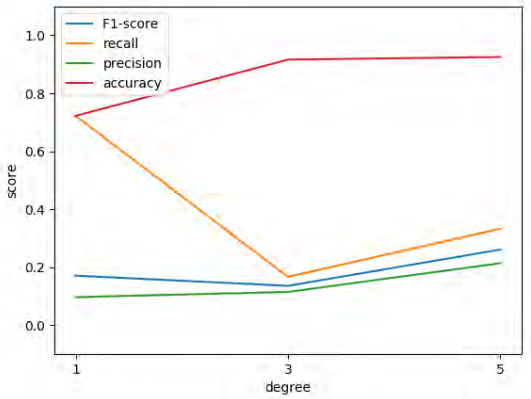


Figure 3.45: The effect of degree on the metrics for pumping station PSl

gamma:
Because more values have been tried for this variable, it offered the opportunity to present more interesting figures. And it did (Figure 3.46). Do note that due to the 'auto'-option the x-axis is not scaled as probably assumed. In general it seems like a lower gamma is better for recall (there are only three exceptions: PSe, PSh and PSk), see Figure 3.47. It also has quite an effect on accuracy, but less so on precision resulting in a lessened effect on the F1-score.

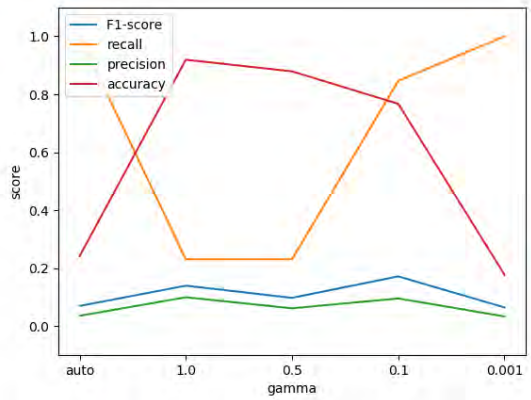


Figure 3.46: The effect of gamma on the metrics for pumping station PSm

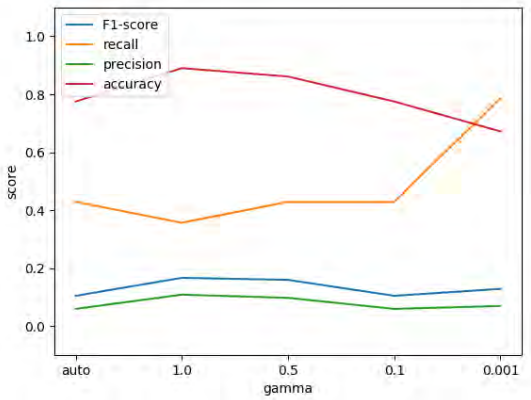


Figure 3.47: The effect of gamma on the metrics for pumping station PSg

kernel:
Just as with `gamma` this parameter results in some interesting figures. In general it seems like sigmoid is a good choice for recall, and bad for accuracy (Figure 3.49), with exceptions for pumping stations PSh, PSi and PSm (Figure 3.48). Also the kernel does not have as much effect on precision as on recall and accuracy resulting in not too much effect on F1-score either.

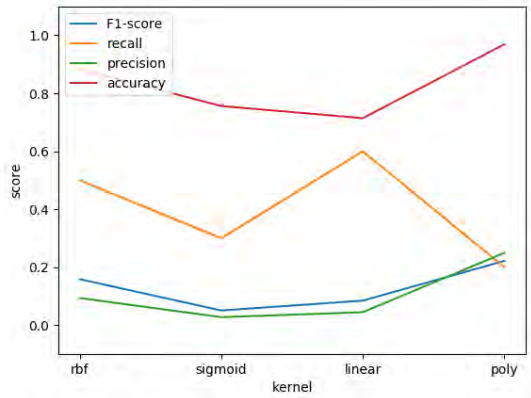


Figure 3.48: The effect of kernel on the metrics for pumping station PSi

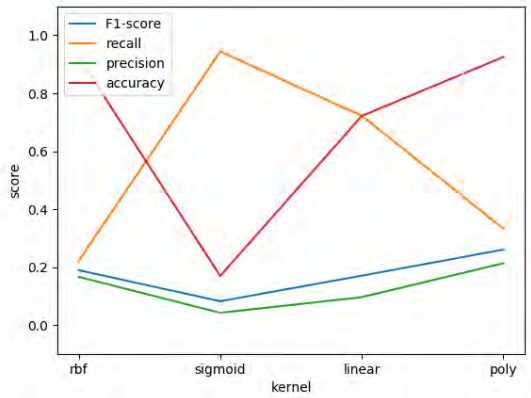


Figure 3.49: The effect of kernel on the metrics for pumping station PSl

tol:
This variable has nearly no effect, only with pumping station PSi there is a very small effect, this is shown in Figure 3.50.

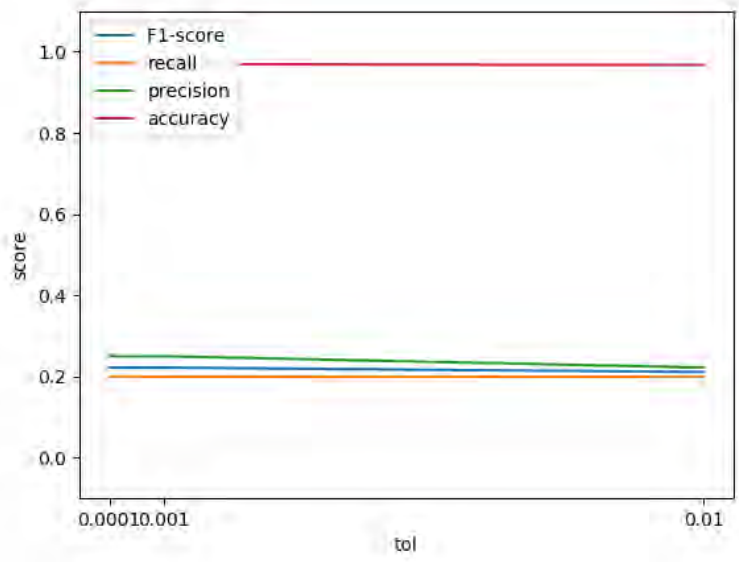


Figure 3.50: The effect of tol on the metrics for pumping station PSi

Summary
`Tol` seems to have very little effect in general. In the case of SVM many variables simply will have to be tried every time because it depends on the pumping station what works best, this is the case with `C`, `coef0` and `degree`. This actually also counts for `gamma` and `kernel`, however a lower `gamma` seems to be better for the recall. Similarly sigmoid `kernel` seems to be better for recall. If F1-score is looked at the difference is not that big anymore, but might move towards rbf or poly.

