university of groningen

faculty of science and engineering

Voltage Predictions in Buried Gas Pipelines

Master's thesis

August 30, 2018

Student: J. F. van Wezel

Primary supervisor: M. Biehl

Secondary supervisor: K. Bunte

Primary (external) supervisor: R. Bosgraaf

Abstract

Cathodic Protection is a method applied to many steel structures like ships, bridges, buildings, and pipelines to protect them from corrosion. It protects these structures from corrosion by applying a current to them, when the currents reach a certain threshold the structures are no longer protected. Voltages are then measured in the structures. Being able to predict these voltages is therefore deemed vital in preventing corrosion and subsequent damages on these structures.

This work focuses on voltage predictions in cathodic protected steel gas pipelines. The pipelines are held by a transmission system operator in The Netherlands called Coteq. Coteq has constructed a dataset containing yearly voltage measurements of the pipelines and a dataset containing the ground these pipelines lay in.

We applied Chebyshev imputation to account for the missing values in the voltage dataset, a sliding window technique, and three Machine Learning models to do the voltage predictions. The applied models are: k Nearest Neighbors , Multiple Linear Regression, and Learning Vector Quantization. The models were trained on a one-step scenario and then applied in a multi-step set-up by reusing the on-step predictions in the sliding window to do the longterm predictions.

We show that the one-step predictions are accurate for the tested models (classification rate of 96% for the best performing model), but improvements can still be made in the longterm situation.

Contents

Acknowledgment

Before you lies my master's thesis 'Voltage Predictions in Buried Gas Pipelines', the crown on my time as a master's student of Computing Science at the University of Groningen. I am proud to be able to present this work, it has been a long ride to finally reach the (most likely) final chapter in my life as a student. Before me lies the life as a professional. I feel I can say my time as a student as a whole has prepared me for what is to come by providing me with invaluable scientific, professional, and personal skills.

Before we dive into the world of Machine Learning and gas pipelines, I would like to thank my supervisors, all of whom have played an incremental role in providing me with advice, structure, and knowledge. Especially I want to thank my primary supervisor Michael Biehl for his guidance during this project. Furthermore, I want to thank my parents for putting up with me, and always supporting me in any possible way during my whole time as a student.

Of course, there are many friends who helped me get through this time, but I want to especially thank my fellow students and dear friends: Sebastiaan van Loon, Laura Baakman, and Rick van Veen. I have cherished memories of the times we worked together at the study-landscape. You were always there for a discussion on any topic where needed. A big thank you to all these people.

I hope you enjoy your reading.

Jelle Ferdinand van Wezel,
August 2018.

Chapter 1

Introduction

In 1959 on of the worlds largest natural gas deposit was found in Slochteren, The Netherlands [1]. It was discovered by the "Nederlandse Aardolie Maatschappij" (NAM) which translates to the Dutch Oil Company. This company was founded by Shell and Esso in 1947 [2] in order to find natural energy resources in the Dutch soil. The discovery of the natural gas deposit was kept with a low profile at first because Dutch law at that time [3] did not give ownership of a naturally occurring resource to its discoverer, instead, it gave ownership to the state. After a follow up exploratory drilling near Delfzijl showed the size of the natural gas deposit, the NAM filed for the drilling rights with the Dutch Government for the Groningen area [1].

Then in 1962, the Dutch government passed the natural gas bill [4]. It created a partnership between the Dutch state, Shell, and Esso. The Dutch state would get a share of 50%, Shell and Esso would both get a share of 25%. The bill also founded the "Nederlandse Gasuni" (the Dutch union for natural gas). This union would be responsible for distributing the natural gas from the Slochteren field across The Netherlands. At that time there were local gas companies [4] producing and distributing light gas distilled from coal. These companies would now distribute the Slochteren gas with their existing distribution network rather than produce their own.

In less than ten years most Dutch households would be connected with the gas network. The Netherlands would cook and be warmed by the natural gas from Slochteren for the foreseeable future [4, 5]. The natural gas from Slochteren brought economic prosperity to The Netherlands in the second part of the 20th century. However, there was also a downside, so much so that an economic term is named after it: 'Dutch disease' [4]. This term is used when a nation's products become expensive due to a strong currency, which is fueled by a newly discovered natural resource. Because of the strong currency, export prices to rise, expensive exports causes the nation's production to decrease and unemployment rates to rise.

Currently, 80 percent of the natural gas from the Slochteren field is thought

%	Type	Example
30	Aging	Oxidation
21	Excavation damage	
16	Soil movement	
12	Construction errors	
11	Unknown causes	
5	Point frictions	Tree roots
5	Other causes	

Table 1.1: Causes of damages on gas pipelines in The Netherlands [8]

to be extracted, and at the current rate of extraction, it is predicted that the gas will last for at least another ten years. The population of The Netherlands is profiting from the gas. However, the 'Mijnwet' gave 50% ownership of the natural gas to the state. This meant that half of the revenues went to The Netherlands as a whole and not to directly to the regionals living near the gas field. Furthermore, due to the extraction of the gas from the lower layers of the Earth, the upper layers start to shift with earthquakes as a result. Because earthquakes are not common in The Netherlands, structures are not built to withstand their impact. This causes the houses of the residents directly above and near the gas field to show signs of damages and sometimes become uninhabitable. [6]

For these reasons the people of Groningen started to oppose the extraction of the gas from the Slochteren field. Multiple protest groups have been formed over the years and not without success. As of 2017, the extraction of the natural gas will be limited over time. There have been made promises by the Dutch government to reimburse the owners of damaged homes. However, a robust framework is still to be implemented.

The earthquakes are not the only incentive for The Netherlands to stop extracting the gas. The Dutch government signed the Paris Agreement in 2015. The Paris Agreement is a climate accord in which 196 nations made promises to reduce their carbon emissions in order to slow the rising global temperature. One of the promises The Netherlands made was to reduce the carbon emissions. The Dutch government has stated that it wants to reduce the dependency on gas and start using other forms of energy instead. [7]

For these reasons, The Netherlands is moving away from fossil fuels and transitioning to sustainable energy sources like wind and solar power. The transition to 'green' energy will take time. The energy infrastructure needs to be able to handle a more significant dependency on electricity and facilities need to be built to produce the electricity. During this time the current gas infrastructure will still be in use.

Coteq is one of these TSOs. It is located in and around Almelo city. The area Coteq is active in is presented in the figure 1.1 below. In 2015 Coteq had

around 140 thousand gas connections [8]. With this number of connections, it is one of the smaller TSOs in The Netherlands. Furthermore, Coteq is part of an umbrella cooperation called Cogas. Cogas is active in multiple TSO related industries like glass fiber and energy production.

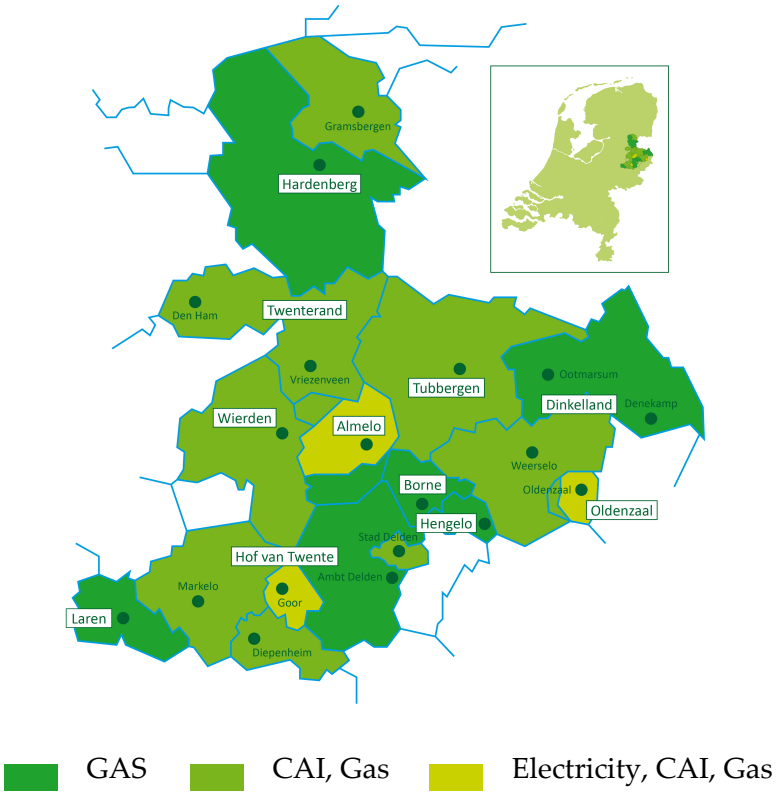


Figure 1.1: The areas of The Netherlands Coteq is active in.

For multiple tech-related solutions, Coteq has employed a 3rd party software development company, ValueA. ValueA facilitates Coteq with multiple software solutions, for example, dashboards to see energy consumptions in certain neighborhoods, communication software, and hardware consulting.

In recent years Machine Learning, a subfield of Computer Science, has made some incremental advancements, This provides us with the ability to, either through simple statistics or more elaborate algorithms, gain knowledge from any data. At the start of the winter in 2017, Coteq and ValueA had multiple datasets on which they wanted to perform an analysis. The University of Groningen was contacted to collaborate on a Machine Learning project with these datasets. This collaboration request resulted eventually in this Master thesis project.

The project is to predict voltages based on measurements done on gas

pipelines to be able to prevent corrosion. The reasoning behind this project is that pipelines in the ground are protected from corrosion by applying a current to them. This current is measured regarding voltages on multiple parts of the pipeline. These voltages change over time due to a multitude of factors. When the voltages rise above a certain threshold, the pipelines are no longer protected, and specific actions need to be taken. Based on the measurements dataset it might be possible to make a voltage prediction in the pipelines for the coming years. Is it possible to find a Machine Learning model, that is capable of predicting the voltages accurately for the coming years?

1.1 Related work

As mentioned before TSOs in general, and thus Coteq, invest primarily in maintaining their pipeline infrastructures. Furthermore, they are searching for new techniques to make their infrastructure more durable or gain new insights. One way of doing this is by formulating research projects and providing datasets for them. Based on the measurement dataset and the project description we decided to make time series predictions on the voltage measurements extracted from the pipelines.

Recent advances in time series predictions have shown promising results and are being employed for a great multitude of applications. Especially Neural Networks have made incremental steps forward in the past few years [9, 10], but also non-linear models [11]. In order to perform the predictions, we applied a Machine Learning model, known as Learning Vector Quantization. Time series prediction with this type of model has been made in the past, but the literature on this topic is scarce. There is, however, a paper by Hammer et al. [12], where the authors lay out a method for predictions with the LVQ model on time series extracted from the Lorenz system, which describes atmospheric pressures based on a differential equation [13].

Learning Vector Quantization is a form of prototype-based learning which has been used to make short and long-term predictions on time series. The authors of [14] used two techniques to predict the conversion rate at the end of the day from the dollar to the rupee. Another example of this is a paper by Poulos et al. [15], where prototype-based learning was used to classify stationary from non-stationary time series. Another example is the work done by de Lautour et al. , where an autoregressive model with a Learning Vector Quantization model was used to predict the structural integrity of a bookcase [16].

Multiple other Machine Learning techniques have been applied to predict some form of deterioration of pipelines in recent the years. An example of a statistical approach to this problems is a paper by Pesinis et al. [17] where a parametric hybrid empirical and nonlinear quantile regression was used

to predict the metal loss in onshore gas pipelines. Work by Qiu et al. exploited a nonlinear regression model to predict the condition of the coating of pipelines [18]. The problem can also be described as predicting the moment of failure of a pipeline. This method was applied by Meyer and Ruth, where a logistic regression model was used to predict a corrosion leak[19]. Furthermore, a Neural Network was exploited to classify the condition of offshore oil pipelines in Qatar by El-Abbasy et al. [20]. Sewer pipelines pose similar problems to that of gas and oil pipelines. Dulcy et al. [21] estimated sewer deterioration by applying a Markovian model.

1.2 Project Pipeline and Thesis Structure

In this section the overall structure of the performed research and thesis will be described.

Figure 1.2 shows a schematic representation of the project. It starts with three different datasets. The datasets contained the voltage recordings done by Coteq on the gas pipelines. The Pipelines dataset contained the geographical locations of the pipelines and their length, size, and other detailed information on the pipelines. The Ground Areas dataset carried information on the ground the pipelines were buried in. It contained for example information on the water level and acidity of the soil as well as soil type and stability and its geographical location.

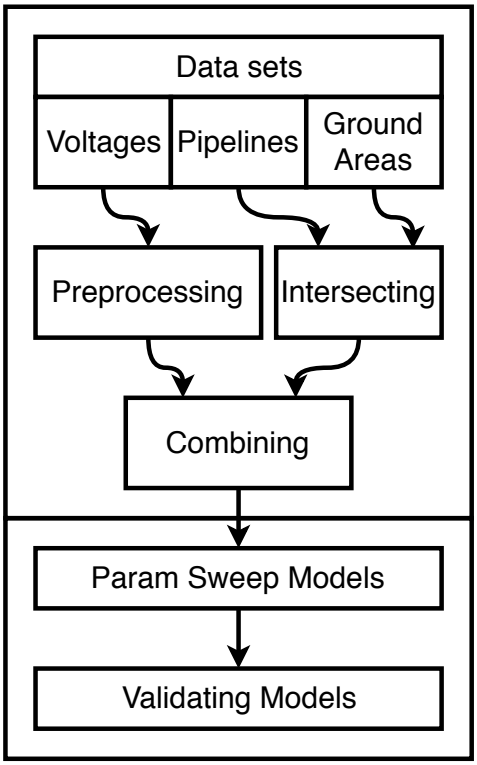


Figure 1.2: Project pipeline

The Voltage data contained human errors and needed to be transformed before it could be given to the applied Machine Learning model. These methods are described in the chapter 3 and generalized in the schema as 'Preprocessing'. The data from the pipelines and the ground was intersected to determine what pipelines lay in what ground, some other problems with this data needed attention and the performed methods on these datasets to achieve the intersections are described in more detail in section 3.6.

After preprocessing and intersection, the data were combined to form a single dataset containing all the voltages and ground information of all the pipelines surveyed by Coteq. With this dataset, a parameter sweep was performed on the Machine Learning models. A total of three different models were tested, the LVQ model already mentioned in chapter 1, and two other models that were used as a comparative baseline to evaluate the performance of the LVQ model. These models are described in detail in chapter 4.

The pipeline ends with validation of the models. This is where the models are introduced to new information it has not seen during the parameter sweep. The parameter sweep is to ensure the model was not overfitted on the presented data during the selection of an optimal model. section 4.4 gives a

more detailed report on the need for validation.

The next chapter will give some background information on Cathodic Protection, and the method applied to protect the pipelines. Chapter 3, will give a detailed account of the dataset and what methods were used to preprocess the dataset. In chapter 4, Modeling and Validation, the models and validation methods applied to the dataset will be addressed. Then the implementation, in chapter 5, will describe choices made on the models and the software written for this project. Then we will show our results in chapter 7 and present a discussion of the work done in chapter 8. Chapter 9 will recap the most significant findings in this project.

Chapter 2

Cathodic Protection

The dataset was partly obtained from voltage changes in buried steel pipelines. These pipelines carry currents making them cathode in order to protect them from corrosion. This method is called Cathodic Protection. In this chapter the history, fundamentals, and application of Cathodic Protection will be addressed. We will look at the background of this method section 2.1 followed by the electrochemical process, section 2.2, and the type of Cathodic Protection, and coating that is applied to the pipelines by Coteq will be addressed section 2.3. This chapter is concluded with section 2.5 where the monitoring method utilized by Coteq to record the voltages are described.

2.1 Background

When steel pipelines are buried in the ground they get exposed to a multitude of damaging factors. The main causes for damages are shown in table 1.1 in chapter 1. Other causes can be for example aggressive soil conditions, microorganisms, and stray currents e.g. from railway tracks. The main way of protecting the pipelines from these hostile factors is by applying a coating to it. However, even this coating gets damaged over time. To ensure the pipelines stay protected, cathodic protection can be employed. Cathodic protection is, as indicated by recent research a promising method to protect metal pipelines [22–24].

As early as 1824 Sir Humphrey Davy [25] reported that by connecting copper to one of the lesser galvanic metals, zinc, and iron, it could be protected against corrosion. About one hundred years later in the 1920's the method was applied for the first time on buried pipelines transporting gases and oil. Since then Cathodic Protection became a widely used method for protecting pipelines, metal structures, and ships.

2.2 Electrochemical process

Cathodic protection is defined as a reduction or elimination of corrosion by making the metal a cathode [26]. This can be achieved by attaching a sacrificial metal (anode) or by impressing a current. By doing so, an electrochemical process is started. Cathodic polarization can then be used to influence the corroding processes. The processes described here is based on work by M. Kutz in [26].

