IDEA League

Master of Science in Applied Geophysics

Research Thesis

Automated assessment of slope stability

Application of machine learning to ERT monitoring data and integration with geomechanical models

Feliks Kiszkurno

Delft University of Technology

ETH zürich

Division of Earth Sciences and Geography | RWTH AACHEN UNIVERSITY

BERKELEY LAB

Automated assessment of slope stability

Application of machine learning to ERT monitoring data and integration with geomechanical models

Master of Science Thesis

for the degree of Master of Science in Applied Geophysics

by

Feliks Kiszkurno

06.08.2021

IDEA LEAGUE
JOINT MASTER'S IN APPLIED GEOPHYSICS

Delft University of Technology, The Netherlands
ETH Zürich, Switzerland
RWTH Aachen, Germany

Dated: *August 6, 2021*

Supervisor(s):	Dr. Sebastian Uhlmann
	Prof. Florian Wagner
Committee Members:	Dr. Sebastian Uhlmann
	Prof. Florian Wagner
	Dr. Kees Weemstra

Eidesstattliche Versicherung

KISZKURNO, FELIKS	421969
Name, Vorname	Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit/Bachelorarbeit/~~ Masterarbeit* mit dem Titel

Automated assessment of slope stability: Application of machine learning to ERT monitoring data and integration with geomechanical models

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Adliswil, 06.08.2021	Feliks Kiszkurno
Ort, Datum	Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Adliswil, 06.08.2021	Feliks Kiszkurno
Ort, Datum	Unterschrift

Abstract

Landslides pose a risks to the communities settled in their vicinity. They pose a threat to human lives and significant economical damage to the affected communities. As the scientific community agrees that the climate change will further increase the risks associated with landslides, it is important to develop a reliable, cost effective and widely applicable technique to assess the stability of the potentially unstable slopes. Such a technique could allow to create an early warning systems meant to help evacuate the communities at risk before the landslide event happens. There have been many studies applying different methods of achieving this goal. This study will apply electrical resistivity tomography to investigate the subsurface and use machine learning methods in the form of supervised and unsupervised learning to interpret the measurements. The electrical resistivity tomography is a widely applied technique to investigate the stability of slopes. It can determine the water table in the subsurface, which plays the crucial role in the stability of the slopes. However, the manual interpretation of the electrical resistivity tomography profiles is a time-consuming and cumbersome task. In order to develop a reliable early warning system both the measurements and their interpretation have to be automatized. The first one has been already achieved. There are systems available that can perform electrical resistivity tomography automatically and continuously. In order to automatize the second part, the following research investigated the capability of machine learning to interpret resistivity profiles alongside with various methods of improving the accuracy of the results. The results show that machine learning is capable of performing this task. Some limitations were identified and guidelines for the preprocessing of the data were proposed. Several areas that require further investigation were identified.

August 6, 2021

Acknowledgements

First of all I want to thank my supervisors Dr. Sebastian Uhlemann and Prof. Florian Wagner who provided me with enormous amount of support throughout this project. I would like to thank my friends Karolina Janowska, Henry Holsten and Mona Wang for their moral support and advice. I would like to dedicate this work to my parents Marta and Adam for all their faith in me and constant support no matter what.

Feliks Kiszkurno

06.08.2021

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1-1 Problem description

Landslides are a common and frequent natural hazard that causes destruction and claims lives every year. The monitoring of the landslides and early warning of the population are important issues. In the years 1995-2015, landslides have taken lives of 20 000 people [Wahlstrom M., Guha-Sapir D., 2015], while other studies suggest even higher numbers of almost 60 000 fatalities from 2004 to 2016 [Froude and Petley, 2018]. On average landslides take 800 to 1 000 lives every year [Borgatti and Soldati, 2010]. Those numbers may actually grow in the future as research by [Gariano and Guzzetti, 2016] found that vast majority of researchers confirms a relationship between landslides and climate change. However the exact effect of the climate change on the local risk of a landslide is hard to evaluate.

In order to mitigate the risk posed by landslides, there is a need to actively monitor the areas at risk of a landslide in real time. According to the United States Geological Service (USGS), this is important both in terms of understanding this phenomenon (which would benefit engineers designing structures to minimize the damage caused by landslides or to prevent or limit their impact) and can be integrated with warning systems to help protect the local population [Highland and Bobrowsky, 2008]. If the dynamics of landslide systems are known there is a chance that early signs of the landmass movement will be detected providing residents in the affected area with enough time to evacuate.