3.2.1.4 LSTM

The code did seem to work. However with the data from the first pumping station (PSa) with parameter options 144 a memory error occurred after which windows had apparently been somehow corrupted as the computer would have a blue screen, restart back into the blue screen, resulting in an infinite loop. Windows had to be re-installed. Due to time pressure and this last result, this model has not been run any further. And thus no real conclusions can be deduced. It would be interesting to see as future research how temporal models perform.

The first 143 parameter settings with regards to the LSTM model on pumping station PSa were not as promising as the other models, with a maximum F1-score of 0.050. In total 648 settings should have been tried, and only 143 were tried. So actually this F1-score says nothing yet. Comparing these results to the lowest 100 or so results of the RF model on pumping station PSa, those were only 0.060. So actually it gives very little indication on how good or bad the LSTM might have performed.

3.2.1.5 Before oversampling minority

For all three algorithms there are models which return the remarkable result that -despite the skewedness of the data- all instances are labeled 'defect'. The first time all the models were run was without the oversampling of the minority (as explained in the beginning of section 2.2). Since this was an unexpected result some effort was put into finding what could have caused this.

For all three algorithms one model was elected with this weird result. And one by one all parameters were looked at. This was done in a similar method as described in subsection 3.2.1: in this case however the parameter was not altered, but removed (causing if a default was in place, the default would be used).

For each algorithm a few parameters were found which could 'undo' this weird result. For this reason it has been concluded that this was caused by a specific combination of parameter choices. And was not further researched.

Following for each algorithm the model which was looked at and the parameter changes, which caused a more expected result.

name	original value	value with more expected result
n_estimators	5	
criterion	'gini'	
max_features	'auto'	
max_depth	'None'	
min_samples_split	2	
min_samples_leaf	1	
min_weight_fraction_leaf	0	
max_leaf_nodes	None	
min_impurity_decrease	0.25	0.0
bootstrap	True	
oob_score	True	
n_jobs	1	
random_state	0	None
verbose	False	
warm_start	True	
class_weight	'balanced'	None

Table 3.4: Results of looking into one RF model which predicted all instances as 'defect' on pumping station PSa

name	original value	value with more expected result
loss	'deviance'	
learning_rate	1	0.1
n_estimators	100	
max_depth	3	
criterion	'friedman_mse'	
min_samples_split	0.01	2
min_samples_leaf	1	
min_weight_fraction_leaf	0	
subsample	0.50	1
max_features	None	
max_leaf_nodes	None	
min_impurity_decrease	0.5	0.0
init	None	
verbose	0	
warm_start	True	
random_state	None	
presort	'auto'	

Table 3.5: Results of looking into one XGB model which predicted all instances as 'defect' on pumping station PSh

name	original value	value with more expected result
C	5.0	
cache_size	200MB	
class_weight	'balanced'	None
coef0	0	
decision_function_shape	'ovo'	
degree	5	3
gamma	'auto'	
kernel	'poly'	'rbf'
max_iter	-1	
probability	False	
random_state	None	
shrinking	True	
tol	0.01	
verbose	False	

Table 3.6: Results of looking into one SVM model which predicted all instances as 'defect' on pumping station PSe

3.3 Prediction

In Figure 3.51 the maximal F1-score reached for each pumping station is shown. The exact values can be found in Table 3.7. The SVM performs worst for all pumping stations. In general the XGB model performs best of the three models, only for two data sources does the RF outperform the XGB. Also pumping station PSc seems to be the easiest to predict reliably. Whereas pumping station PSj seems to present a more difficult case.

	F1-score		
pumping station	max RF	max SVM	max XGB
PSa	0.500	0.294	0.571
PSb	0.530	0.373	0.608
PSc	0.612	0.545	0.617
PSd	0.467	0.219	0.485
PSe	0.340	0.224	0.483
PSf	0.250	0.187	0.435
PSg	0.500	0.167	0.522
PSh	0.519	0.186	0.552
PSi	0.600	0.222	0.500
PSj	0.350	0.316	0.383
PSk	0.320	0.314	0.500
PSl	0.480	0.261	0.600
PSm	0.545	0.172	0.462

Table 3.7: For each pumping station comparing the best F1-scores for all three models. To give a first indication, the threshold is set on the lower bound found in subsection 2.2.5 (0.2726). Green means 'better than the threshold' and red means 'worse than threshold'. For visualisation see Figure 3.51.