With the Wagner Traud mixed potential theory [27, 28] the principle of Cathodic Protection can be explained. As a simple example of the process, iron (Fe) is placed in an aerated neutral electrolyte. The corrosion reactions that occur are as follows:

$$\mathrm{Fe} \rightarrow \mathrm{Fe}^{2+} + 2\,\mathrm{e}^{-}, \tag{2.1}$$

$$\mathrm{O_2} + 2\,\mathrm{H_2O} + 4\,\mathrm{e}^{-} \rightarrow 4\,\mathrm{OH}^{-}. \tag{2.2}$$

Corrosion processes are divided into two or more oxidation and reduction partial reactions. The oxidation reaction for the example is shown in eq. (2.1) and the reduction reaction in eq. (2.2). During this reaction, none of the partial reaction no net accumulation of electric charge should occur. This ensures that an equilibrium state between the partial reactions can be reached. Here the total rate of oxidation equals the total rate of reduction. In fig. 2.1 the relationship between the two partial reactions is shown in an Evens diagram.

In fig. 2.1 the potential of the equilibrium state is indicated as E_{corr} and the current as I_{corr}. At the equilibrium state, the total rate of oxidation is equal to the total rate of reduction. Here the oxidation reaction supplies the exact amount of electrons the reduction reaction needs to occur.

The reversible potential for iron is indicated as $E_{eq,\mathrm{Fe}}$. Here the iron is in its equilibrium state, and it does not corrode. The difference between the corrosion potential and the reversible potential is the driving factor for the corrosion to occur. When the system is polarized by applying a current from I_{corr} to I'_{corr} , with a known current I_{app}, the effects of the corrosion current is decreased. The corrosion processes can be halted entirely when the corrosion current is brought back to the metal its reversible potential ($E_{eq.\mathrm{Fe}}$).

An example of this is shown in fig. 2.2. Here iron is shown corroding in an acidic environment. The current needed to halt the corrosion processes is shown as $i_{\text{protection}}$

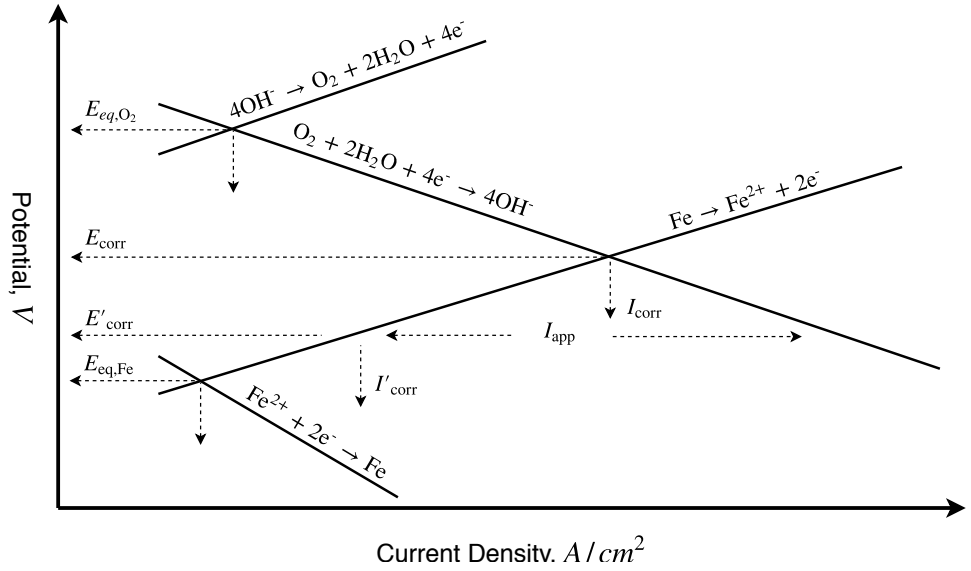


Figure 2.1: Evens diagram for Fe system in a neutral environment. This image was adapted from [26]

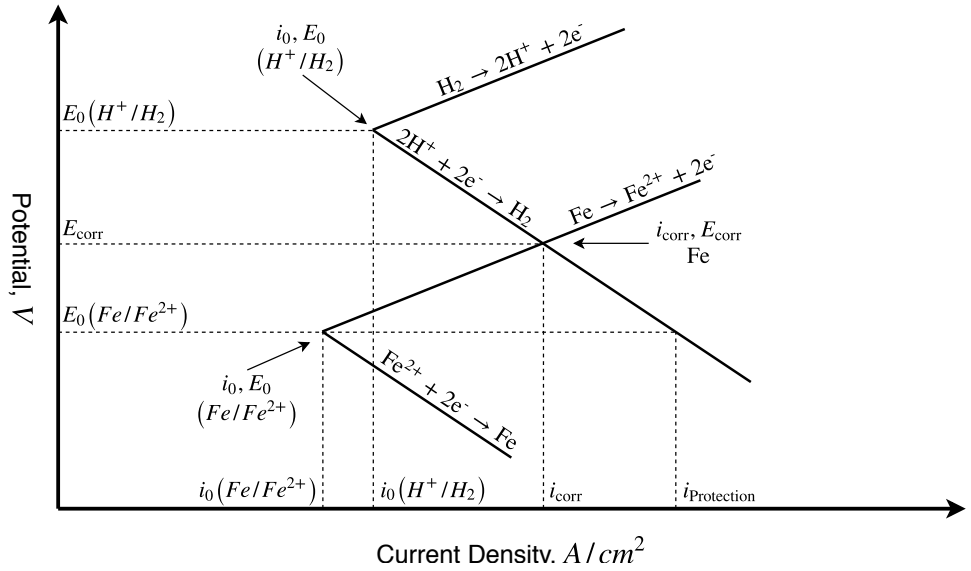
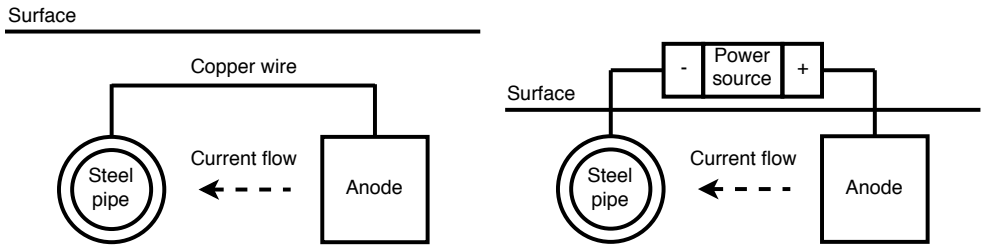


Figure 2.2: Evens diagram for Fe system in an acidic environment. This image was adapted from [26]

2.3 Types of Cathodic Protection



(a) System with a passive anode. (b) System with an active anode.

Figure 2.3: Schematic side view of the two types of cathodic protection. These figures were adapted from [26]

Cathodic protection can be divided in generally two groups: Sacrificial anode and impressed current. As discussed in the previous section 2.2, the corrosion processes can be halted by introducing an outside current to the system. This can be achieved in two ways. The first way is to use an anode. The anode is a piece of metal that is less noble than the to be protected cathode in the galvanic series. The anode is more electronegative than the pipeline, causing a current to flow. The corrosion then happens on the anode, and the cathode stays intact. This method generally has low maintenance because there are no moving or electronic circuits involved. This is schematically shown in fig. 2.3a.

In fig. 2.4 two examples are given where sacrificial anodes are employed. In fig. 2.4a it is visible that multiple anodes are needed to cover the entire structure. The range an anode covers is a limiting factor when using passive cathodic protection. Another disadvantage of passive cathodic protection is the pacification of an anode. This is depicted in fig. 2.4b where an anode is placed on the hull of a ship and is corroding instead of the ships hull. However when the anode is pacified the anode either needs to be replaced, or the ships hull will start to corrode.

The second method of cathodic protection is with an impressed current. This is schematically shown in fig. 2.3b. Here the current is added to the system by a rectifier. The external current is used to polarize the steel pipeline cathodically. In theory, the pipeline is protected by current impressed cathodic protection, and it can be used for bare pipelines or partially coated pipelines. However, there are multiple drawbacks to be taken into consideration. This method has more maintenance than the sacrificial anode system because it has a multitude of electronic circuits, the amount of current impressed needs to be monitored, and the system is vulnerable to power outages [26]. The voltage needed in the pipelines at any point needs to be -850 mV to be fully protected from corrosion [26, 29, 30].

(a) Cathodic protection with multiple sacrificial anodes on a steel structure. Image by Wikipedia user Chetan and shared under the Attribution-ShareAlike 2.5 Generic license. This image was not altered in any way.

(b) Cathodic protection with a sacrificial anode on the hull of a ship. Image by Wikipedia user Zwergelstern and shared under the Attribution-ShareAlike 3.0 Unported license. This image was not altered in any way.

Figure 2.4: Examples of cathodic protection with a sacrificial anode.

2.4 Coating

Cathodic protection is always applied as a secondary method of protection. The first method is usually a coating applied to the metal structure. When the coating breaks or fails the cathodic protection ensures the structure stays free from corrosion. Multiple materials can be used as a coating, tar and asphalt enamels, mastics, waxes, polyvinyl chloride, polyethylene tapes, thermosetting epoxy resins, and epoxy coating [30]. Coatings are exposed to the same dangers as the pipelines themselves. These dangers are shown in table 1.1. The pipelines used by Coteq are coated with to different types of coating. Older pipes were coated with tar and newer pipes with Poly Vinyl Chloride. The type of coating might influence the voltage measurement and in section 3.6 we will further address the implementation of this data.

When too much current is applied to the pipelines, the electrochemical process causes hydrogen to form on the surface of the pipelines. The hydrogen then forms bubbles between the coating and the pipeline. Eventually, this bubble bursts, damaging the coating [29]. Another concern is a process called Hydrogen Embrittlement, damaging the metal of the pipelines them selfs [30].

2.5 Monitoring

In order to keep track of the effectiveness of the cathodic protection with an impressed current, frequent measurements of the potentials, voltages, or cor-

rosion are needed. There are several measuring techniques. Here we will only describe the method employed in the field by the company Coteq i.e., potential measurements with a reference electrode. The records of this method resulted in the obtained dataset. Other monitoring methods like CIPS, DCVG, IR Coupons and corrosion rate measurements are described in [26] for the interested reader.

A potential measurement with a copper-copper sulfate $(Cu/CuS04)$ reverence electrode was carried out on average every year since the 1980s. The copper-copper sulfate electrode is the most commonly used reference electrode for soil environments and cathodic protection [30].

(a) Cathodic protection rectifier as used in the field to impress current into the system. Image by Wikipedia user Cafe Nervosa and shared under the Attribution-ShareAlike 3.0 Unported license. This image was not altered in any way.

(b) Cathodic protection measure point in Leeds, England. Image by Wikipedia user Mtaylor848 and shared under the Attribution-ShareAlike 3.0 Unported license. This image was not altered in any way.

Figure 2.5: Monitoring attributes for cathodic protection

The measurement is carried out by bringing the reference electrode in contact with a surfacing part of the pipeline. In fig. 4.1c a measurement point is shown where a wire from the pipeline is surfaced explicitly for measurement purposes. The reference electrode is placed in contact with the ground. In order to ensure good contact, the ground should be dampened. The potential measurement can now be carried out [29].

When the potential of the pipeline is measured, there is always a measurement error. This error is caused by the resistance R of the ground. Due to the direction of the current, this leads to a $(I \times R)$ loss of potential measured in the pipelines and thus to an unknown error. The I in $I \times R$ is the amount of impressed current at the rectifier, shown in fig. 4.1a. This loss of potential can, however, be estimated. According to Klink BV [29] there are multiple complicated methods for determining this loss, but it is sufficient to turn the rectifier off and on in a small time interval (seconds). The idea behind turning

the rectifier on and off is shown in fig. 2.6.

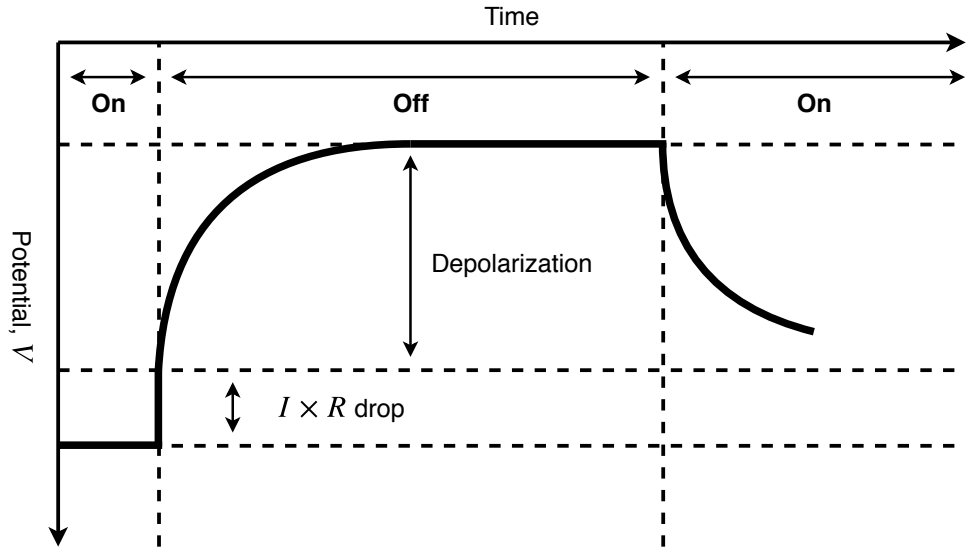


Figure 2.6: This diagram shows the depolarization of pipelines over time in a current impressed cathodic protected system. The $I \times R$ drop is indicated and happens right after impressing current into the system is stopped.

In fig. 2.6 the depolarization of current impressed metal is schematically shown. When the rectifier is turned off the ground depolarizes followed by the metal of the pipeline. When the potential of the pipeline is measured right after the rectifier is turned off it may be assumed that the true potential is measured [29]. The Figure shows the full depolarization of the metal in a 4 to 24 hour period. After this period the current is impressed again, and the metal starts to polarize again.

One of the complications of using the measure points, shown in fig. 4.1c, is that older concrete measure points offer the ideal habitation spot for ants. The old concrete measure points are relatively short and have enough room for ants to create their nests in them. These ants excrete acid and cause oxidation of the electronic contacts in the measuring point, ultimately causing outages of the cathodic protected system. Newer measure points are made of synthetic material and are raised higher above the ground than the old short concrete measure points, making them immune [29]. A comment, attached to the measurements, often described ant nests being present in the measure points which may have influenced the obtained dataset. This will be further addressed in section 3.2.

Chapter 3

Data and Processing

"The goal is to turn data into information, and information into insight" - Carly Fiorina, former executive, president, and chair of Hewlett-Packard Co. As Carly Fiorina stated, data alone is not enough. A big part of any scientific project is concerned with turning data into something useful. In order to predict measurements for the measurement points, meaningful data needs to be used. We decided to not only use the voltage measurements but data about the pipelines in the ground as well. This chapter will layout what data was used and how it was obtained to ultimately form the dataset that was used to solve the problem.

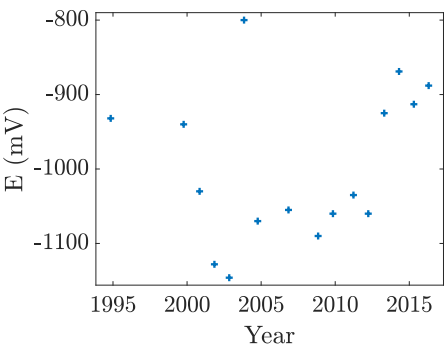
3.1 Voltage measurements

To shortly recap the previous chapter when employing cathodic protection on buried pipelines, a small current is impressed in to the ground. This current polarizes the pipelines and protects them from corrosion, see chapter 2. The potential in the pipelines can be measured and should be below $-850mV$, section 2.3, for them to be completely protected. However too much impressed current causes the potential of the pipelines to be too low and can cause hydrogen embrittlement, section 2.4. Not only hydrogen embrittlement but also Dutch regulations limit the amount of current impressed into the ground.

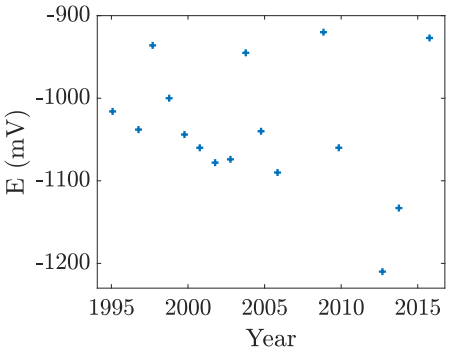
The potential of the pipelines can be measured. This is done by the method described in section 2.5. The first measurement was performed at the third of November 1987, from then on the potential of the pipelines was measured once every year until the last recorded measurement at the first of September [1] 2016. In figure 3.1 two examples of measurements are shown. In the figures, a line connects the observed data points, but the data seems to fluctuate greatly. The observed fluctuation might be small errors in the measuring process. However, the true distribution behind the data depends on many

[1] Coincidently the authors birthday.

factors: stray currents coating, the weather, the soil, etc. The observations shown have a span of twenty years, but there were only sixteen measurements performed.



(a) Voltage measurements from the area Deurningen measure point number 477



(b) Voltage measurements from the area Almelo Windmolenbroek and measure point number 1225.

Figure 3.1: Two examples of measurements from different measure points. Potentials were measured versus a $Cu/CuSO_4$ electrode. Both of these measurements span a period of twenty years but have sixteen measurements.

3.2 Missing Data

The voltage measurements are done at designated measuring points. A pipeline can have multiple of the points. Usually, the measurements are conducted once per year at every measure point. The measurements started as early as 1987 up-until 2016. This means there should be 39 measurements per measuring point. However, this is not the case as is shown in fig. 3.2. In this figure, the number of measure points is shown that have a certain number of measurements i.e., recordings.