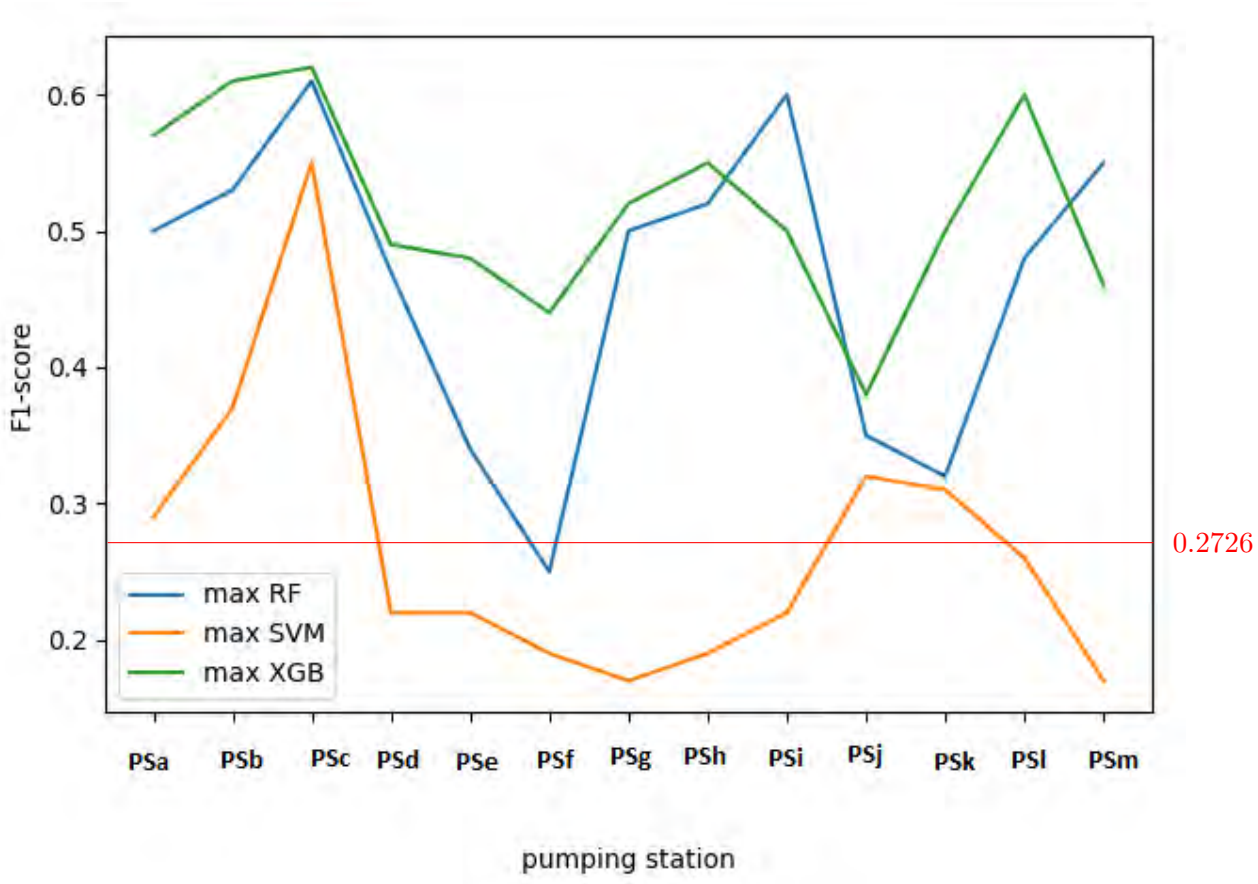
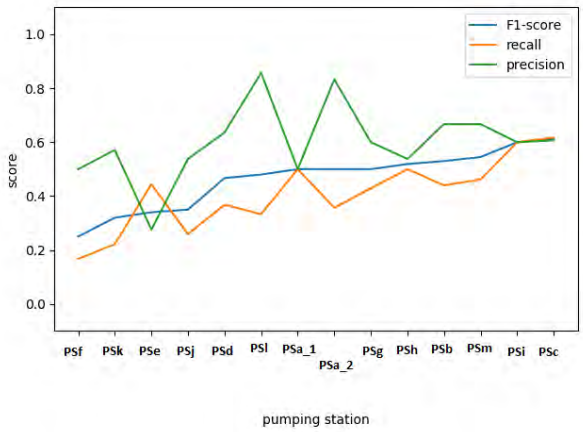


Figure 3.51: The maximal F1-Score for each pumping station by each method. This is the visualization of Table 3.7. The red line is the threshold.

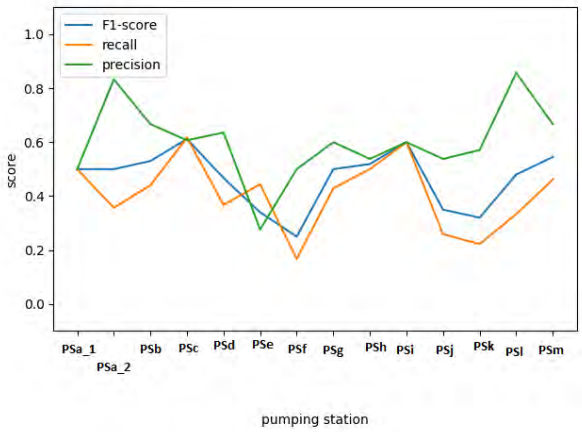
		best model scores		
pumping station	**algorithm**	**precision**	**recall**	**F1-score**
PSa	RF	0.500	0.500	0.500
PSa	RF	0.833	0.357	0.500
PSa	XGB	0.857	0.429	0.571
PSa	SVM	0.250	0.357	0.294
PSb	RF	0.667	0.440	0.530
PSb	XGB	0.596	0.620	0.608
PSb	SVM	0.252	0.720	0.373
PSc	RF	0.607	0.617	0.612
PSc	XGB	0.617	0.617	0.617
PSc	SVM	0.447	0.700	0.545
PSd	RF	0.636	0.368	0.467
PSd	XGB	0.571	0.421	0.485
PSd	SVM	0.127	0.789	0.219
PSe	RF	0.276	0.444	0.340
PSe	XGB	0.636	0.389	0.483
PSe	SVM	0.135	0.667	0.224
PSf	RF	0.500	0.167	0.250
PSf	XGB	0.455	0.417	0.435
PSf	SVM	0.150	0.250	0.187
PSg	RF	0.600	0.429	0.500
PSg	XGB	0.667	0.429	0.522
PSg	SVM	0.109	0.357	0.167
PSh	RF	0.538	0.500	0.519
PSh	XGB	0.533	0.571	0.552
PSh	SVM	0.108	0.643	0.186
PSi	RF	0.600	0.600	0.600
PSi	XGB	0.667	0.400	0.500
PSi	SVM	0.250	0.200	0.222
PSj	RF	0.538	0.259	0.350
PSj	XGB	0.450	0.333	0.383
PSj	SVM	0.245	0.444	0.316
PSk	RF	0.571	0.222	0.320
PSk	XGB	0.571	0.444	0.500
PSk	SVM	0.242	0.444	0.314
PSl	RF	0.857	0.333	0.480
PSl	XGB	0.750	0.500	0.600
PSl	SVM	0.214	0.333	0.261
PSm	RF	0.667	0.462	0.545
PSm	XGB	1.000	0.300	0.462
PSm	SVM	0.096	0.846	0.172

Table 3.8: For each pumping station and each algorithm the best F1-score with the belonging precision and recall. A visual representation is given in Figure 3.52, Figure 3.53 and Figure 3.54.

Figure 3.52: The precision, recall and F1-score of the best models from the RF algorithm according to the F1-score [3]

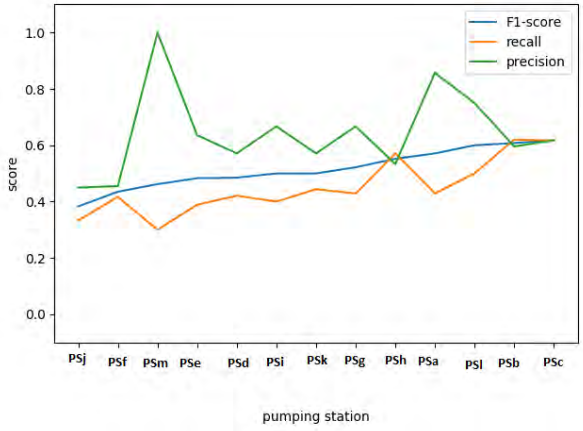


a: Ordered by increasing F1-score

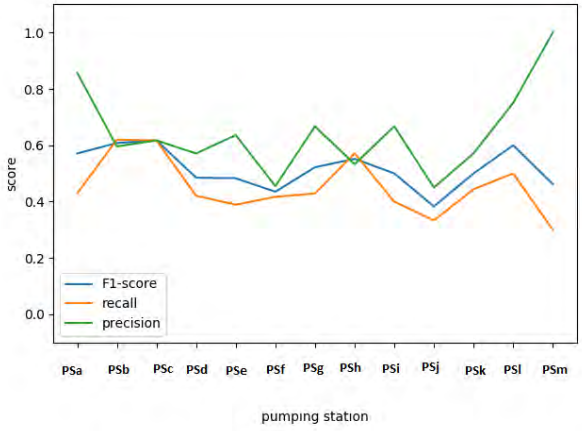


b: Ordered by pumping station

Figure 3.53: The precision, recall and F1-score of the best models from the XGB algorithm according to the F1-score



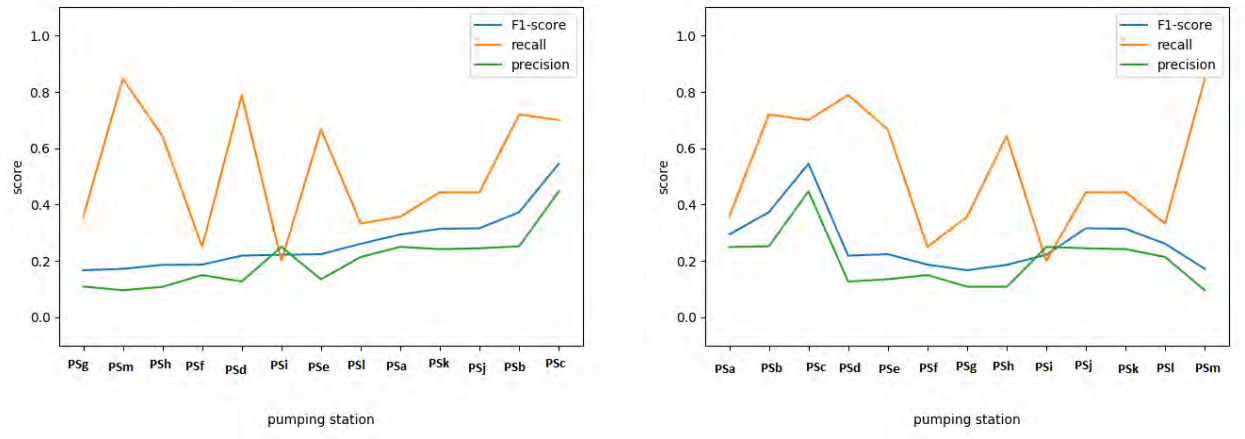
a: Ordered by increasing F1-score



b: Ordered by pumping station

[3] In the RF algorithm two combinations of a precision and recall score resulted in exactly the same F1-score at pumping station PSa, which also happened to be the best. For this reason there is a PSa_1 and PSa_2, so both these combinations can be shown.

Figure 3.54: The precision, recall and F1-score of the best models from the SVM algorithm according to the F1-score



a: Ordered by increasing F1-score

b: Ordered by pumping station

How the recall, precision and F1-score move together is visually shown in Figure 3.52, Figure 3.53 and Figure 3.54 respectively for the RF, XGB and SVM algorithm. Looking at those figures it seems like RF and XGB favor precision over recall, and SVM seems to favor recall over precision.

3.3.1 Insights

F1-score & feature importance
What features were deemed top-10 important seems to have no direct link with the F1-score.

F1-score & empty variables
The F1-score and empty variables, be it in of all the variables, only the important variables or the top 10 most important variables, do not seem to have anything to do with each other. However what is interesting is to see is that both the important features and the top 10-features seem to be bound by some sort of demand on the % of missing values. As shown in Figure 3.55 the percentage of empty variables in total differs a lot. But looking at the weighted average and the average of both the important and the top-10 features, none of them pass the 12.2% of missing values on average. Looking further into it the feature importance did not only look at the missing values, some of the top-10 have more missing values than the minimum % of missing values for that pumping station. However it seems safe to say that lower % of missing values has a positive influence on being elected as an important feature.

This 'demand' from the data seems to be even more strict than what was demanded during the data preparation, since there all features with more than 25% missing values have been removed.