Figure 3.2 shows a fast drop in measurement points after the 20 measurements mark. At 26 measurements the drop stagnates, and the maximum number of measurements is reached at 37 measurements. This means that none of the measuring points have the maximum number of 39 measurements.

As mentioned in section 2.5 the measurements were taken by surveying the pipelines and recording the observations. The measurements started in 1987 and in that time measurements were recorded merely with pen and paper. Later these observations were stored in a database, and the previous measurement typed over from the paper records and inserted into the database. This practice introduced human errors into the data. A simple analysis of the observations shows this. The lowest observed potential in

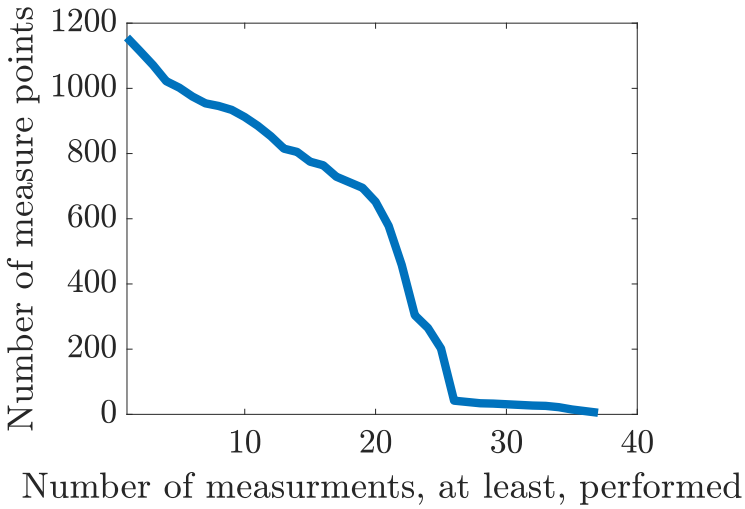


Figure 3.2: The number of measurements performed on the measure points, e.g. each measure point has at least one measurement but not all have measure point have 30 measurements. As the number of measurements increases the number of measure points containing that number of measurements decreases with a substantial drop at the 20 measurements mark.

the dataset was $-11320000mV$. Although possible, the rest of the data had values between 0 and -1600 mV thus $-11320000mV$ seems excessive. These observations were therefore completely removed.

On the other side of the spectrum, 129 observations had a recorded value of $0mV$. This is also not an impossible value since the difference in potential versus an $Cu/CuSo2$ electrode was measured and the potential of a pipe segment can very well be the same as that of the electrode. However, twenty of these measurements had a comment stating 'Unreachable', 'Need repair' or another reason indicating something was either wrong with the measurement point or with the recording equipment. It is plausible that at these moments a $0mV$ was recorded. We decided to remove the measurements with a comment stating something was wrong.

3.3 Chebyshev Polynomials

As discussed in section 3.2, the obtained data from the measure points showed a high frequency of missing data combined with inconsistent times of measuring and notation errors. In order to extract consistent time series data from these inconsistent measurements, a form of interpolation was applied to extract evenly spaced time intervals from the dataset. Melchert et al. used a similar method on different datasets in [31, 32]. Here the authors apply a first order Chebyshev polynomial approximation of functional data on example

datasets. The method presented here is based on this approach. For a more detailed description of Chebyshev polynomials, we refer to [33].

We assume the discrete time data obtained from the measure points result from sampling an unknown function $f(t)$. The time intervals were scaled to $t \in [-1 \dots 1]$, and with this, the observations are denoted as

$$x_{i,j} = f_i(t_j). \tag{3.1}$$

According to the authors of [31], the function $f(t)$ can be expressed as a weighted sum of a set of suitable basis functions $g_k(t)$

$$f_i(t) = \sum_{k=0}^{\infty} c_{i,k} g_k(t). \tag{3.2}$$

If k is limited to an appropriate number of coefficients n the approximation of f is obtained. The authors note that limiting the number of coefficients gives in general an approximation of f.

$$\hat{f}_i(t) = \sum_{k=0}^{n} c_{i,k} g_k(t). \tag{3.3}$$

As basis functions Chebyshev polynomials were used. The first order Chebyshev polynomials are defined as follows:

$$T_n(x) = \cos\left(n \ \cos^{-1}(x)\right), \quad x \in [-1, 1], \quad n = 0, 1, 2, \cdots. \tag{3.4}$$

From this we can derive,

$$T_n(\cos \ \theta) = \cos(n\theta), \quad \theta \in [0, \pi], \quad n = 0, 1, 2, \cdots. \tag{3.5}$$

By using the above equation, the recursive definition can be stated as

$$T_0(x) = 1; \qquad T_1(x) = x; \qquad T_n(x) = 2xT_{n-1} - T_{n-2}(x). \tag{3.6}$$

In fig. 3.3 the first six polynomials are plotted to show the increasing complexity as n increases.

The coefficients c_i, k of the approximation can then be found by minimizing an error function like the square error: $e = \sum_{j=1}^{d} (f_i(t_j) - \hat{f}_i(t_j))^2$ or the maximum deviation error: $e = \max_{j=1\cdots d} (f_i(t_j) - \hat{f}_i(t_j))^2$. However as mentioned in [31, 33] the properties of the limited Chebyshev series can be exploited to obtain the coefficients more efficiently.

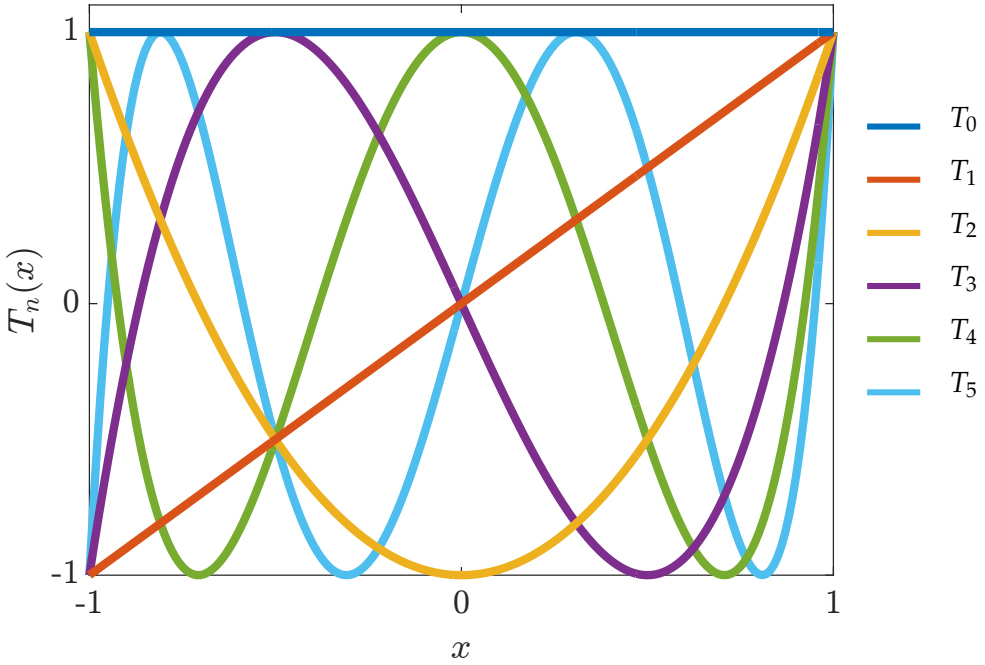


Figure 3.3: The first 6 first order Chebyshev polynomials. The progression of the polynomials complexity as n increases from 0 to 5 is clearly visible.

$$c_{i,k} = \frac{2}{n+1} \sum_{n}^{l=0} f_i(t_l) T_k(t_l), \quad \text{with} \quad t_l = \cos\left((l + \frac{1}{2}) \frac{\pi}{n+1} \right). \tag{3.7}$$

Here t_l are so-called sampling points that represent the roots of the Chebyshev polynomial of degree $(n+1)$ [31]. The real values for $f(t_l)$ will in most cases be unknown. However, by applying a linear interpolation between two known points, we can get an approximation of the real sample. According to the author of [31] this is justified since if the number of samples brings enough density to the time series the real point will most likely lay close to the approximated point. Furthermore, there are more complicated methods to predict these samples [31, 34] but for our purposes, linear interpolation reduces the complexity of the overall model and is therefore deemed sufficient.

3.4 Sliding Window

As mentioned in chapter 1, we implemented a classification model to predict the voltages from a known history of voltages. In section 3.1 these measurements are discussed and in section 3.3 the recorded measurements were interpolated to form consistent time series. In order to feed these time series to the classification models, a fixed number of features needs to be extracted.

One way of obtaining a fixed number of features is to find the coefficients by for example a Fourier transform or another appropriate method and use the coefficients as features to classify a time series. However, our objective is to make predictions based on the recorded measurements. In order to achieve this, a sliding window was applied to the time series.

A sliding window is a widely used practice to express a time series or signal in smaller parts with a fixed size. We define a time series by: $Y = \{Y_t : t \in T\}$ where T the set of integers from 1 to the width of Y notated as Y^w. Here S is a subset of Y with a fixed width: S^w. The size of the window determines the number of subsets that can be extracted from a series: $T^w - S^w + 1$. The table below gives an example of a sliding window where $S^w = 3$ and $Y^w = 9$.

Y	[	1	2	3	4	5	6	7	8	9	]
S_1	[	1	2	3	]						
S_2		[	2	3	4	]					
S_3			[	3	4	5	]				
S_4				[	4	5	6	]			
S_5					[	5	6	7	]		
S_6						[	6	7	8	]	
S_7							[	7	8	9	]

Table 3.1: Example of a Sliding window with $s^w = 3$ and S_i the i^{th} window of Y.

The window gives a sub-history of the data. From the window, the last element is taken as a label. The label is needed for training and testing the classification models. When we feed a model the first window S_1, minus its label, from the example the excepted answer will be 3.

Instead of trying to predict the next value, we can also try to predict the change from the last known value to the label. This can be done by taking the gradient $\Delta Y/\Delta t$ from the last known value to the next value. The gradient can then be taken as the label. The next value can then still be determined by adding the gradient to the Y value at time-step t.

The elements of the sliding window can now be used as the features in a feature vector $\vec{x}$ with a label y to train and test the classification models.

3.5 Binning

The labels extracted with the sliding window from the voltage recordings are continues values. However, in order to classify a feature vector based on its label, the class needs a discreet index. This can be achieved by a practice called binning.

Binning is generalizing a range of continues values to a discreet index. We took the maximum value of the voltage recordings and the minimum values seen in the recordings. A number of bins were chosen, and the range between

the maximum value and minimum values was divided by the number of bins. This resulted in a subrange per bin. When a label fell in a particular bin range, it was given that bin's index. The indexed values where then used as labels during classification.

When the real values are needed after binning in a later prediction step, an approximation of the real value can be achieved by taking the average value of a bin's real value range. When the number of bins is chosen large enough, this will result in a reasonable approximation of the actual value. However, some expected error, depending on the bin size, is always included.

3.6 Static Data

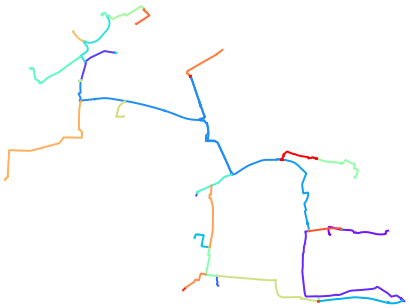
The steel pipelines were buried in the ground, the locations of the pipelines and the measure points were recorded in the obtained dataset as a 'geom' object. A geom object is a geometrical object containing coordinates of the actual location of the pipelines. The given dataset also contained a sub-dataset with areas describing the soil compositions and the soil's features. In the figure below the four examples are plotted of the cathodic protected areas.

The data consisted of 24 cathodic protected areas. Each of these areas have one anode point where some current is pushed into the ground. The current is then propagated through the ground and led back to the anode through the steel gas pipes completing the electric system.

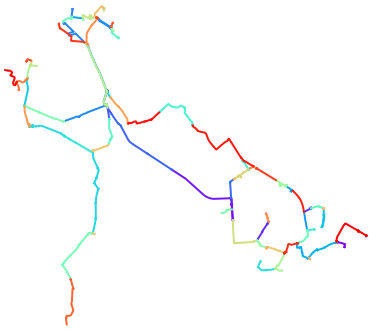
The pipes spread from the anode and branch off from each-other, creating a tree like structure. To prevent cycles in this system some pipes are connected by a plastic sub-pipe. The pipes are divided into segments, usually where a pipe branches into two pipelines, two new segments form. A pipe can only be part of one segment but a segment can consist of multiple pipes. In between the segments measuring points are placed to measure the current and difference in voltages in the segment. The voltages recorded from these measure points are the measurements used in the rest of this project. The number of segments and measuring points differ per area, the number of measure points per area and their statistics is presented in appendix A

The static features extracted from the geological location of the pipes. The ground itself and the features of the pipes were extracted from three different datasets. The identity numbers of the pipes were known and the identity numbers of the measure points. With this the pipes were intersected with the ground areas they lie in as well as some information of the pipes themselves, like pipe length, construction year, and coating.

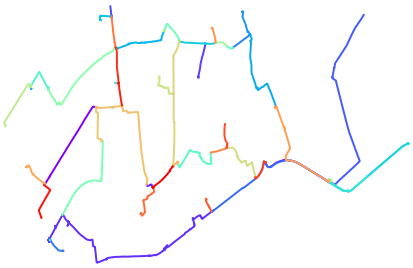
The ground types, the pipes lie in, were stored with only a geographical location. The geographical system used to store the coordinates were different from the one the pipes were stored in. After transforming both systems to the same geographical system, we intersected the pipes with the ground areas. These ground areas had different features: acidity, water level, sta-



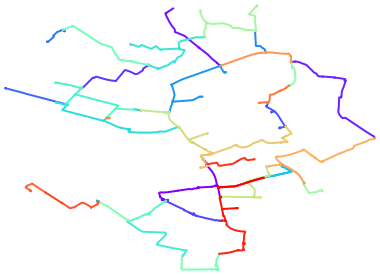
(a) Cathodic protection area: Almelo ten Cate



(b) Cathodic protection area: Almelo Tusveld



(c) Cathodic protection area: Almelo de Pook



(d) Cathodic protection area: Oldenzaal

Figure 3.4: Four examples of cathodic protection areas with a top down view. The different colors indicate the pipelines that lead current to the same measure point.

bility, and ground type. These features are generalized to a small number of categories. The full list of the features and their categories are shown in appendix B.

A pipeline can be in multiple ground areas and can thus have multiple ground area features. We represented this by taking the length a pipeline that is in a ground area and saving that length as a feature. For example a pipeline is in two ground areas, the first area has high acidity and the pipeline runs for 100 meters in this area. The second area has low acidity and the pipeline runs for 50 meters through this area. The partial feature vector for the acidity part will then look as shown in the table below:

#	Feature...	$\text{Acid}_{\text{high}}$	Acid_{low}	Feature...
1	$\cdots$	$100m$	$50m$	$\cdots$

Table 3.2: Example of a feature vector with the extracted data.

This approach is applied for all the pipelines and all the ground areas and their categories. This resulted in feature vectors with 22 of these static features. Features were extracted from the pipes them-selfs too, here the coating is noteworthy since there were two types of coatings, one of plastic and one of tar. This feature was registered as the percentage of meters a pipe segment was coated with a plastic coating. The remainder of the pipeline was thus coated with tar. We assumed that all pipelines had either one coating or the other.

Chapter 4

Models and Validation

In this chapter, the applied classification models and their validation will be discussed. Section 4.1 will discuss the popular Nearest Neighbor method. Then we will continue with multiple regression in section 4.2. These methods form the baseline for prediction of the voltages in the pipelines. In section 4.3 Learning Vector Quantization, the focus model of this study, is reviewed. This chapter concludes with section 4.4 where we lay out the validation of these models.

4.1 Nearest Neighbors

Nearest Neighbors (NN) is, since its introduction in 1967 by Cover and Heart [35], often used as a baseline in classification problems because of its simplicity and high applicability on a broad set of classification problems.

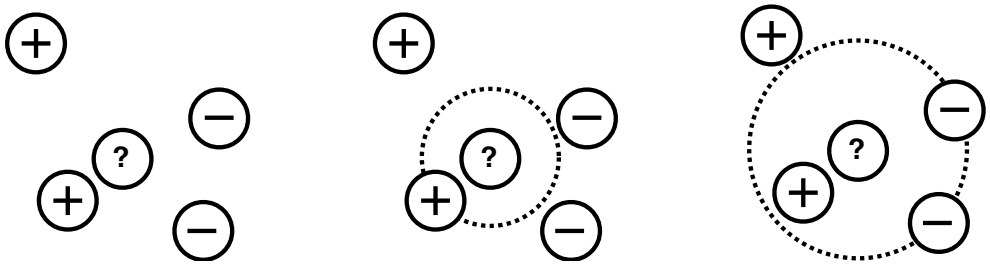
Let $D = \{\vec{x}_1, \ldots, \vec{x}_n\}$ be a dataset with n data points of which the labels are known. According to the Nearest Neighbor rule, we can classify a test point $\vec{x}$ by letting $\vec{x}' \in D^n$ denote a prototype nearest to $\vec{x}$ and assigning it the prototype's known label. In other words: 'If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.' Nearest neighbors will, however, lead in most cases to a suboptimal error rate greater than the possible minimum, the Bayes rate, but it will never be greater than twice this rate [36].