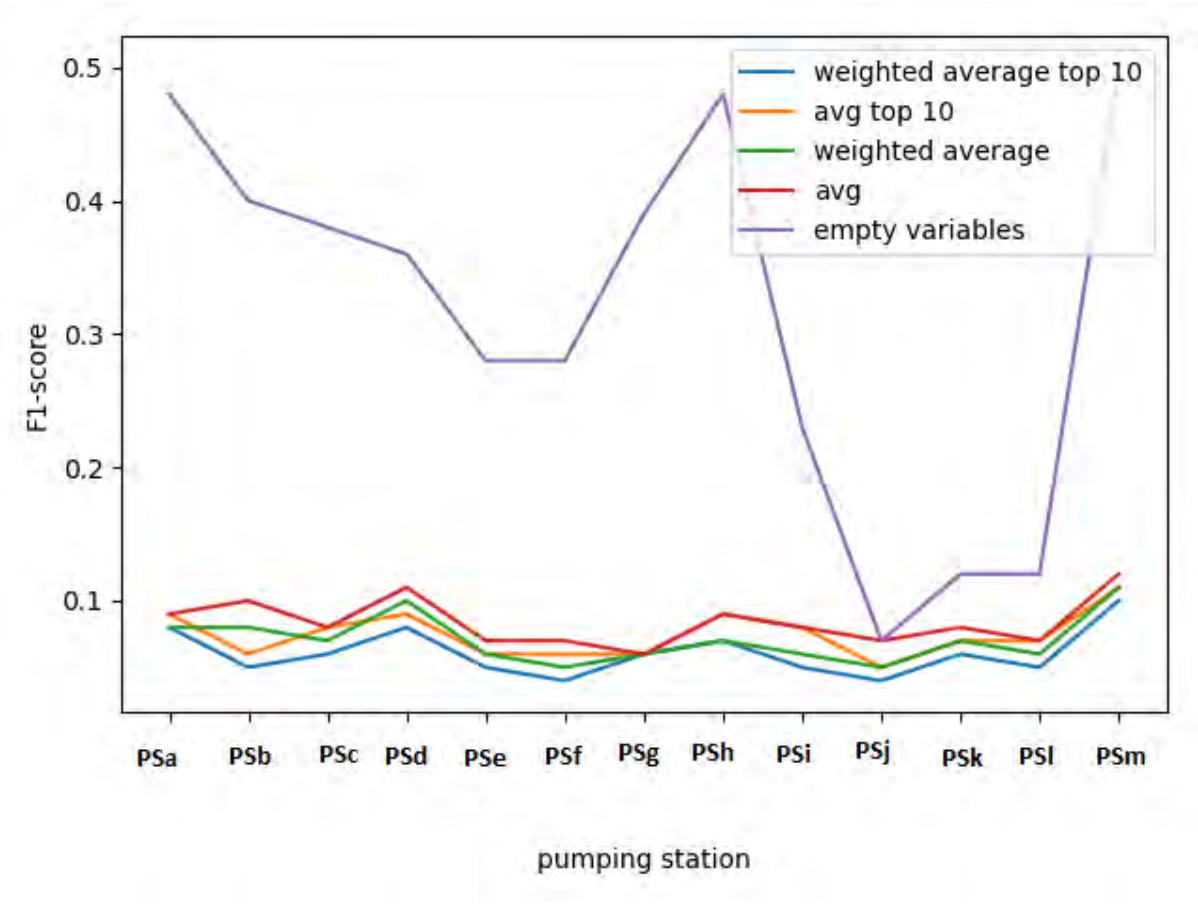


Figure 3.55: The total percentage of missing values compared to the percentage of missing values of the used features and the top-10 most important features

F1-score & Skewedness of the data

In Figure 3.56 the pumping stations are ordered from highest % minority to lowest. This percentage is shown in the dark red line. The other three lines are the best F1-scores of each algorithm. This plot seems to indicate that especially the SVM-algorithm suffers from skewed data.

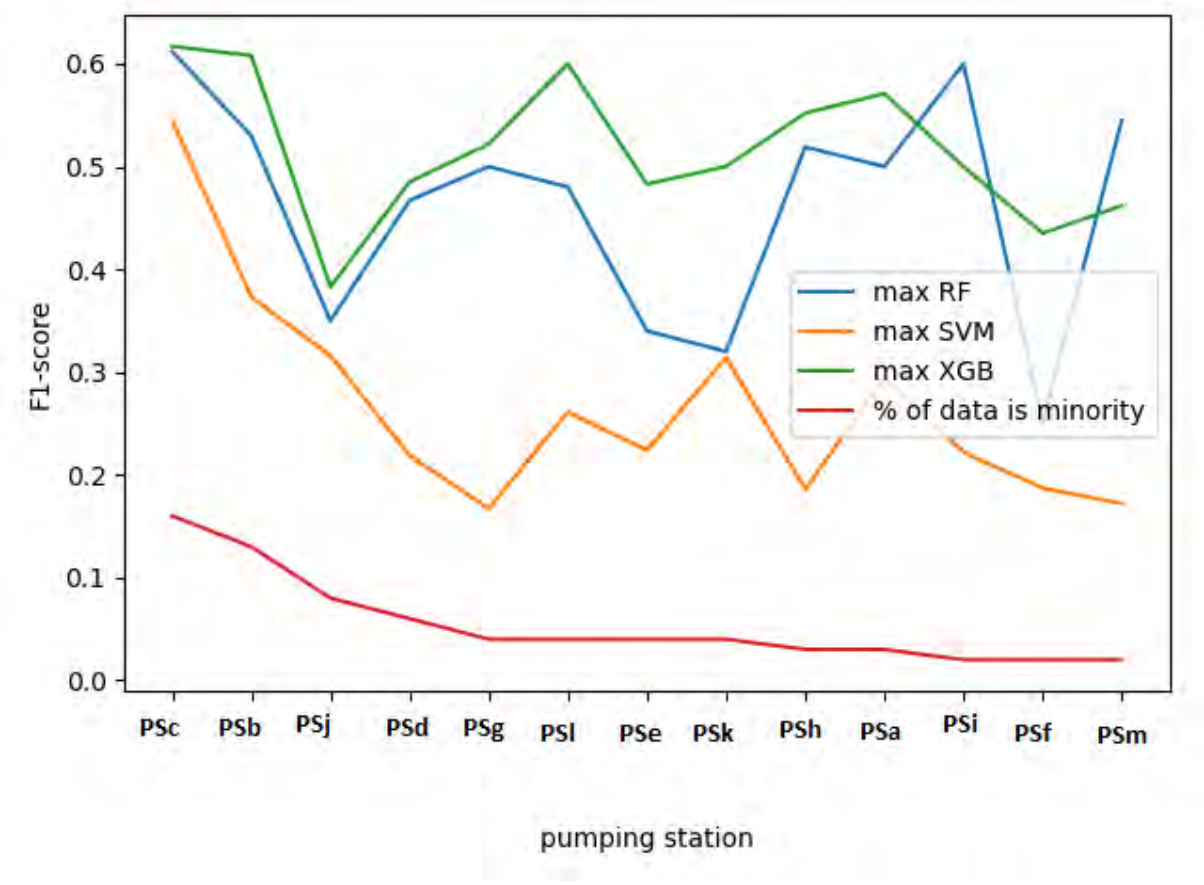


Figure 3.56: The F1-scores of the different algorithms plotted together with the % of data which is a minority

It also gives a possible explanation why pumping station PSc seems so much easier to predict than other pumping stations. As it is the pumping station with the least skewed data.

Chapter 4

Discussion

4.1 Usefulness of the results

The usefulness of these results depends greatly on the necessary certainty. None of the models catch all defects without some FP. But these models are definitely better than simply tossing a coin. Any pumping station can be used for this example, but it has been chosen to work with pumping station PSj because with that particular pumping station all algorithms had some trouble. The XGB algorithm did perform best of the three. So with XGB 33.3% of the defects will be called a defect, and when an instance is being called a defect, in 45% of the cases it will actually turn out to be true. Comparing this to the number of instances which are actually defect (7.95%), this still results in an increase in certainty about whether the situation is heading for a defect or not. Such an increase in the pumping station which performed most poorly is promising.

Depending on whether recall or precision is valued more, a different algorithm seems in place. SVM seems to focus on recall, with seven pumping stations the maximum F1-score of the model yields the highest precision of the three algorithms for that particular pumping station. However it has such a poor precision that the F1-score is generally lower. With both RF and XGB the precision is better than the recall. XGB and RF seem to be about as good at precision. However go with XGB when precision and recall are important, because XGB in general has the highest F1-score.

4.2 Validity of results

Assumptions
Many assumptions have been made throughout this research. Here an effort is being made to give an overview of these assumptions.

To start with, the problem statement already introduces a first assumption: the defects are predictable 3 days in advance. It could very well be that a different amount of time would produce better results. However for reasons stated in the problem statement it has been chosen to continue with the 3 days assumption.

Another introduced assumption is the data aggregation per day. As a lot of fluctuation is going on in the data, it could very well be that some important information is aggregated out of the data by choosing this long time frame. An option which Marly van der Mei's [13] research seems to back up, by getting better results with the RF algorithm but aggregating the data per 10

minutes.

A third assumption hides in the feature engineering. By choosing for a linear scale for both the weekdays and the seasons, the 'circularity' of the data is completely lost. To the model Sunday is as far away from Monday as possible, however in reality it is just as close to Monday as Tuesday is. This assumption can cause that the model has a harder time figuring out if something only happens on Sunday or Monday. This seems illogic because they are so far away from each other. The assumption has been made due to time constraints.

The assumption has been made that the 'outlier values' of the level measurements (red box in Figure 2.12) could better not be altered as they might present information. This was done next to also creating the variable days_since_emptying, which arguably also holds this information. It would be interesting to see what happens if those values are replaced by for instance the predetermined 'switch off' level [1].

As choices had to be made and not 'all' model settings could be tried, the choice of which settings to try automatically form some assumptions. It would be great to find what the results of the models would be if more options could have been tried.

The choice to fill all empty values with one predetermined value includes some assumptions to the model. If that value happens to be telling, this choice will definitely make it less telling now. An assumption had to be made, another option could have been to perform a random draw from the other values aiming to maintaining the distribution the same way it currently is. Or -if more time could be invested- predict the missing value and use those.

Another assumption which has been made is that when more than $^1/_4$ of the entries of a certain variable are empty, the variable was no longer taken into consideration. This assumption could potentially throw away valuable information. However as Figure 3.55 shows it seems like the models themselves prefer an even lower threshold. For this reason this assumption probably did not have that much effect.

An assumption with probably a huge impact is which type of defects are looked at. On this database three researches have been done by now. The one by Arjen Kruit [12], Marly van der Mei [13] and this research itself. All three have chosen to -partially- look at different defects. Making it in essence somewhat difficult to compare them.

On the matter of defects another possible important assumption has been made: when a mechanic is present, no defects can occur.

Just like choosing which type of defects to look at, it is quite important which pumping stations are looked at. 13 stations have been chosen to look at, based on their defect-rates and a minimum demand on the data. However questions may rise if the results for the chosen pumping stations are therefore transferable to other pumping stations with either fewer defects and/or less data.

And a final assumption is that the feature importance has been done before splitting the test and the training set. This -in fact- causes an unfair advantage: a bit of the information held by the data in the test set is being used to predict those same values.

Improvements

[1] The water level at which the pumping stations stop pumping again

Next to the assumptions some improvements could be made to this research. Next to the already mentioned ones, possible improvements that have been thought of will be listed here.

To start with, no cross-fold validation was applied. This was done due to the combination of a time-constraint and oversampling the minority. To be able to use all the data -as cross-fold validation causes- would definitely improve the sturdiness of this research.

It would be great to see what the LSTM-algorithm would have produced as results. Especially since it is a completely different look on things. It would definitely improve the research had this algorithm been properly implemented and run.