Nearest Neighbor performs better when the size of the dataset is large. This can be expressed in probabilities. Let θ' be the known label of a prototype and ω_i the label of a test point. The label θ' connected to a prototype can be seen as a random variable. Then $\theta' = \omega_i$ is the a posteriori probability $P(\omega_i|\vec{x}')$. When the dataset is large and thus the number of data points is large then $P(\omega_i|\vec{x}') \simeq P(\omega_i|\vec{x})$ is a reasonable assumption because $\vec{x}$ will generally be close to $\vec{x}'$.

A logical extension of Nearest Neighbors is k Nearest Neighbors (KNN). This algorithm classifies a test point x by taking the k nearest data points and

assigning it the label of the majority represented prototype labels (fig. 4.1). For a majority vote on the label to be reached k is in most cases chosen to be an odd number to avoid ties [37].

An exciting property of Nearest Neighbors and k Nearest Neighbors is their variable window sizes. The algorithms evaluate an area around an unknown test point. If the region of this test point has a high density the chance there are k data points nearby will be substantial, the classification, therefore, will be based on this small local area. When the data is more sparse in this region, the area will automatically increase because points will be further away. The resulting classification will, therefore, be based on a larger area.



(a) Data point to be classified in a 2 dimensional input space.

(b) Classification with $k = 1$ (Also called Nearest Neighbors), here the test point will by assigned the label '+'.

(c) Classification with $k = 3$, here the test point will by assigned the label '−' because the majority of the data points have this label.

Figure 4.1: KNN with different values for k in a binary classification problem. The circle is to show which data points are actually closest to the test point. The size of the search area is defined by the k^{th} furthest point.

A practical issue with k Nearest Neighbors is that the distance between the test point and the data points is usually calculated with the Euclidean distance measure. The issue arises when assessing classification problems with a large number of features.

Take as an example a 20-dimensional space where only two dimensions are relevant for the classification task at hand. When classifying the test point, the two relevant features might be close together but there is an equal chance the other 18 features are far away from each other. This results in a misleading similarity metric. This issue is also referred to as the 'curse of dimensionality' [38].

4.2 Multiple Regression

'Multiple regression analysis is one of the most widely used of all statistical methods.' [39] Here we will describe the basics of this method, because of the popularity of this method there is a wide range of literature available for a more detailed view on this and adjacent methods, we refer to [39].

Multiple Regression is a form of linear regression where the variable y depends on multiple independent variables $x_0, x_1, \cdots, x_d$ where d is the number of independent variables and also the number of dimensions in the dataset. Here x_i is the i^{th} observation and y_i is a known continues value belonging to this observation. The different observations $(\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_n)$ are here notated as the matrix $\boldsymbol{X}$ with the observations on its rows and the dimensions on its columns. The linear model is expressed as,

$$y_i = \beta_0 + \beta_1 \boldsymbol{X}_{i,1} + \beta_2 \boldsymbol{X}_{i,2} + \cdots + \beta_d \boldsymbol{X}_{i,d} + \epsilon_i, \tag{4.1}$$

which can be expressed as a sum,

$$y_i = \sum_{k=0}^{d} \beta_k \boldsymbol{X}_{i,k} + \epsilon_i, \quad \text{with } \boldsymbol{X}_{i,0} = 1 \tag{4.2}$$

When there is one feature, $d = 1$, the equation in 4.2 is reduced to the simple linear regression model with one variable:

$$y_i = \beta_0 + \beta_1 \boldsymbol{X}_{i,1} + \epsilon_i, \tag{4.3}$$

Here $\vec{\epsilon}$ are the residuals and are independent normal distributed random variables with $(\epsilon) = 0$. Since $\vec{\epsilon}$ is expected to be 0, (y) can be written as,

$$(y) = \beta_0 + \beta_1 \boldsymbol{X}_1 + \beta_2 \boldsymbol{X}_2 + \cdots + \beta_d \boldsymbol{X}_d \tag{4.4}$$

The equation in 4.2 is often written in its matrix form. In order to express eq. (4.2) in matrix form, the following matrices and vectors are defined:

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \tag{4.5}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & \boldsymbol{X}_{1,1} & \boldsymbol{X}_{1,2} & \cdots & \boldsymbol{X}_{1,d} \\ 1 & \boldsymbol{X}_{2,1} & \boldsymbol{X}_{2,2} & \cdots & \boldsymbol{X}_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \boldsymbol{X}_{n,1} & \boldsymbol{X}_{n,2} & \cdots & \boldsymbol{X}_{n,d} \end{bmatrix}, \tag{4.6}$$

$$\vec{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \end{bmatrix}, \tag{4.7}$$

$$\vec{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}. \tag{4.8}$$

With these definitions, we can write the multiple regression model as follows (where a resemblance can be noticed with eq. (4.3))

$$\vec{y} = X\vec{\beta} + \vec{\epsilon}. \tag{4.9}$$

The expected value $(\vec{\epsilon}) = 0$ and the variance-covariance matrix for $\vec{\epsilon}$ is defined by,

$$\sigma^2(\vec{\epsilon}) = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} = \sigma^2 \boldsymbol{I}. \tag{4.10}$$

Where $\boldsymbol{I}$ is the identity matrix. Because the expected value of $\epsilon = 0$, the expected value for the $\vec{y}$ is

$$(\vec{y}) = \boldsymbol{X}\vec{\beta}. \tag{4.11}$$

And the covariance-variance matrix for $\vec{y}$ is the same as that of $\vec{\epsilon}$. To estimate the regression coefficients $\vec{\beta}$ the least squares method is applied,

$$Q = \sum_{n}^{i=1} (y_i - \beta_0 - \beta_1 \boldsymbol{X}_{i,1} - \beta_2 \boldsymbol{X}_{i,2} - \cdots - \beta_d \boldsymbol{X}_{i,d})^2. \tag{4.12}$$

The least squares estimators are the values that makeup $\vec{\beta}$ and minimize Q (and consequently $\vec{\epsilon}$). Let $\vec{b}$ be the vector of the least squares estimated coefficients:

$$\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \end{bmatrix}. \tag{4.13}$$

Then the least squares normal equations of eq. (4.9), with $(\vec{\epsilon}) = 0$, can be written as

$$\boldsymbol{X}^\top \boldsymbol{X}\vec{b} = \boldsymbol{X}^\top \vec{y}. \tag{4.14}$$

Then $\vec{b}$ can be isolated and expressed as

$$\vec{b} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \vec{y}. \tag{4.15}$$

While the inverse of $\boldsymbol{X}^\top \boldsymbol{X}$ is here simple denoted as $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ in reality this can be a computationally expensive operation [39]. Furthermore, invertibility of $\boldsymbol{X}^\top \boldsymbol{X}$ is not always guaranteed. Searching for $\vec{b}$ in this way can thus be costly and time consuming. $\vec{b}$ can also be determined by minimizing eq. (4.12) with for example a gradient descent approach.

4.3 Learning Vector Quantization

Learning Vector Quantization (LVQ) was introduced in 1986 by Kohonen [40] and it is akin to the Self Organizing Map (SOM) [41]. It is a prototype-based supervised classification algorithm. Prototype-based tells us the algorithm employs prototypes. In LVQ, one or more prototypes represent a class in the dataset, and thus a class label is associated with each prototype. Two or more prototypes are allowed to have the same label, but each class needs to be represented by at least one prototype.

Supervised classification is one of the most common forms in machine learning [42]. It is the practice of giving a sample to a learner and knowing the associated class beforehand. The learner will give an answer based on the sample and the current state of its model. Then with this answer and the beforehand known answer an appropriate action is taken to alter the model. The goal is to find a model which will label any sample from the dataset with the correct label.

In order to determine to what class a sample belongs i.e., to classify, LVQ combines the prototypes with a distance measure. The prototypes of LVQ are associated with a class and live in the feature space such that a distance can be determined between the prototype and a sample. This distance can be interpreted as a similarity i.e., a smaller distance means two points are more similar whereas a larger distance means two points are less similar. The distance from the sample is calculated to all the prototypes in the model, and the sample is assigned the label of the prototype with the smallest distance between it and the sample.

The classification scheme that is employed by LVQ is closely related to the intuitive KNN section 4.1 classifier. However, the locations of the prototypes in LVQ are not known by forehand. Whereas in KNN each data point in the known data can be seen as a prototype. LVQ needs to be trained, which is the process of moving the prototypes around in the feature space to find some optimal location for the prototypes. When LVQ is trained, it does not need the entire dataset to classify a novel sample, and it only needs its prototypes. This also means it needs less computational effort than the KNN classifier. A potential drawback, however, is when new data is introduced, LVQ

needs to be retrained, a new optimal location needs to be found for its prototypes, where KNN has no training phase and can be used instantly. Despite this, LVQ classification does scale better than KNN because the number of prototypes is generally less. Classification only depends on the prototypes. Because of this fast nature, LVQ is well suited for situations where there is limited computing power or speed is an issue e.g. as web-service.

In other popular classifiers like the Support Vector Machine (SVM) [43], Multi-Layer Perceptron (MLP) [44, 45], and the more complex Deep Learning architectures (DL) [46, 47] it is hard to explain or justify certain classifications due to the black box nature of these methods. However, in LVQ it is intuitively clear and often visually explainable why a prototype represents a sub-fold of the data and consequentially why a sample is classified as a particular class. Because of this intuitiveness, the LVQ algorithm has gained some attraction in the scientific world, [48–50].

Due to these aforementioned features the LVQ model becomes an attractive classifier for real world applications as well. For example in image analysis [51–53], the medical field [54–57], the industrial field [58–61], and in the financial sector [62], to name a few.

In the following sections, we will discuss the different extensions on the LVQ model.

4.3.1 Learning Vector Quantization 1 and 2.1

In section 4.3, early versions of LVQ are described. Here the basic idea of LVQ will be introduced as well as the shared terminology between the different variations of the algorithm. Let us start by defining a dataset of the size N

$$\mathcal{D} = \{(\vec{x}_i, y_i) | \vec{x} \in \mathbb{R}^d, y_i \in \{1, \cdots, C\}\}_{i=1}^{N}. \tag{4.16}$$

Here $\vec{x}$ is a data point and y is its associated label, with y in the set C of mutually exclusive classes. The number of features (dimensions) of the dataset is denoted by d. With this dataset, we can define the set of prototypes ($\mathcal{W}$) of size P that live in the same space as the data points.

$$\mathcal{W} = \{(\vec{w}_j, c(w_j) = y_j) \in \mathbb{R}^d \times \{1, \cdots, C\}\}_{j=1}^{P}. \tag{4.17}$$

The prototypes are associated with a class, the function $c(\vec{w})$ returns the class label of a prototype. The number of prototypes must be equal to or larger than the number of classes such that every class is represented by at least one prototype. In the original paper [40], Kohonen refers to $\mathcal{W}$ as the cookbook.

The LVQ algorithm has two distinct phases, a classification phase which is generally the same for all variations of the algorithm. A novel data point $\vec{x}$ is assigned a label by the classifier through finding the nearest prototype. Finding the nearest prototype is done with an appropriate distance measure.

When the nearest prototype is found, its label is assigned to the data point. This practice creates so-called receptive fields in the data that are subsets of the input space. These fields have the prototypes as their centers if one prototype per class is used. The chosen distance measure defines the sizes, shapes, and boundaries of these fields. A popular distance measure is the Euclidean distance measure, which is a special case of the general Minkowski distance. The distance measure is not necessarily a global choice, different classes or prototypes can have different distance measures depending on what works well with the dataset.

Before classification can take place, the model needs to be optimized, the so-called training phase. Generally, in the training phase, the prototypes are moved around in the input space until they converge on some optimal local position. The end position of the prototypes depends on their initial state, the distance measure, the update step, and the data itself. The number of prototypes is a hyperparameter and must be determined before training. There have been many variations of LVQ; here we will describe two early versions by Kohonen. The first version (LVQ 1) and an improved version (LVQ 2.1). The latter was later improved and extended further by others to finally become the version used in this study. These versions will also be described in their sections. Below the LVQ1 algorithm is shown.

Algorithm 1: The LVQ 1 algorithm

Data: $\mathcal{D}$, $\mathcal{W}$, α, maxIteration
Result: $\mathcal{W}$
initialize each $\vec{w}$ appropriately;
while Current iteration $\leq$ maxIteration do
    Select a training sample $\vec{x}_i$ at random;
    Find the nearest prototype w_J to $\vec{x}_i$;
    if $c(w_J) = y_i$ then
        $w_J \leftarrow w_J + \alpha(\vec{x}_i - w_J)$;
    else
        $w_J \leftarrow w_J - \alpha(\vec{x}_i - w_J)$;
    end
end

Here α is an appropriate learning rate and can be static or annealed over time [63]. The winning prototype w_J is updated towards the data point if the labels are the same and repelled from the data point if the labels differ. This process is meant to move the prototypes to the general class regions they represent.

LVQ 2.1 is an updated version of LVQ1 and was devised to more efficiently approximate the Bayesian decision boundaries. It does this by selecting two winning prototypes w_J and w_K nearest to a training sample. The two near-

est prototypes are updated if they have different class labels, and the closest prototype has the same label as the training sample $\vec{x}$:

$$w_J \leftarrow w_J + \alpha(\vec{x} - w_J), \tag{4.18}$$
$$w_K \leftarrow w_K - \alpha(\vec{x} - w_K). \tag{4.19}$$

Additionally an active window is defined. This active window has a predefined width through the hyperparameter ω. The training sample is required to fall in this window and is defined by

$$\min\left(\frac{d(\vec{x}, w_i)}{d(\vec{x}, w_j)}, \frac{d(\vec{x}, w_j)}{d(\vec{x}, w_i)}\right) > s, \text{ with } s = \frac{1-\omega}{1+\omega}. \tag{4.20}$$

The window is introduced to ensure convergence, however especially in unbalanced datasets, and divergence can still be a problem [63].

4.3.2 Generalized Learning Vector Quantization

In 1996 Sato and Yamada introduced Generalized Learning Vector Quantization (GLVQ) [64]. It is based on LVQ 2.1 (subsection 4.3.1), but unlike its predecessor, it minimizes a cost function to ensure the prototypes continue approximating the class distributions, [64]. Like LVQ 2.1 GLVQ selects two 'winning' prototypes, $\vec{w}_J, \vec{w}_K$ that are nearest to a labeled sample $\vec{x}$. Here $\vec{w}_J$ is the nearest prototype with the same label as $\vec{x}$ and $\vec{w}_K$ the nearest prototype with a different label. Sato and Yamada described GLVQ in terms of gradient descent, the approach here and later extensions on GLVQ will also be described in terms of gradient descent. The squared Euclidean distances from $\vec{x}$ to $\vec{w}_J$ and $\vec{w}_K$ are denoted as d_J and d_K. With these distances, the authors define a relative distance μ,

$$\mu(\vec{x}) = \frac{d_J - d_K}{d_J + d_K}. \tag{4.21}$$

Here $-1 < \mu(\vec{x}) < 1$, if $\mu(\vec{x})$ is smaller than 0 $\vec{x}$ is classified correct and incorrect when it yields a value larger than 0. Thus when all samples are correctly classified it results in only negative values. With this the authors of [64] define an energy function that should be minimized:

$$E = \sum_{i=1}^{N} \Phi(\mu(\vec{x}_i)). \tag{4.22}$$

The function, $\Phi(\mu)$, is monotonically increasing and N represents the total number of samples. In the GLVQ paper, Sato and Yamada use the squared

Euclidian distance to derive the learning rules. With the learning rules $\vec{w}_J$ and $\vec{w}_K$ are updated to minimize E. The squared Euclidean distance can be written in vector notation as

$$d_i(\vec{x}, \vec{w}_i) = (\vec{x} - \vec{w}_i)^\top (\vec{x} - \vec{w}_i) \text{ , with } i = J, K, \tag{4.23}$$

with $\vec{x}$ the data-point and $\vec{w}_i$ one of the winning prototypes. In this derivation, the squared Euclidean distance is applied as a global distance, but it is also possible for the prototypes or classes to have an individual distance measure associated with them. The prototypes are updated with a gradient descent method

$$\Delta\vec{w}_i = -\alpha \frac{\partial E}{\partial \vec{w}_i} \text{ , with } i = J, K, \tag{4.24}$$

with α as an appropriate learning rate. The learning rate can either be fixed or changed over time. Then from eqs. (4.23) and (4.24) the update rules for the prototypes are derived to be

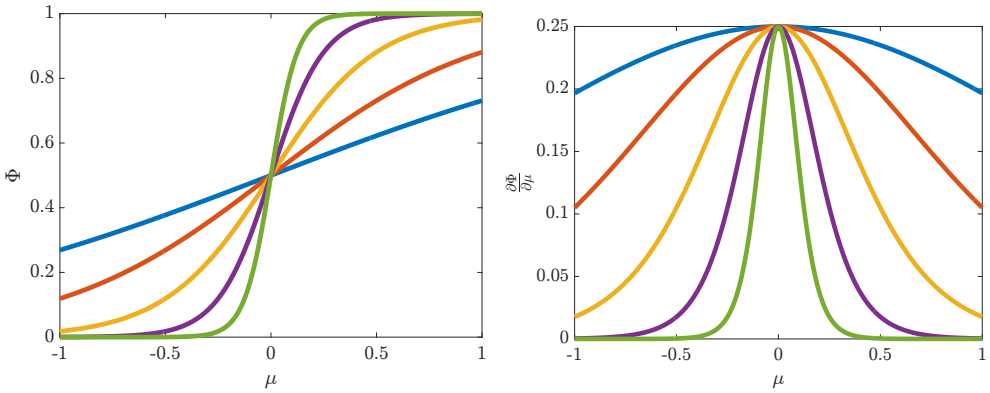
$$\Delta\vec{w}_J = +\alpha \frac{\partial \Phi}{\partial \mu} \frac{d_K}{(d_J + d_K)^2} (\vec{x} - \vec{w}_J), \tag{4.25}$$

$$\Delta\vec{w}_K = -\alpha \frac{\partial \Phi}{\partial \mu} \frac{d_J}{(d_J + d_K)^2} (\vec{x} - \vec{w}_K). \tag{4.26}$$

In eq. (4.25) and eq. (4.26) we can see prototype $\vec{w}_J$ is updated towards the data-point and prototype $\vec{w}_K$ is updated away from the data-point minimizing the energy function. $\partial\Phi/\partial\mu$ acts like an active region around the decision boundaries because the samples around the decision boundaries carry the most information [63]. The authors state in the original GLVQ paper that Φ should be monotonically increasing and use the logistic function: $1/\left(1 + e^{-\mu t}\right)$, with $\partial\Phi/\partial\mu = \Phi\left(1 - \Phi\right)$. Here t denotes the training time. $\partial\Phi/\partial\mu$ has a single peak at $\mu = 0$, this peak becomes narrower as t increases, shown in fig. 4.2. This increases the relative update step size to the prototypes for samples near the decision boundaries over t while decreasing them for samples far away from the decision boundaries.