Due to time constraints not as much time and effort could go to feature engineering as the researchers would have liked to see. As this is often a vital part of ML, it is very well possible that the results could be further improved by adding more possible features. One of the features which could -for instance- be considered is whether a pumping station is situated in an industrial or living area.

4.3 How transferable are the results?

With the difference in availability of the variables per catchment it is difficult if not impossible to draw any conclusions on whether a method could be transferable. Luckily however there are two sets of pumping stations which are in the same catchment, the first couple being PSe and PSf and the second couple being PSk and PSl. Those couples have the same variables at their disposal. With the same empty values. The only difference between each is the distance between the pumping station and the respective measurement sensor. But even while looking at the results of those couples, it does not seem like one can give any sort of indication on how well the algorithms will perform on the other.

Without more information on the pumping station (for instance type, brand, age) it is difficult to make any real statements here. But with the data at hand, this research seems to indicate that the results are not transferable.

Marly van der Meij's research [13] managed to get the following confusion matrix with the RF algorithm on pumping station PSe:

		predicted	
		0	1
real values	0	27673	2
	1	22	275

Resulting in a recall of 0.926, precision of 0.993 and F1-score of 0.958. A lot better than the results this research manage to find. In her research she made many different assumptions than this research did. For instance aggregating her data by 10 minutes instead of by day (also shown by many more instances in her confusion matrix).

Chapter 5

Conclusion & Recommendations

5.1 Conclusion

So as far as the main goal went, the results indicate that it is possible to predict a defect in a sewer system within the coming three days given the measurements in the system using Machine Learning. Note that these results should only be seen as a show of the possibility with ML. Before anything can be used in real-life it would be strongly advised to have another research look for the best model and optimize that one for a more specified task at hands.

Looking at the first secondary goal: of the three algorithms/techniques tested in this research it seems like XGB is best suited for this goal. But it could very well be that another method would perform even better.

The final secondary goal: expert advice and data analyses give different indications on what the most important characteristics are to predict a possible failure of a sewer system. To start with the data analyses, this could only look at the data which was presented. It indicates that the time spent in defect the past 3 days is an important feature, together with the time since the last emptying of the pumping station. Secondly the level measurements seem to harbor more information in this respect than the flow measurements. Now looking at what the experts told us, energy consumption by the pumping station should have been a very important feature (but was not present in the database), also how 'clean' the water is, should be an important factor. The database did include turbidity measurements, but there were not enough of those to be able to include them in this research. Therefore this will be a recommendation towards the municipality of Utrecht, to put an effort into getting the energy consumption of the pumping stations into the database as well as getting more (and more reliable) turbidity measurements.

5.2 Recommendations

The municipality of Utrecht could consider changing the 'goal' of their database. As we understood it, currently the goal is to collect data whenever possible to see what can be done with it later on [11]. However now three studies have been done on the data and questions have been formed on what the municipality would like to get from this database. This gives more information on what type of data are needed or could be useful and the reliability with which these data are collected. If the goal is to predict defects with this data, it is recommended to get the energy consumption of the pumping stations into the database as well as to get more turbidity measurements. Also more continuity could improve the results. Currently there are 'gaps' in the data of several days (sometimes over two weeks at a time). Considering that some measurements are done every minute, this forms a huge amount of missing data.

In the very first month of this research, in the process of getting to know the data, all the defects of all pumping stations had been combined in a single section of the database. This was a relatively simple section as it would only look at the measurements directly at the pumping station. With only a Decision Tree the following confusion matrix had already been achieved:

		predicted	
		0	1
real values	0	4508	133
	1	277	204

reaching a recall of 0.20, a precision of 0.35 and an F1-score of 0.26. No feature engineering, and for all the missing values simply the value '0' had been filled out. As a starting point this seems quite promising, and for future research we suggest walking this path. Quite possible with a more limited set of defaults, some feature engineering and more advanced handling of missing data, this might prove very useful.

Instead of focusing on 'most priority' defects, it would be interesting to see if any of the defect types maybe have a higher correlation with some data, and/or are better predictable. This research could very much help understand the data better. And maybe the bigger defects can be led back to a combination of smaller defects.

An interesting line of research would be to predict the level measurements and flow measurements and fill the missing data with the predicted values. This could have a very different effect than the chosen form in this research of simply interpolating the missing data.

This research tried to work with a temporal model (LSTM), but sadly no results could be drawn from this in the end. As future research it could be very interesting to see what the results would be. And whether looking at the data as a time-series has a positive effect on the results.

As stated before, not all possible engineered features have been taken into account. Other possibilities that passed by, but could not be implemented were, to start with the groundwater levels surrounding the pumping station. Tourist information on the number of people visiting Utrecht. This might be interesting because experts expect that more litter means a higher chance on defects, and more people might mean more litter.

Interesting features in the case of combining multiple pumping stations into one model might be whether the location is near to a shopping mall or restaurants, or for instance closer to a park. Probably within one city it will differ a lot, but if looked at a wider range the % of calcium in the water might be telling. Similarly whether the majority of the ground is sand or clay.

In the case of choosing a smaller aggregation time than a whole day, interesting features might include whether the time is during office hours or not.

Peroration

It was interesting to do this ML research in a company with very little internal knowledge on ML. The first and final mentors did know about ML. But the middle two have no knowledge on ML. They had more domain knowledge. In ML, and in general when trying to get insights from big data, it is always a challenge to maintain an useful balance between looking at the data and implementing domain knowledge. It is a somewhat treacherous path because as researcher you hope to find insights from the data which the domain experts do not know yet (and might even expect to not be there at all). However you do not what to go on a wild goose chase either. Looking at this research we feel we went astray from this path. In this case we let the knowledge of the domain lead us away from promising results data-wise. As we mentioned we had -at the start- some very promising results looking at all the pumping stations together. But according to the domain experts this was not possible with this method and we stopped looking down that path.

During the process of the research we have learned a lot. To start with the effect a mentor has on the process. During the 7 months of research, we had at the company -due to different circumstances- 4 different mentors. Each with their own expertise, experience and method of mentoring. It takes quite some communication to bring a new mentor up to speed on where you are currently in your research and why this is the chosen path. Also a new mentor always brings in a new point of view, which is very valuable, but the later in the process this happens the more difficult it is to still process it in the research.

Another aspect we got taught quite painfully is, as one of my mentors said: 'The effort needed for a part of the research should be in relation to what it contributes to the research'[1]. Looking back at the LSTM we feel like we definitely lost this out of sight. It was a great idea to look into. But as soon as we noticed that it took so much longer to get *one* model working than it took the other *three* combined, we should have stopped and have a really hard thought about it.

[1] Freely translated from Dutch

Bibliography

[1] Jaap de Rue & Marco Versluys. Asset management, powerpoint presentation at witteveen+bos. Jan-2018.

[2] Azima|DLI. Justifying predictive maintenance, (accessed: 7-5-2018). https://azimadli.zendesk.com/hc/en-us/articles/202870220-Justifying-Predictive-Maintenance, summary of the Technical paper, 2009.

[3] Adriaan van Hooijdonk. Assetmanagement leidt tot meer doelmatigheid rioleringsbeheer. http://edepot.wur.nl/396307,Wageningen University & Research.

[4] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44(1):206–227, 2000.

[5] RIONED. Ontstaan van het riool, (accessed: 30-4-2018). https://www.riool.info/ontstaan-van-het-riool.

[6] Wikipedia. History of water supply and sanitation, (accessed: 30-4-2018). https://en.wikipedia.org/wiki/History_of_water_supply_and_sanitation.

[7] Frank W. Geels and René Kemp. Dynamics in socio-technical systems: Typology of change processes and contrasting case studies. *Technology in Society*, 29(4):441 – 455, 2007.

[8] British Medical Journal (BMJ). Bmj readers choose the 'sanitary revolution' as greatest medical advance since 1840. *BMJ*, page 334:111, 2007.

[9] CBS. Bevolking: ontwikkeling in gemeenten met 100.000 of meer inwoners, 14-11-2017 (accessed: 30-2-2018). http://statline.cbs.nl/StatWeb/publication/?VW=T&DM=SLNL&PA=70748NED&D1=0,2,4,16,18,20,22,24&D2=a&D3=0&D4=a&D5=l&HD=090707-1905&HDR=T&STB=G4,G2,G1,G3, website with statistics.

[10] M. Moens E.W. Rebergen, N. Jonkers. Gemeentelijk afval-, hemel- en grondwaterplan utrecht 2007-2010. 19-01-2007. http://www.savedigiplan.nl/iprutrecht/utrecht/showfile.php?file=/iprutrecht/utrecht/userfiles/files/Gemeentelijk%20Rioleringsplan%202007-2010%20ontwerp%2019%20januari%202007.pdf, Design by municipality Utrecht for 2007-2010.

[11] author: unkown Wageningen University & Research. Meten aan rioleringen: wat, waar, hoe en met wie? H_2O, page 25/26, 2006. http://edepot.wur.nl/346440.

[12] Arjen Kruit. Predictive maintenance rioolgemalen, tableau story, (accessed: 22-3-2018). https://public.tableau.com/views/PredictiveMaintenanceGemalen/StoryPredictiveMaintenance?:embed=y&:display_count=yes.

[13] Marly van der Meij. Predictief onderhoud. 1 juni 2018. De Haagse Hogeschool te Delft.

[14] John Daly. What is turbidity? https://www.isanorcal.org/download/tech2007_presentations/turbidity.pdf, powerpoint presentation South Fork Instruments, Inc.

[15] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[16] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC Press, (book), June 6, 2012. ISBN 9781439830031.

[17] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, August 13 - 17, 2016.

[18] V. Vapnik and A. Chervonenkis. On a perceptron class. *Automation and Remote Control*, 25:112–120, 1964.

[19] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In David Haussler, editor, *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992.*, pages 144–152. ACM, 1992.

[20] scikit learn. 1.4 support vector machines, (accessed: 3-5-2018). http://scikit-learn.org/stable/modules/svm.html, (online documentation on package).

[21] (On the Quora Forum). What are some pros and cons of support vector machines?, (accessed: 12-3-2018). https://www.quora.com/What-are-some-pros-and-cons-of-Support-Vector-Machines.

[22] (On StackExchange Forum). Advantages and disadvantages of svm, (accessed: 16-5-2018). https://stats.stackexchange.com/questions/24437/advantages-and-disadvantages-of-svm.

[23] Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2342–2350. JMLR.org, 2015.

[24] Mark Hoogendoorn and Burkhardt Funk. *Machine Learning for the Quantified Self - On the Art of Learning from Sensory Data*, volume 35 of *Cognitive Systems Monographs*. Springer, 2018.

[25] Jeff Heaton. The number of hidden layers, (accessed: 31-5-2018). https://www.heatonresearch.com/2017/06/01/hidden-layers.html, (Blog).

[26] Keras. Keras documentation usage of optimizers, (accessed: 30-5-2018). https://keras.io/optimizers/, (online documentation on package).

[27] Pedro M. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, 2012.

[28] Mark Hoogendoorn and Burkhardt Funk. Machine learning for the quantified self - code, 2017 (accessed: 7-4-2018). https://ml4qs.org/source-code/, (website with the code used in the book).

[29] scikit learn. 3.2.4.3.1. sklearn.ensemble.randomforestclassifier, (accessed: 23-5-2018). http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, (online documentation on package).

[30] scikit learn. 3.2.4.3.5. sklearn.ensemble.gradientboostingclassifier, (accessed: 25-5-2018). http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html, (online documentation on package).

[31] scikit learn. sklearn.svm.svc, (accessed: 9-5-2018). http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html, (online documentation on package).

[32] Sagar Sharma. Epoch vs batch size vs iterations, (accessed: 30-5-2018). https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9, (Blog).

[33] Keras. Keras documentation lstm, (accessed: 17-5-2018). https://keras.io/layers/recurrent/#lstm, (online documentation on package).

[34] Keras. Keras documentation usage of loss functions, (accessed: 30-5-2018). https://keras.io/losses/, (online documentation on package).