4.3.3 Relevance Learning in LVQ

In the previous sections, LVQ1, LVQ2.1, and GLVQ were discussed. These methods all make use of a predefined distance measure, usually the Euclidean distance or the squared Euclidean distance (as in (GLVQ, subsection 4.3.2). In these distance measures, the features (dimensions) of the dataset are all treated the same. However, in many datasets, there are statistical regularities



(a) Φ for different values of t where t is steepest around $\mu = 0$

(b) $\partial\Phi/\partial\mu$ for different values of t, where the active region becomes narrower as t increases.

$t = 1$ $t = 2$ $t = 4$ $t = 8$ $t = 16$

Figure 4.2: The monotonically increasing function Φ acting as the active region over the training time t, with $\Phi = 1/\left(1 + e^{-\mu t}\right)$ and $\partial\Phi/\partial\mu = \Phi\left(1 - \Phi\right)$

that can be learned. For example, some features might be correlated or scaled differently or some dimensions might not be essential for classification at all.

The distance measure can be altered by weighting the features to try and overcome the aforementioned difficulties. The weights applied to the dimensions can be learned during the LVQ training phase. The first approach to this method was applied by Bojer et al. [65]. The authors used an adaptive weight vector combined with the squared Euclidean distance to scale the features:

$$d^{\vec{\lambda}}(\vec{x}, \vec{w}) = \sum_{i=1}^{n} \lambda^i (x^i - w^i)^2. \tag{4.27}$$

Here x^i and λ^i denotes the i^{th} element of that vector. The $\vec{\lambda}$ is defined by: $\vec{\lambda} \in \mathbb{R}^n, \lambda^i > 0, \sum_i \lambda^i = 1$. The weight vector $\vec{\lambda}$ is also known as the relevance vector because the weights of the features can be seen as its relevance. The algorithm is therefore appropriately called Relevance LVQ (RLVQ). Later GLVQ was extended by relevance learning by Hammer and Villmann [66]. In this approach it is possible to use a local relevance vector per class or prototype, like a different distance measure per class or prototype in GLVQ, subsection 4.3.2), in the prototype per class variant the squared Euclidean distance is notated as

$$d^{\vec{\lambda}}(\vec{x}, \vec{w}_j) = \sum_{i=1}^{n} \lambda_j^i (x^i - w_j^i)^2. \tag{4.28}$$

The index j denotes the index of the prototype. This distance measure allows for local relevance learning. According to Schneider [63] it is an efficient approach to increase the classification performance of LVQ and it improves on interpretability (which was already a benefit of LVQ through its prototypes, section 4.3) of the resulting model, since the relevances can be used to explain or describe underlying structures in the data. This extension of GLVQ is known as Relevance GLVQ or RGLVQ in short.

4.3.4 Matrix Learning in LVQ

Extending on the idea from the previous subsection 4.3.3, of using a relevance vector $\vec{\lambda}$ to alter the distance measure, matrix LVQ applies a matrix in its distance measure with adaptive relevances. The general distance which is the same as the squared Euclidean distance of eq. (4.23) when $\mathbf{\Lambda}$ is the identity matrix $\boldsymbol{I}$, often seen for this practice is the following quadratic form,

$$d^{\mathbf{\Lambda}}(\vec{x}, \vec{w}) = (\vec{x} - \vec{w})^\top \mathbf{\Lambda} (\vec{x} - \vec{w}). \tag{4.29}$$

This distance resembles the well known Mahalanobis [67] distance where $\mathbf{\Lambda}$ is replaced by the inverse of the covariance matrix $\mathbf{\Sigma}$. The relevance matrix $\mathbf{\Lambda}$ is a $d \times d$ matrix. The matrix accounts for the relevances of the features, like the relevance vector, on the diagonal. However, it also keeps the correlations between features on its off-diagonals. In order for the distance measure to produce meaningful distances the $\mathbf{\Lambda}$ has to be symmetric and positive definite. This can be achieved by using a second matrix $\mathbf{\Omega}$ such that

$$\mathbf{\Lambda} = \mathbf{\Omega}^\top \mathbf{\Omega}. \tag{4.30}$$

Unlike $\mathbf{\Lambda}$, $\mathbf{\Omega}$ can be a rectangular $m \times n$ matrix, as long as $m \leq n$. However in this thesis $\mathbf{\Omega}$ was taken to be a square matrix, i.e., $m = n$. The distance substituted with Ω when combining eq. (4.29) and eq. (4.30) now reads,

$$d^{\mathbf{\Lambda}}(\vec{x}, \vec{w}) = (\vec{x} - \vec{w})^\top \mathbf{\Omega}^\top \mathbf{\Omega} (\vec{x} - \vec{w}). \tag{4.31}$$

$\mathbf{\Omega}$ can be adapted in the training phase of for example LVQ1. For a description of this method and other implementations of matrix learning in LVQ, we refer to [63]. Here we will continue by describing the natural extension to GLVQ with the adaptive relevance matrix $\mathbf{\Lambda}$ and coinciding distance measure, called Generalized Matrix LVQ.

4.3.5 Generalized Matrix Learning Vector Quantization

Generalized Matrix Learning Vector Quantization (GMLVQ) as proposed by Schneider et al. [68] uses the matrix $\mathbf{\Lambda}$ instead of a vector (as in RGLVQ [66]) to map the relevances and the combinations of the relevances between features. This matrix is then adapted during the training phase to optimize classification results. As stated in subsection 4.3.3, in order to gain a meaningful distance measure, $\mathbf{\Lambda}$ has to be semi positive-definite and can be obtained by searching for the matrix $\mathbf{\Omega}$ such that $\mathbf{\Lambda} = \mathbf{\Omega}^\top \mathbf{\Omega}$. Like RGLVQ, GMLVQ is an extension on GLVQ which minimizes an energy function. The energy function of GMLVQ is defined by the authors of [68] as

$$E = \sum_{i=1}^{P} \Phi(\mu^{\mathbf{\Lambda}}(\vec{x}_i)) \text{ , with } \mu^{\mathbf{\Lambda}}(\vec{x}) = \frac{d_J^{\mathbf{\Lambda}} - d_K^{\mathbf{\Lambda}}}{d_J^{\mathbf{\Lambda}} + d_K^{\mathbf{\Lambda}}}. \tag{4.32}$$

Here d_J^{Λ} and d_K^{Λ} denote the squared distance. the distance is influenced by the relevance matrix $\mathbf{\Lambda}$ as in eq. (4.29) and here shown to find the distances for the two winning prototypes w_J and w_K. This distance is then defined by,

$$d_i^{\mathbf{\Lambda}} = (\vec{x} - \vec{w}_i)^\top \mathbf{\Lambda} (\vec{x} - \vec{w}_i), \text{ with } i = J, K. \tag{4.33}$$

Where $\vec{w}_J$ is the closest prototype with the same label as $\vec{x}$ and $\vec{w}_K$ is the closest prototype with a different label as $\vec{x}$. Based on the energy function in eq. (4.32) and this distance measure eq. (4.33) the new update rules for the two winning prototypes w_J and w_K are derived to

$$\Delta\vec{w}_J = \alpha_1 \frac{\partial \Phi}{\partial \mu}(\mu^{\mathbf{\Lambda}}(\vec{x}))\, \mu_J^{\mathbf{\Lambda}}(\vec{x})\, \mathbf{\Lambda}\, (\vec{x} - \vec{w}_J), \tag{4.34}$$

$$\Delta\vec{w}_K = -\alpha_1 \frac{\partial \Phi}{\partial \mu}(\mu^{\mathbf{\Lambda}}(\vec{x}))\, \mu_K^{\mathbf{\Lambda}}(\vec{x})\, \mathbf{\Lambda}\, (\vec{x} - \vec{w}_K). \tag{4.35}$$

In these update rules α_1 is an independent learning rate for the prototypes. $\mu_J^{\mathbf{\Lambda}}(\vec{x})$ and $\mu_K^{\mathbf{\Lambda}}(\vec{x})$ can be derived to

$$\mu_J^{\mathbf{\Lambda}}(\vec{x}) = \frac{4 d_K^{\mathbf{\Lambda}}}{(d_J^{\mathbf{\Lambda}} + d_K^{\mathbf{\Lambda}})^2}, \tag{4.36}$$

$$\mu_K^{\mathbf{\Lambda}}(\vec{x}) = \frac{4 d_J^{\mathbf{\Lambda}}}{(d_J^{\mathbf{\Lambda}} + d_K^{\mathbf{\Lambda}})^2}. \tag{4.37}$$

From the update rules in eq. (4.34) and eq. (4.35), the update rule for the matrix $\mathbf{\Omega}$ is derived to

$$\begin{aligned}\Delta\mathbf{\Omega}_{lm} = &-\alpha_2 \frac{\partial \Phi}{\partial \mu}(\mu^{\mathbf{\Lambda}}(\vec{x})) \\ &\Bigg(\mu_J^{\mathbf{\Lambda}}(\vec{x})\Big((x_m - w_{J,m})[\mathbf{\Omega}(\vec{x} - \vec{w}_J)]_l\Big) \\ &- \mu_K^{\mathbf{\Lambda}}(\vec{x})\Big((x_m - w_{K,m})[\mathbf{\Omega}(\vec{x} - \vec{w}_K)]_l\Big)\Bigg),\end{aligned} \tag{4.38}$$

where α_2 is an independent learning rate from the learning rate α_1 used in the update rules for the prototypes. After each update $\mathbf{\Lambda}$ needs to be normalized (as with $\vec{\lambda}$ in RGLVQ) to prevent the matrix from degenerating, this can be done by dividing all elements of $\mathbf{\Lambda}$ by $\sum_m \mathbf{\Lambda}_{mm}$ and thus enforcing $\sum_m \mathbf{\Lambda}_{mm} = 1$ [69]. This is a generalization for a simple diagonal metric of the normalization in GRLVQ where $\sum_m \lambda_m = 1$ is enforced. [63]

4.3.6 Localized Generalized Matrix Learning Vector Quantization

The Localized GMLVQ (LGMLVQ) works with a more complex model using local matrices either attached to each prototype or in a class-wise manner. It was introduced along with GMLVQ by Schneider et al. in [68]. It is a natural extension on GMLVQ and reminds of the proposed use of a different distance measure per class or per prototype in GLVQ in subsection 4.3.2. In LGMLVQ the localized distance measures are however optimized and acquired during the training phase, not chosen beforehand. The update rules for the closest prototypes with the same and different labels are derived from the global matrix implementation in eq. (4.38) for a local matrix per prototype,

$$\begin{aligned}\Delta\mathbf{\Omega}_{J,lm} = &-\alpha_2 \frac{\partial \Phi}{\partial \mu}(\mu^{\mathbf{\Lambda}}(\vec{x})) \\ &\mu_J^{\mathbf{\Lambda}}(\vec{x})\Big((x_m - w_{J,m})[\mathbf{\Omega}_J(\vec{x} - \vec{w}_J)]_l\Big),\end{aligned} \tag{4.39}$$

$$\begin{aligned}\Delta\mathbf{\Omega}_{K,lm} = &+\alpha_2 \frac{\partial \Phi}{\partial \mu}(\mu^{\mathbf{\Lambda}}(\vec{x})) \\ &\mu_K^{\mathbf{\Lambda}}(\vec{x})\Big((x_m - w_{K,m})[\mathbf{\Omega}_K(\vec{x} - \vec{w}_K)]_l\Big).\end{aligned} \tag{4.40}$$

Here Ω_J denotes the matrix for the winning prototype and Ω_K the matrix for the losing prototype. Each prototype w_i has one matrix Λ_i where $\Lambda_i = \Omega_i^\top \Omega_i$ to preserve positive semi-definitiveness of Λ. If this scheme is employed with

a class-wise matrix, the matrices are considered the same for all prototypes associated with the same class.

In the results section, we only show results from the LGMLVQ method because there was a sizable difference in error rates between LGMLVQ and other LVQ variants.

4.4 Model Validation

Model validation is one of the most essential parts of comparing and testing machine learning models. The methods described in this section are based on the work by Alpaydin and Ethem in chapter 19 of [70].

When validating models, we are concerned with two questions:

1. How can we ensure the expected error of a machine learning model will hold up in a real world application?

2. When comparing multiple learning algorithms, how can we determine if one performs better than the other on a given dataset? Even when these algorithms are different. In our case, how can we compare LVQ section 4.3 with KNN section 4.1 and Multiple Regression section 4.2.

By only looking at the training error of a learning algorithm we cannot compare results because by definition the training errors are always smaller than the error obtained from unseen samples[70]. This holds for the same algorithm with different parameters and different algorithms.

- Randomization: the order in which different runs are performed should be at random. This ensures the results to be independent. In a real-world application, it is not known what the order of given samples will be. In order to simulate this randomization is enforced.

- Replication: implies that the results of an experiment can be replicated. This is done by running the same experiment on different configurations of the dataset. By taking the average and variance over the results of the performed experiments, an accidental good or bad result can be identified.

- Blocking: is applied to regulate variances that might occur during the resampling of the dataset. If different learning algorithms are trained on the same dataset but on different samples sets of this dataset, the sets should be the same for the tested algorithms to exclude a 'bad' batch due to the sampling. This can (unfairly) influence the results of a single learning algorithm. In statistics, when two populations are used, blocking is also known as pairing.

In order to ensure the criteria as mentioned earlier, we divided the datasets into three different subsets: train, validation, and test. The train set will be used to train the learning algorithms, and the validation set will be employed after training to validate the results. This process can be repeated for the different algorithms and different parameters for these algorithms. When a learning algorithm is chosen with the best validation result for this dataset, it should be tested once more against the test set. The test set is independent of the train and validation sets and is utilized only once. This is done to replicate a real-world situation where new data points are introduced to the learning algorithm. The test error should then be considered as the final. After this error is obtained a learning algorithm should not be altered anymore to improve the test error [70].

4.4.1 *K*-fold Cross Validation

In *K*-fold cross validation the dataset X is randomly divided into $K-1$ equal sized parts. Then one part is applied for validation and $K-1$ parts are applied for training. This is called a fold. The process is repeated until all parts are used once for validating. This is visualized in table 4.1. *K*-fold cross validation has two problems. One, the validations set is in most cases smaller than the training set. Two, the different sets overlap by $K-2$ parts.

An extreme variant of *K*-fold is leave-one-out, where one data point is applied as validation, and the rest of the data points is applied for training. This is often employed when there is little data available because it makes the validation set as small as it can be. When leave-one-out is applied on a bigger data, set the compute time grows with the size of the dataset because of the many folds generated [70]. However *K*-fold cross validation does lend itself to parallelization because the learner can be trained and validated on the different folds concurrently.

Table 4.1: *K*-fold cross validation shown for $K = 4$, Here the data is divided into four equal parts and one part is employed for validation in every fold. The T stands for a train part and the V for a validation part. After the folds are created they can be used to validate the effectiveness of a model.

	$Part_1$	$Part_2$	$Part_3$	$Part_4$
$Fold_1$	V	T	T	T
$Fold_2$	T	V	T	T
$Fold_3$	T	T	V	T
$Fold_4$	T	T	T	V

The dataset consists of multiple measure points in 24 different areas containing two types of data namely static features and a sliding window. When applying *k*-fold cross validation the dataset is split into multiple parts. However, simply splitting the dataset into parts would break the temporal order of

the data. Because we chose to use one model for all the 24 areas, the data from the measure points were combined. Then when splitting the data for cross-validation, it was ensured that the different splits contained the full data from the measure points, i.e., split one contained data from measure point one and two, then split two can contain data from measure point three, four and so on. This also means that the test set contained only measure points the models have never seen before.

4.4.2 Performance Measures

When validating a model, it is important to be able to measure the performance of that model. For two-class problems, there are many performance measures available [70]. However, in this study, we are only considered with multi-class problems. We will mainly look at the error rates of the model, i.e., the number of wrongly classified data points divided by all the data points.

Furthermore, we will look at the confusion matrix of the models. The confusion matrix of a model is a matrix of the size $d \times d$ where d is the number of features (dimensions) in the dataset. The columns of this matrix represent the classified labels of the model on a test set. The rows of the matrix represent the true labels of this test set. When a data point is classified 1 is added to the matrix on the true label row and the classified label column. Eventually, the classification rate ($1 -$ error rate) can then be calculated by summing the diagonal of the matrix and dividing this by the sum of the whole matrix.

The confusion gives insight into the number of classes the model is off to the real label and this the interclass errors. Because binning is used and the label represents a range of continues values, the confusion matrix gives us an approximation of the actual error made by the model. However, when the number of bins is sufficiently large, the error made because of the binning will go to zero. Moreover, because we are predicting a real value, a small distance to the real label is less of an error to make than a considerable distance to the real label. For example, when the model classifies a data point to have label three but the actual label is four, the model was wrong, but it was less wrong than if it would have classified the data point as label one. This is, of course, due to the ordering of the labels. Furthermore, because the labels are representations of voltages, we can give an average approximation of the amount of volts the model is wrong.

Chapter 5

Implementation

This chapter will describe and explain the choices made during the implementation of the system. We will elaborate on the used toolkits, the configuration of the toolkits, programming languages, the configuration of the performed experiments, and implementation choices of the models. First, we will elaborate on what programming languages and toolkits were used. Then we will continue with some choices made during the implementation of the LVQ system. This chapter will conclude with a layout of the performed experiments.

5.1 Toolkits and Languages

The project was written in the programming language Python with the SKLEARN kit [71], SciPy [72], and NumPy [73]. This is a library that gives basic functionality for multiple Machine Learning algorithms and models. For the LVQ model, specifically, the library by Jensen was utilized [74]. This library has multiple implementations available for the different versions of the LVQ algorithm. We used the default parameters of this library unless stated otherwise for this project. The library by Jensen is based on a MatLab (programming language) implementation by Biehl et al. [75].

The datasets were stored in a Postgresql database, which is an open-source relational database. To communicate with the database a library called Pandas was used [76]. This library allows for simple connection with multiple types of databases and gives a way of storing the data in the software itself through its very useful data-frames, which can be best described as data tables.

The Cross Validation, as mentioned in section 4.4, lends itself naturally very well for parallelization. We achieved parallelization for this CPU and time-consuming method with the IPython library [77] which lends some tools for this task that are not native to the Python programming language.

The code base for this project is stored in a Git repository. Git is a version management application and widely used in the software industry. The repository can be found on the GitHub website through the following link in the footnote[1].

5.2 Monotonic Function in LVQ

In order to optimize a LVQ model, an appropriate function for Φ must be chosen. Φ was discussed in subsection 4.3.2 and should be a monotonically increasing function. Sato and Yamada used the logistic function in the GLVQ paper [64]. We also applied the logistic function but with a slightly different partial as presented in the original paper,

$$\Phi = \frac{1}{1 + e^{-\beta\mu}}, \tag{5.1}$$

$$\frac{\partial \Phi}{\partial \mu} = \frac{\beta e^{\beta\mu}}{\left(1 + e^{\beta\mu}\right)^2}. \tag{5.2}$$

Here β is a constant and was set to 2 and is not dependent on the time unlike the variable t in the paper by Sato and Yamada.

5.3 Optimization for LVQ

In subsection 4.3.2 it was mentioned that GLVQ, and extensions thereof, are described in terms of gradient descent. A gradient descent is an optimization technique that is often seen in Machine Learning. Especially in Neural Networks where generally some cost function needs minimizing. The idea behind gradient descent is to find the derivative of the function that needs to be minimized and follow the slope of that function to a local or global minimum by adjusting its parameters in that direction. This is done with a predetermined step size schema.

The stochastic gradient descent, an on-line variant, is particularly popular in Machine Learning as described in [78]. However, these methods have some drawbacks, namely the necessary parameter tuning, in particular the step size. The step size is problem dependent and often needs to be searched for by means of cross-validation. This is expensive in terms of time and computing power. Multiple methods have been proposed in the past to automate finding the step size, e.g. Shaul et al. [79].

We applied an optimization method called Limited Memory Broyden-Fletcher-Goldfarb-Shanno (LM-BFGS) [80]. This method uses a line-search and the Hessian matrix to find a local optimum of a function. Furthermore,

[1] https://github.com/jellevanwezel/FinalThesisProject

it does not need a predetermined step size, avoiding this problem altogether. Furthermore this method was already implemented in the applied toolkit.

Chapter 6

Experiments

The experiments are divided into two scenarios. The first scenario consists of a one-step prediction. Here the sliding window addressed in section 3.4 is applied to provide the feature vector, and the classifier is tasked with finding the next bin, section 3.5, the next point is in. This way a prediction is made based on a one-time step.

The second scenario is the longterm prediction scenario. Here a one-step prediction is carried out by the classifier and this prediction is then used as the new value in the window. Consequently, the last value in the window is removed from it, making the window act as a queue. With the new window, the next prediction is carried out, and the process is repeated until a sufficient number of predictions is performed. It should be noted that if there is an error in any prediction in the second scenario, prediction following this first prediction might suffer from it. The static features, the features associated with the pipeline and the soil they lay in, are assumed not to change over the years, hence the name. These features are therefore reused at every time step. This method is based on an approach described by Takens [81] and by Bontempi et al. [82]. We predict ten years since most time series hold about 20 years, 3.1 and we need to keep space for the window.

For cross-validation purposes section 4.4 the data was split into two sets, 80% validation and 20% test. The data was split per measuring point in such a way that every Cathodic Protected area contributed at least one measuring point for the test set. The one-step scenario was carried out on the validation set to optimize the parameters of the models and then tested on the test set. The longterm scenario was optimized as the one step scenario i.e., the models from the one step scenarios were used, and then the longterm method was applied the test set. The optimization itself was performed with 10 fold cross-validation.

As mentioned above the models were optimized for the one step scenario. Model optimization is done through cross-validation on the models with combinations of parameters. This practice results in a model, optimized

with the found parameters that are deemed to work best for a certain dataset. We will now address these parameters for the models and methods that formed the dataset.

The dataset was dependent on the size of the sliding window, the number of coefficients used by the Chebyshev interpolation and the type of label. As mentioned in chapter 3 the label can be chosen to represent the real next value in the sliding window or the gradient to the real next value. The window size was taken to be in the range from 1 to 9. The Chebyshev coefficients were cross-validated separately by looking at the error made by their approximation for the different number of coefficients. The range taken for this is from 1 to 50 coefficients.

The first coefficient in the Chebyshev method represents a constant height of the approximated function. If this coefficient is removed, we obtain a function centered around zero. This can be desirable if we are interested in only the slopes of the time series. We optimized the models with and without the first Chebyshev coefficient.

The binning, addressed in 3.5, was chosen to be 20, uniformly distributed over the data. Since most points are present in the center of the data, the points falling outside a 1 to 13 range were taken to fall in bin 13, since there were no to little data points found above bin number 13.

For Learning Vector Quantization only one parameter was optimized during cross-validation, namely the number of prototypes per class. We took a range of 1 to 15 prototypes per class and will look what model performs best. Standard k Nearest Neighbors is depended on the number of k. We searched in a range from 1 to 20 neighbors. As with LVQ, one could use different distance measures for KNN . However, we applied the Euclidean distance. For Multiple Linear Regression, no additional parameters were searched for.

Chapter 7

Results

In this chapter the results obtained from the models described in chapter 4 and results from the Chebyshev interpolation described in section 3.3 are presented. We will start by looking at the number of coefficients used for the Chebyshev interpolation, then the number of prototypes will be briefly discussed. Finally, we will review the model's performances in two scenarios.

The scenarios are a one-step scenario and a longterm scenario, as described in chapter 6. The models were cross-validated and optimized based on their one step performance. Then the longterm approach was applied based on the best-deemed models in the one step scenario.

7.1 Chebishev Cross Validation

As described in chapter 3, the Chebyshev interpolation depends on the number of coefficients to approximate a function. When the number of coefficients is infinite, the original function can be expressed with this method. However, an infinite number of coefficients makes our overall model too complex. Therefore an appropriate number of coefficients is sought by means of leave-one-out cross-validation. Where one data-point was removed from a time series then to be approximated by the Chebyshev fit, repeated for all data points.

We chose to run this experiment from 1 to 50 coefficients. The number of coefficients can, however, be taken much higher, taking a higher number of coefficients, however, consumes more time and CPU power since this method is applied with a recursive algorithm (section 3.3). It also introduces overfitting of the underlying the function, causing the error after 50 coefficients not to go down.

We chose to run this experiment from 1 to 50 coefficients. The number of coefficients can, however, be taken much higher, taking a higher number of coefficients, however, consumes more time and CPU power since this method is applied with a recursive algorithm (section 3.3). It also introduces overfit-

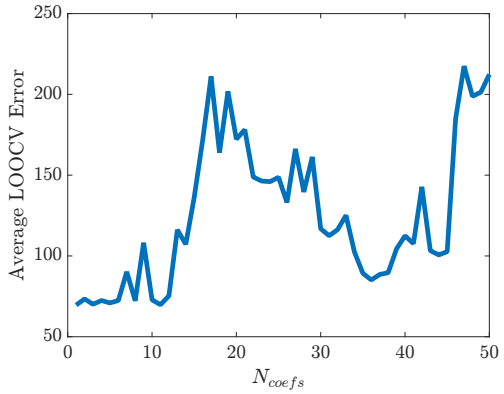


Figure 7.1: The leave one out cross-validation error versus the number of coefficients for the Chebyshev imputation method. The error is the average over 10 runs.

ting of the underlying the function, causing the error after 50 coefficients not to go down.

7.2 Parameter Sweep

On the obtained time series a parameter sweep was conducted for the four applied models. It differed per model what parameters worked well. A table with all the results can be found online in the projects GitHub repository [1]. Here we will show the best results from the parameter sweep and their error rates.

Model	Window Size	Coefficients	First Coefficient	Label	Error rate
LGMLVQ	8	10	Yes	Real	0.04
KNN	3	10	Yes	Real	0.12
MLR	9	9	Yes	Real	0.03

Table 7.1: Resulting parameters and error rates on the validation set from the applied 10 fold cross validation.

The k for k Nearest Neighbors for the results in table 7.1 had a value of 2. Furthermore, both KNN and MLR performed better without the so-called static features. It should also be noted that an error on validation of 0.12 by the KNN model stands out since only a window size of 3 was utilized. Additionally, it stands out that all models performed best with the first Chebyshev coefficient and the real binned data point instead of the gradient.

As mentioned in chapter 6, the number of prototypes per class for LVQ was optimized. The parameter sweep was performed on a one prototype per

[1] https://github.com/jellevanwezel/FinalThesisProject

class model, and then the found parameters were used to find an appropriate number of prototypes. This will be addressed separately in the next section.

7.3 Number of Prototypes for LVQ

With the parameters mentioned in the previous section, the LVQ model was further optimized by finding an appropriate number of prototypes per class. The number of prototypes is ranged between 1 and 15. Fifteen was chosen as the upper boundary because a large number of prototypes per class makes the model more complex.

There are two dips observable in fig. 7.2, One at one prototype and one around eight prototypes. We chose to use these two values because the average error of the two models was low. Furthermore, applying one prototype will result in a simple LVQ model where only one prototype represents every class. However, the standard deviation for the one prototype variant is relatively high. The eight prototype has a higher average error but a smaller standard deviation, however. Additionally, the data, in general, might be better approximated by multiple prototypes per class.

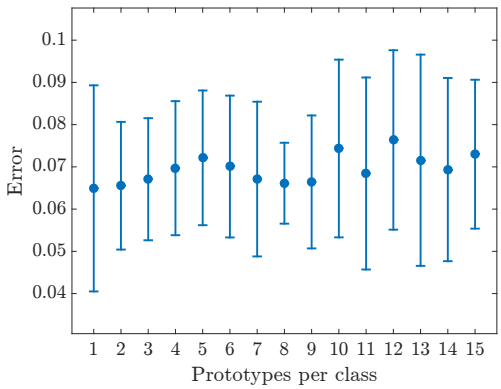


Figure 7.2: The error rates for LGMLVQ with the number of prototypes per class. This is the average error over the 10-fold cross-validation runs, with the standard deviation in the error bars.

Looking at the relevances in fig. 7.3, There is a clear distinction visible between the static features and the features extracted from the sliding window. The barrier between these two feature types is marked by the transition from blue to yellow (at the 10th feature mark). Furthermore, the most relevant features are the features in the sliding window representing the closest year to the actual label. After this first year a drop-off is visible in relevances. What stands out is that the last year is a dark blue column and thus not deemed relevant. Still this model performed better than a model using a 8 year window during the cross-validation. The relevances for the sliding window is most pronounced in the center of the data. The relevances in the outer regions are

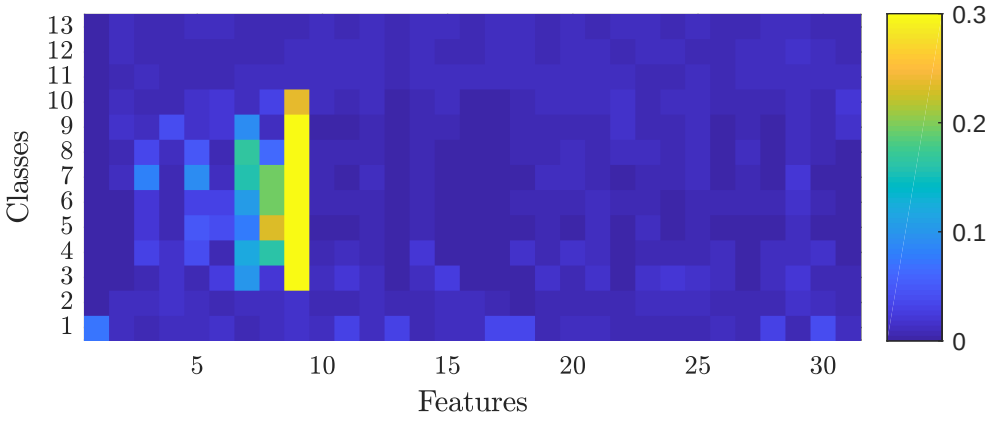
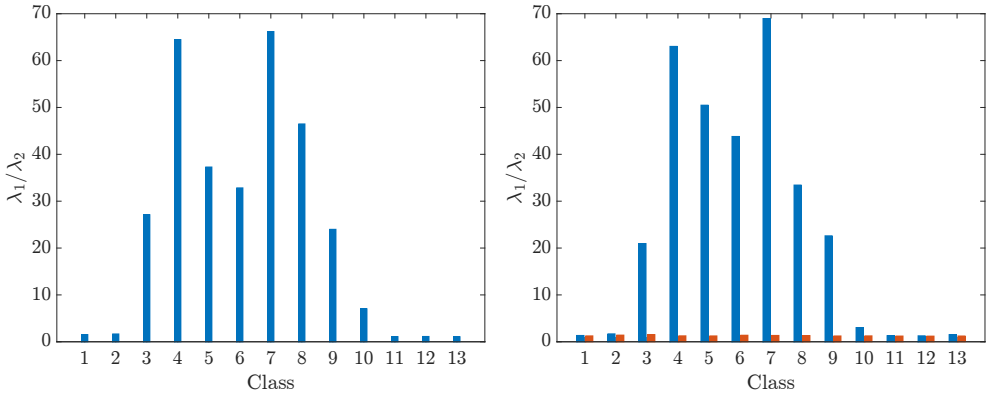


Figure 7.3: The relevances for the LVQ model with 1 prototype. The colors are scaled from 0 (dark blue) to 0.3 (yellow). However, yellow means 0.3 and up to 1.

less pronounced, this is most likely caused by the lack of data points in these regions.

Further analyzing the prototypes is done by examining the eigenvectors and eigenvalues of the matrix Λ applied in the distance measure. Presented in fig. 7.4 is the difference of the two largest eigenvalues of a prototype's matrix. For the one prototype per class, it is observable that the difference for the classes in the 'center' of the data is larger than on the edges. This can be explained by the smaller amount of data points in these regions causing the prototypes for these classes to be undertrained.

In the eight prototypes per class figure, the two prototypes with matrices containing the most significant difference in eigenvalues are shown. It stands out that one prototype is sufficient, at least for the classes in the center of the data to be described by one prototype. This also explains the low error rate for the one prototype variant. However, the remaining seven prototype might be helpful in describing the decision boundaries more carefully finding a balance between under and overfitting.



(a) λ_1/λ_2 for the relevance matrix Λ of each single prototype (w_1) in the model.

(b) λ_1/λ_2 for the relevance matrix Λ of the two prototypes (w_1, w_2) with the highest difference in eigenvalues in the model.

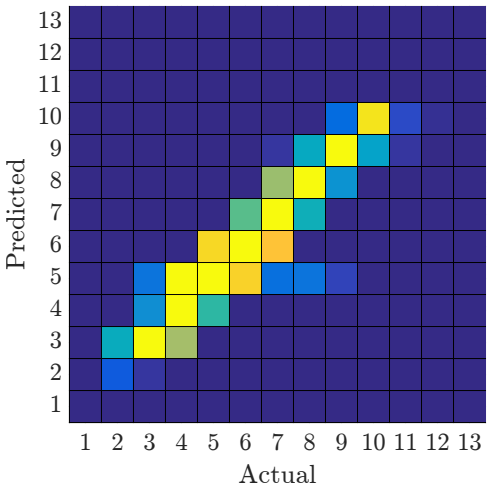
Figure 7.4: The eigenvalues of the relevance matrix $\Lambda = \Omega^\top \Omega$ for a one prototype per class run and an eight prototypes per class run. The eigenvalues of the two most prominent eigenvectors per w and corresponding Λ were taken to show the difference between them per class and prototypes.

7.4 Validation

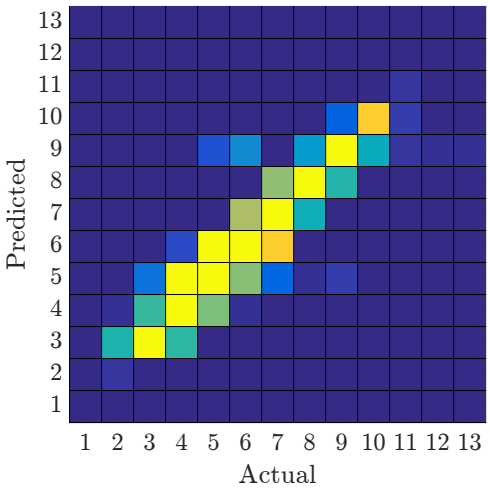
The cross-validation in section 7.3 resulted in 2 LVQ models, one with 1 prototype per class and one uses 8 prototypes per class. Here the confusion matrices are presented to be able to compare the four models better.

From fig. 7.5 it can be observed that MLR performs best since most classifications happen on the diagonal of the confusion matrix. KNN shows the worst performance of the four models and both LVQ models perform about the same. The LVQ model with 8 prototypes does, however, make some mistakes where it misclassifies points as class 9. The 1 prototype variant makes similar mistakes for class 5.

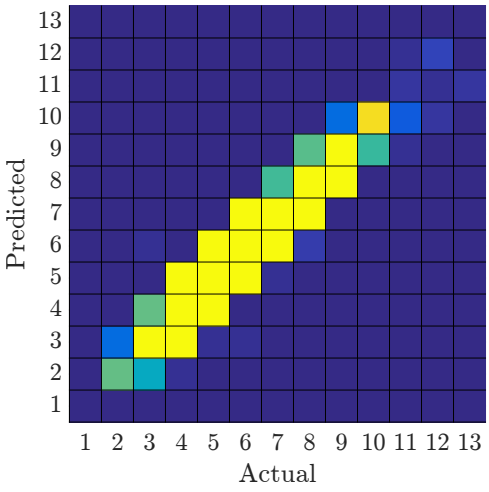
It should be noted that if the classifier is off by one bin, it is still close to the actual value. A misclassification by missing the bin by one is less of an error than missing it by, for example, five bins. All models perform well in that sense since they all are almost one bin of when misclassifying a data point.



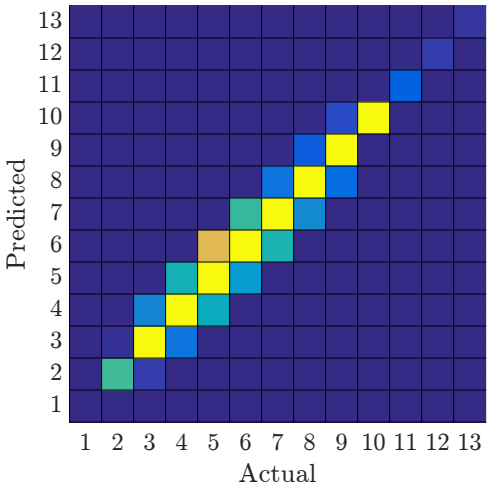
(a) Confusion matrix for Local Generalize Matrix LVQ with one prototype per class. With a windows size of 8



(b) Confusion matrix for Local Generalize Matrix LVQ with eight prototypes per class. With a windows size of 8



(c) Confusion matrix for Local k Nearest Neighbors with a window of size 3 and $k = 2$, without the static features



(d) Multiple Linear Regression, without the static features and a window size of 9

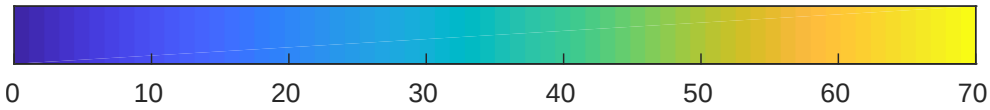


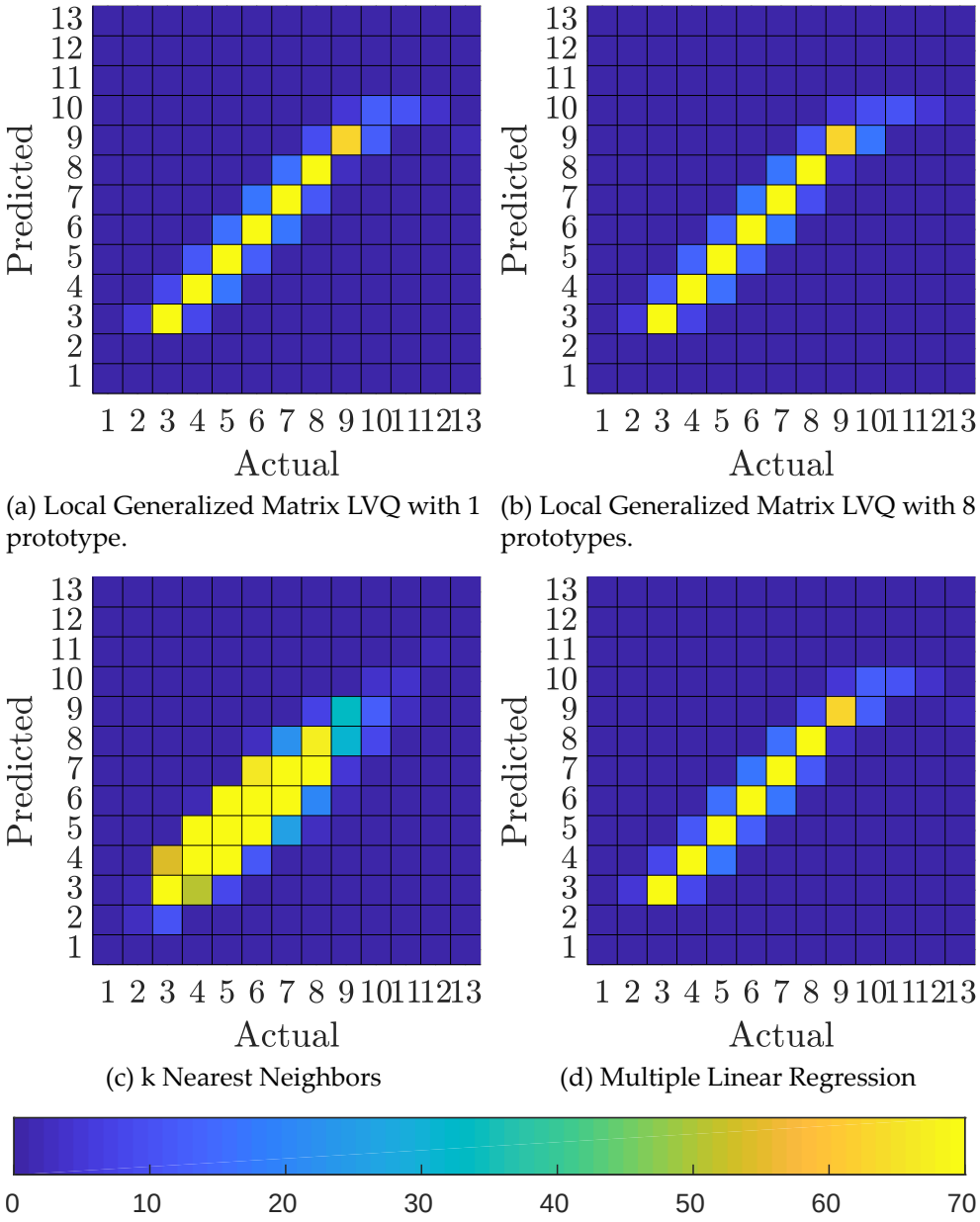
Figure 7.5: The confusion matrices for the different models during the cross validation. These are the means over the 10 folds in the validation set.

7.5 Test

In this section, the results of the test set are presented. The error rates for all four models were higher than the errors in the validation stage, but this is to be expected [70]. The same patterns seen in the validation stage are observed, MLR performs best almost never leaving the diagonal and KNN performs worst but stays close to the diagonal meaning the predictions are not far off. However, in the longterm scenario, this might eventually become a problem since the next prediction is based on the previous one.

LGMLVQ 1	LGMLVQ 8	KNN	MLR
0.08	0.08	0.12	0.03

Table 7.2: Error rates for the four models on the test set.



(a) Local Generalized Matrix LVQ with 1 prototype.

(b) Local Generalized Matrix LVQ with 8 prototypes.

(c) k Nearest Neighbors

(d) Multiple Linear Regression

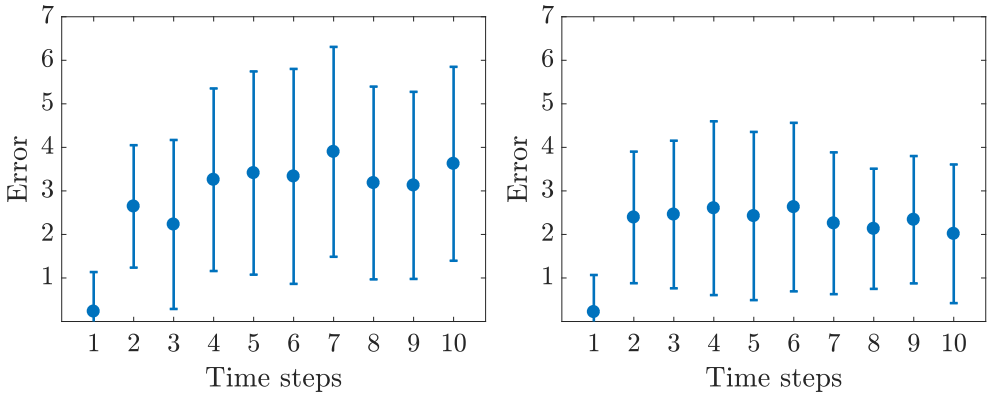
Figure 7.6: The confusion matrices for the different models on the test dataset, this dataset contained approximately 20% of the data, with 83 measure points.

7.6 Longterm results

In this section, the results for the longterm scenario are presented. As described in chapter 6, the test set contains multiple time series for which the labels are known. The predictions are started by taking the first window of the time series and predicting the next 10 points based on that. The first pre-

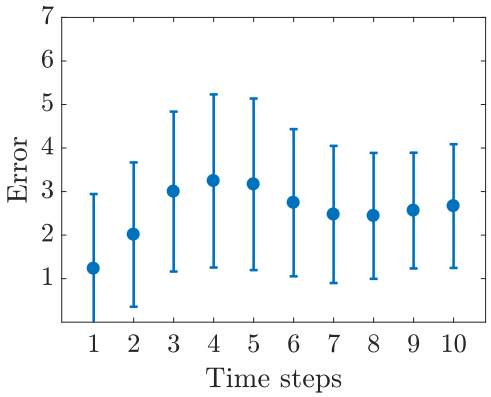
diction is the same as that in the one step scenario. The errors seem to all get higher after the first prediction and then stabilize except for the MLR that seems to keep increasing.

KNN and LVQ with 8 prototypes seem to perform best in this scenario resulting in the smallest bin difference. The 8 prototype LVQ seems to have a small edge on the KNN model since its average bin difference is overall slightly lower.

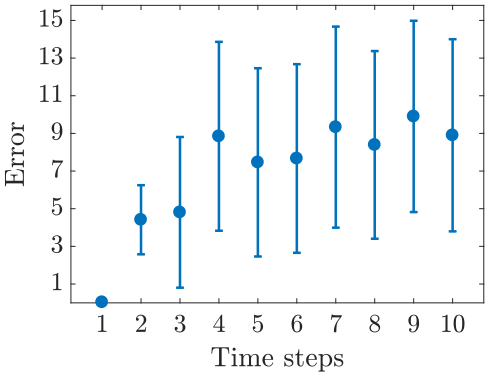


(a) LVQ with one prototype per class

(b) LVQ with eight prototypes per class



(c) KNN



(d) Multiple Linear Regression

Figure 7.7: The longterm results for the four different models. In the above figures the means and standard deviation per time step over the test set are shown. The error indicates the number of bins the classifier was wrong i.e., the distance to the real bin.

Chapter 8

Discussion

In this chapter, we will discuss our data, method and results. Steel pipelines are protected from corrosion by a method called Cathodic Protection. This method applies a current to the pipelines making them cathode. The pipelines should carry a voltage of -850 mV for them to be fully protected from corrosion. Coteq, a local Transmission Systems Operator (TSO) in The Netherlands obtained a dataset containing measurements of these voltages. With this information and secondary data from other sources, it should be possible to optimize a Machine Learning model in order to predict the voltages in the pipelines for the coming years. Predictions of the voltages can then help prevent problems stemming from the subsequent corrosion.

Our research partially answered the posed question in the sense that the applied Learning Vector Quantization model was capable of making accurate predictions in the one-step scenario, but a significant error was introduced after this first time-step in the multi-step scenario. Learning Vector Quantization was however not the only model suffering from this problem since all tested methods showed a significant drop off in performance in the longterm scenario.

We applied a Learning Vector Quantization model to see if this model would be capable of making the predictions as mentioned earlier. The main reason for choosing this model was because a similar approach on a different dataset by Hammer et al. [12] showed promise. Furthermore, other research on this approach is scarce and therefore interesting to investigate.

It was surprising to see how well the KNN model performed in the one-step scenario, compared to the more complex LVQ model. Especially because KNN operated on a relatively small window and without the use of the static feature data. This suggests two things. Namely, the problem posed in the one-step scenario is not a hard problem since it can be solved, to a certain degree, with a relatively simple model, and the static features did not contribute much to the predictions. The excellent performance of the MLR in the one-step scenario contributes to this thought.

The performance of KNN in the one-step scenario was based on the two nearest neighbors, resulting in a local classification. The LVQ models were tested with a 1 prototype per class model and an 8 prototype per class model. The 1 prototype and 8 prototypes performed about the same in the one-step scenario. However, in the longterm prediction scenario, the 8 prototype LVQ model and the KNN model both performed slightly better than the 1 prototype LVQ model. This suggests the solution to the overall problem posed in the longterm scenario, benefits from a more local based classification.

The 8 prototype per class LVQ model slightly outperformed the KNN model in the longterm scenario. The 1 prototype per class LVQ model tries to classify a novel data point on this one prototype resulting in a rather global classification, the reason it still performed only slightly worse than the KNN and the 8 prototype LVQ might be because of the local distance measure it optimizes, resulting in semi-local classification with only the one prototype. The 8 prototype LVQ benefits as well from this same feature, where it optimizes a distance measure for each of its prototypes, only classification is more local because it employs multiple prototypes resulting in a kind of hybrid between KNN and the 1 prototype LVQ, resulting in the best performance of the tested models in the longterm scenario.

Multiple Linear Regression performed well in the one step scenario but fell off in the multi-step scenario. This is likely due to the method with which the regression is performed. The regression fits a hyperplane on the data and does not have any curvatures but does have an angle. This angle is then reinserted in the window for the next classification which yields a worse result because of it.

In order to be able to compare MLR to LVQ and KNN, the result of MLR needed to be binned, because MLR itself yields a continues value while LVQ and KNN yield discreet values. This caused the result of every MLR time step to be generalized to a specific bin, then the middle value of the bin was inserted back into the window, introducing an error in the prediction. However, this mid value was also inserted into the window of the other tested methods, but their performance was impacted less. Still, it would be interesting to see what MLR is capable of when the real values were used and binned only after all classification steps are performed.

The longterm results for MLR, however, seem to show a wave-like pattern, this pattern can be seen in all the figures in the longterm scenario results. The pattern can be explained by the waves in the data itself and the classifiers not following this wave precisely causing them to be closer at one time-step and further away at another.

The dataset of the voltage measurement consisted of incomplete and inconsistent time series. To amend this problem, we applied Chebyshev interpolation to gain consistent evenly spaced data points. In order to validate the method itself cross-validation was used to determine the number of Chebyshev coefficients. Even though the fit this method provided was

deemed sufficient for this work, other methods might have yielded better results, e.g. Fourier based or simple linear interpolation. Furthermore, the measure points in the dataset had about 20 measurements each, with one measurement per year. This means contained contained a history of 20 years. To be able to do long-term predictions based on only 20 years becomes difficult at some point because there is not enough history to test and train on.

Chapter 9

Conclusions

The ability to predict voltage in Cathodic Protected steel pipelines can help prevent corrosion of these pipelines. In this study, we showed that predicting these voltages is possible in a short-term scenario. All four of the tested models performed well, where the worst model (KNN) had a classification rate of 88% in this scenario and the best model had a classification rate of 97% (MLR). The LVQ model performed slightly worse than the MLR with a classification rate of 96%.

Non of the models was able to transfer these good classification results from the one-step scenario to the longterm scenario. However, in this scenario, the LVQ model using 8 prototypes per class performed best. This model had a worst case error of three bins as shown in section 7.6, the model can, however, still be used by Coteq and ValueA to predict the voltages in the pipelines, given it has the mentioned error margin.

Future work on this study might include different types of interpolation and different models. As addressed in the introduction 1.1, LSTM-Neural-Networks have made some significant developments in recent years and might fit well on the problem posed here. However, these networks, as well as other deep learning methods, tend to become overly complex. Another way of predicting the voltages can also come through an auto-correlation approach or a nonlinear regression.

As mentioned in chapter 8 the dataset contained a relatively short period of time the measurement were performed in. It would be interesting to be able to do the predictions, with the in this thesis applied approach, on a dataset stretching over a longer period of time. This would allow more data to be extracted from the dataset and for new patterns to be learned by the used models, which in turn might lead to better long-term results.

Bibliography

[1] "Gasveld groningen." https://goo.gl/cEroz6, 2014. Accessed: 25-04-2018, URL shortened.

[2] "Our history." `https://www.nam.nl/over-ons/onze-historie.html`. Accessed: 25-04-2018.

[3] "Mijnwet 1903." http://wetten.overheid.nl/BWBR0001870/1999-02-17, 1903. Accessed: 10-05-2018, Dutch Law.

[4] J. Schenk and P. Timmer, Groningen-gasveld vijftig jaar. Stationsweg 66, 7941HG, Meppel, The Netherlands: Uitgeverij Boom, 1 ed., January 1999.

[5] B. Botter, "Gas in the netherlands: The vital combination of many small fields and a global giant," in AAPG European Region Annual Conference, pp. 11–12, AAPG European Region Annual Conference, 2009.

[6] M. Brandsma and H. Ekker, De gaskolonie. Boterdiep 117, 9712LM, Groningen, The Netherlands: Uitgeverij Passage, 1 ed., April 2016.

[7] A. Savaresi, "The paris agreement: a new beginning?," Journal of Energy & Natural Resources Law, vol. 34, no. 1, pp. 16–26, 2016.

[8] J. Gerdes, S. Marbus, and M. Boelhouwer, "Energietrends 2016," 9 2016. Dutch, [Energy Trends 2016].

[9] G. Dorffner, "Neural networks for time series processing," Neural Network World, vol. 6, pp. 447–468, 1996.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] H. Kantz and T. Schreiber, Nonlinear time series analysis. Cambridge university press, 2 ed., 2004.

[12] M. Strickert, T. Bojer, and B. Hammer, "Generalized relevance lvq for time series," in Artificial Neural Networks — ICANN 2001 (G. Dorffner,

H. Bischof, and K. Hornik, eds.), (Berlin, Heidelberg), pp. 677–683, Springer Berlin Heidelberg, 2001.

[13] E. N. Lorenz, "Deterministic nonperiodic flow," Journal of the atmospheric sciences, vol. 20, no. 2, pp. 130–141, 1963.

[14] E. Haerani, L. Apriyanti, and L. Wardhani, "Application of unsupervised k nearest neighbor (unn) and learning vector quantization (lvq) methods in predicting rupiah to dollar," in 2016 4th International Conference on Cyber and IT Service Management, pp. 1–4, April 2016.

[15] M. Poulos and S. Papavlasopoulos, "Automatic stationary detection of time series using auto-correlation coefficients and lvq," in IISA 2013, pp. 1–4, July 2013.

[16] O. R. de Lautour and O. Piotr, "Nearest neighbor and learning vector quantization classification for damage detection using time series analysis," Structural Control and Health Monitoring, vol. 17, no. 6, pp. 614–631.

[17] K. Pesinis and K. Fah Tee, "Statistical model and structural reliability analysis for onshore gas transmission pipelines," Engineering Failure Analysis, vol. 82, pp. 1 – 15, 2017.

[18] Q. C. and M. Orazem, "A weighted nonlinear regression-based inverse model for interpretation of pipeline survey data," Electrochimica Acta, vol. 49, no. 22, pp. 3965 – 3975, 2004. The role of electrochemistry in the sustained development of modern societies.

[19] L. Meyer, Predicting corrosion on protected buried steel natural gas distribution pipelines. MIT, 2016.

[20] M. S. El-Abbasy, A. Senouci, T. Zayed, F. Mirahadi, and L. Parvizsedghy, "Artificial neural network models for predicting condition of offshore oil and gas pipelines," Automation in Construction, vol. 45, pp. 50 – 65, 2014.

[21] D. M. Abraham and R. Wirahadikusumah, "Development of prediction models for sewer deterioration," in Proceedings of the Eight International Conference on Durability of building materials and components, Ottawa, 1999.

[22] T. Andersen, A. Misund, et al., "Pipeline reliability: an investigation of pipeline failure characteristics and analysis of pipeline failure rates for submarine and cross-country pipelines," Journal of Petroleum Technology, vol. 35, no. 04, pp. 709–717, 1983.

[23] J. Mandke, "Corrosion causes most pipeline failures in gulf of mexico," Oil and Gas Journal;(USA), vol. 88, no. 44, 1990.

[24] N. R. Council, Improving the Safety of Marine Pipelines. Washington, DC: The National Academies Press, 1994.

[25] D. Humphry, "On the corrosion of copper sheeting by sea water, and on methods of preventing this effect; and on their application to ships of war and other ships," Philosophical Transactions of the Royal Society of London, vol. 114, pp. 151–158, 1824.

[26] M. Kutz, Handbook of Environmental Degradation of Materials. 13 Eaton Avenue, Norwich, NY 13815: William Andrew Publishing, 1 ed., 7 2005.

[27] D. A. Jones, Principles and prevention of corrosion. Upper Saddle River, NJ: Prentice Hall, 1996.

[28] C. Wagner and W. Traud, "The interpretation of corrosion phenomena by superimposition of electrochemical partial reactions and the formation of potentials of mixed electrodes," Z. Elektrochem, vol. 44, pp. 391–402, 1938.

[29] K. BV, Kathodische Bescherming, ondergrondse buisleidingen. Bamendaweg 62, 3319 GS, Dordrecht, Netherlands: Ingenieursbureau Klink BV, 1 ed., 3 2011.

[30] A. Peabody, Control Of Pipeline Corrosion. 1440 South Creek Drive Houston, Texas 77084: NACE International, 2 ed., 11 2001.

[31] F. Melchert, U. Seiffert, and M. Biehl, "Functional approximation for the classification of smooth time series," Machine Learning Reports, vol. 04, April 2016.

[32] F. Melchert, U. Seiffert, and M. Biehl, "Polynomial approximation of spectral data in lvq and relevance learning," in Workshop on New Challenges in Neural Computation 2015 (B. Hammer, T. Martinetz, and T. Villmann, eds.), Machine Learning Reports, pp. 25–32, Bielefeld University, 10 2015. ISSN:1865-3960 http://www.techfak.uni-bielefeld.de/fschleif/mlr/mlr032015.pdf.

[33] A. Gil, J. Segura, and N. Temme, Numerical Methods for Special Functions. Society for Industrial and Applied Mathematics, 2007.

[34] M. S. Floater and K. Hormann, "Barycentric rational interpolation with no poles and high rates of approximation," Numerische Mathematik, vol. 107, pp. 315–331, Aug 2007.

[35] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, vol. 13, pp. 21–27, January 1967.

[36] R. Duda, P. Hart, and D. Stork, Pattern Classification. Wiley, 2nd ed., 2000.

[37] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," The American Statistician, vol. 46, no. 3, pp. 175–185, 1992.

[38] T. M. Mitchell, Machine Learning. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[39] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li, Applied Linear Statistical Models (McGraw-Hill/Irwin Series Operations and Decision Sciences). McGraw-Hill/Irwin, 5 ed., 2004.

[40] T. Kohonen, Self-Organizing Maps. Springer, 1995.

[41] P. Somervuo and T. Kohonen, "Self-organizing maps and learning vector quantization for feature sequences," Neural Processing Letters, vol. 10, pp. 151–159, Oct 1999.

[42] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no. 7553, p. 436, 2015.

[43] B. Scholkopf and A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge, MA, USA: MIT Press, 2001.

[44] P. Werbos, Beyond regression: New tools for predicting and analysis in the behavioral sciences. PhD thesis, Havard University, 07 2018.

[45] G. Cybenko, "Approximation by superpositions of a sigmoidal function," Mathematics of Control, Signals and Systems, vol. 2, pp. 303–314, Dec 1989.

[46] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[47] Y. Bengio, "Learning deep architectures for ai," Found. Trends Mach. Learn., vol. 2, pp. 1–127, jan 2009.

[48] D. Nova and P. A. Estévez, "A review of learning vector quantization classifiers," Neural Comput. Appl., vol. 25, pp. 511–524, Sept. 2014.

[49] M. Kaden, M. Lange, D. Nebel, M. Riedel, T. Geweniger, and T. Villmann, "Aspects in classification learning-review of recent developments in learning vector quantization," Foundations of Computing and Decision Sciences, vol. 39, no. 2, pp. 79–105, 2014.

[50] T. Villmann, A. Bohnsack, and M. Kaden, "Can learning vector quantization be an alternative to svm and deep learning? - recent trends and advanced variants of learning vector quantization for classification learning," Journal of Artificial Intelligence and Soft Computing Research, vol. 7, 01 2017.

[51] S. Li, Z. Wang, and J. Wang, "Study of face orientation recognition based on neural network," International Journal of Pattern Recognition and Artificial Intelligence, 2018.

[52] M. Biehl, R. Breitling, and Y. Li, "Analysis of tiling microarray data by learning vector quantization and relevance learning," in Intelligent Data Engineering and Automated Learning, pp. 880–889, Springer, Springer LNCS, 12 2007.

[53] M. Biehl, P. Pasma, M. Pijl, L. Sánchez, and N. Petkov, "Classification of boar sperm head images using learning vector quantization.," in European Symposium on Artificial Neural Networks, pp. 545–550, 2006.

[54] L. Reinhardt, R. Vesanto, J. Montonen, T. Fetsch, M. Makijarvi, G. Sierra, and G. Breithardt, "Application of learning vector quantization for localization of myocardial infarction," in Proceedings of 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, vol. 3, pp. 921–922 vol.3, Oct 1996.

[55] C. Markopoulos, P. Karakitsos, E. Botsoli-Stergiou, A. Pouliakis, A. Ioakim-Liossi, K. Kyrkou, and J. Gogas, "Application of the learning vector quantizer to the classification of breast lesions.," Analytical and quantitative cytology and histology, vol. 19, no. 5, pp. 453–460, 1997.

[56] P. Karakitsos, B. Cochand-Priollet, A. Pouliakis, P. Guillausseau, and A. Ioakim-Liossi, "Learning vector quantizer in the investigation of thyroid lesions.," Analytical and quantitative cytology and histology, vol. 21, no. 3, pp. 201–208, 1999.

[57] R. van Veen, "Analysis of missing data imputation applied to heart failure data," Master's thesis, University of Groningen, the Netherlands, 2016.

[58] J. Liu, Y. Liang, and X. Sun, "Application of learning vector quantization network in fault diagnosis of power transformer," in 2009 International Conference on Mechatronics and Automation, pp. 4435–4439, Aug 2009.

[59] W. Nianfang, W. ZHANG, and L. Kelei, "Fault identification of high frequency current ripple of fuel cell based on learning vector quantization neural network," in 2nd International Seminar on Applied Physics, Optoelectronics and Photonics, no. 2, pp. 395–401, 2017.

[60] G. R. Lloyd, R. G. Brereton, R. Faria, and J. C. Duncan, "Learning vector quantization for multiclass classification: application to characterization of plastics," Journal of Chemical Information and Modeling, vol. 47, no. 4, pp. 1553–1563, 2007.

[61] V. K. Gupta, J. G. Chen, and M. B. Murtaza, "A learning vector quantization neural network model for the classification of industrial construction projects," Omega, vol. 25, no. 6, pp. 715 – 727, 1997.

[62] K. Sookhanaphibarn, P. Polsiri, W. Choensawat, and F. C. Lin, "Application of neural networks to business bankruptcy analysis in thailand," International Journal of Computational Intelligence Research, vol. 3, no. 1, pp. 91–96, 2007.

[63] P. Schneider, Advanced methods for prototype-based classification. PhD thesis, University of Groningen, 2010.

[64] A. Sato and K. Yamada, "Generalized learning vector quantization," in Advances in neural information processing systems, pp. 423–429, 1996.

[65] T. Bojer, B. Hammer, D. Schunk, and K. T. von Toschanowitz, "Relevance determination in learning vector quantization," in ESANN'2001 (M. Verleysen, ed.), pp. 271–276, D-facto publications, 2001.

[66] B. Hammer and T. Villmann, "Generalized relevance learning vector quantization," Neural Networks, pp. 1059–1068, 2002.

[67] P. C. Mahalanobis, "On the generalised distance in statistics," in Proceedings National Institute of Science, India, vol. 2, pp. 49–55, 1936.

[68] P. Schneider, M. Biehl, and B. Hammer, "Adaptive relevance matrices in learning vector quantization," Neural Computation, no. 12, pp. 3532–3561.

[69] P. Schneider, K. Bunte, B. Hammer, and M. Biehl, "Regularization in matrix relevance learning," IEEE Transactions on Neural Networks, pp. 831–840, 2010.

[70] E. Alpaydin, Introduction to Machine Learning. The MIT Press, 2nd ed., 2010.

[71] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, T. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[72] E. Jones, T. Oliphant, P. Peterson, et al., "SciPy: Open source scientific tools for Python." http://www.scipy.org/, 2001–. [Online; accessed <today>].

[73] T. E. Oliphant, Guide to NumPy. USA: CreateSpace Independent Publishing Platform, 2nd ed., 2015.

[74] J. Jensen, "Sklearn lvq." https://mrnuggelz.github.io/sklearn-lvq/modules/api.html, 2017. [Online; accessed 21-July-2018].

[75] M. Biehl, P. Schneider, and K. Bunte, "Lvq toolbox." `http://matlabserver.cs.rug.nl/gmlvqweb/web/`, 2012. [Online; accessed 21-July-2018].

[76] W. McKinney et al., "Data structures for statistical computing in python," in Proceedings of the 9th Python in Science Conference, vol. 445, pp. 51–56, Austin, TX, 2010.

[77] F. Pérez and B. E. Granger, "Ipython: A system for interactive scientific computing," Computing in Science & Engineering, vol. 9, no. 3, pp. 21–29, 2007.

[78] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning.," in ICML (L. Getoor and T. Scheffer, eds.), pp. 265–272, Omnipress, 2011.

[79] T. Schaul, S. Zhang, and Y. LeCun, "No more pesky learning rates," in International Conference on Machine Learning, pp. 343–351, 2013.

[80] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," SIAM Journal on Scientific Computing, vol. 16, no. 5, pp. 1190–1208, 1995.

[81] F. Takens, "Detecting strange attractors in turbulence," Lecture Notes in Mathematics, Berlin Springer Verlag, vol. 898, p. 366, 1981.

[82] G. Bontempi and M. Birattari, "A multi-step ahead prediction method based on local dynamic properties.," in European Symposium on Artificial Neural Networks, pp. 311–316, 2000.

Appendix A

Cathodic Protection Areas

#	Name	P	$\mu(M)$	$\sigma(M)$
1	Deurningen	25	18.24	6.27
2	Denekamp 2	2	16	5
3	Almelo Castello	17	15.06	8.23
4	Almelo Tusveld	90	14.24	8.66
5	Almelo Windmolenbroek	6	21.17	1.86
6	Wierden	53	15.96	7.97
7	Vriezenveen	18	17.56	8.65
8	Almelo Ten Cate	38	20.53	12.55
9	Enter	11	15.91	8.4
10	Tubbergen	44	21.55	6.22
11	Rossum	25	23	5.78
12	Markelo	20	19.35	5.09
13	De Krim	13	24.92	1.33
14	Oldenzaal	46	9.59	6.92
15	Hardenberg	33	19.48	6.4
16	Almelo De Pook	52	15.17	6.7
17	Vroomshoop	41	22.83	4.66
18	Denekamp 2	2	13	8
19	Goor	6	20.83	1.67
20	Almelo Windmolenbroek	8	17.63	3.04
21	Denekamp	24	13.63	6.35
22	Hengelvelde	22	18.18	9.02
23	Delden	15	19.2	5.41
24	Slagharen	6	23	2.83

Table A.1: Here P denotes the number of measure points and M the number of measurements recorded for P.

Appendix B

Static Features

Acid	Ground Water	Stability
High Water Low	Deepest Deep Water High Highest	Stable Reasonably Stable Unstable Water
Ground Type	Pipe	
Sand Sand and Loam Sand and Clay Sand and Peat Peat Loam Water	Coating Length Construction Year	

Table B.1: Features for the static ground dataset