There are different methods applied to study landslide areas, such as ground penetrating radar (GPR) [Abolmasov et al., 2013], microseismics [Occhiena et al., 2013] and many other geophysical techniques. Yet, the electrical resistivity tomography (ERT) is perhaps the most frequently used technique due to its sensitivity to changes in soil moisture content [Jongmans and Garambois, 2007; Whiteley et al., 2019]. Monitoring requires continuous data acquisition and automatic operation. This can be achieved with several methods. Some systems are based on the inclinometers installed in boreholes, weather stations [Bednarczyk, 2013], extensiometers installed in cracks [Milenkovic et al., 2013], or global navigation satellite system (GNSS) based systems [Thuro et al., 2013]. While most of those techniques are cheap and easy to deploy, they often only provide measurements of ongoing failure (i.e. active deformation) or provide data at a single point only. This is often not sufficient to investigate the spatio-temporally heterogeneous processes triggering landslides. Hence, geophysical monitoring, and geoelectrical monitoring in particular, are often used to provide time-varying images of subsurface properties that can be used to investigate those complex processes [Perrone et al., 2014; Uhlemann et al., 2017]. Recently, developments in the geophysical hardware design allowed for low-cost, remote deployment of geoelectrical monitoring systems, which can acquire and telemeter data several times a day. While this is enabling monitoring at high spatial and temporal resolution, the amount of data makes manual interpretation challenging. Hence there is a need to develop automated processing techniques that can simplify the data analysis.

The ERT is a commonly used method (for example: [Reci et al., 2013]). It can delineate between high and low resistivity zones [Reci et al., 2013]. It can be used as both an independent method and together with other ones (for example with GPR [Abolmasov et al., 2013]). It is a very good tool to investigate landslides but there are some disadvantages to it too. One of them is complicated process of the data processing and interpretation. The measurement itself can be done automatically and continuously but deriving actionable information from large amounts of monitoring data is cumbersome and time consuming. It often makes the interpretation of the data in real-time impossible with the conventional approaches such as manual processing and interpretation. Because of this the processing and interpretation phases can create a bottleneck in the wide application of the early warning systems. This problem could be solved by automatizing of the data processing and interpretation phases. If the system operators were freed from the tasks related to data processing and manual interpretation and were needed only to supervise and do the quality control of the results it would lower the cost of the warning system and increase its capacity. To address the need for an automated data interpretation, recent studies have investigated the use of machine learning (ML) algorithms for the extraction of actionable information from ERT monitoring data. [Ward et al., 2014] uses ERT with computer vision methods to track flow of a plume of a tracer through the subsurface. It proves that the ERT combined with computer vision tools can track tracers movement through the subsurface in the presence of noise. It is shown that this can be achieved without the need for manual fine tuning of the parameters. ERT can also be combined with clustering methods to delineate various features in the time-lapse ERT measurements [Delforge et al., 2021]. That study includes temporal changes to the investigated profile which includes hydrological features that can vary over time. Both of those studies execute ERT measurements to obtain information about the subsurface and apply ML methods to interpret those measurements.

1-2 Motivation of this thesis

In this project it will be investigated whether those methods (ERT and ML) are applicable to an automatic and independent monitoring of subsurface features (i.e. variations in water table) that affect the stability of slopes. As described above, water table or soil moisture variations in slopes are critical for slope stability. Since the previous works has shown that ERT is capable of providing subsurface thresholds to landslide initiation [Perrone et al., 2014; Uhlemann et al., 2017], Hence, if ML can be used to automatically extract changes in the subsurface, this information could eventually be used to inform hydrological parameters of geomechanical models, which in turn would allow to provide real-time, robust estimates of slope stability assessments. This will further enable the integration of ERT monitoring into landslide early warning systems. To assess the applicability of ML, here I investigate the performance of clustering and classification methods, but also investigate the impact of data preprocessing to the accuracy of the ML prediction.

Chapter 2

Theoretical Background

In this chapter the theory behind the tools used in this project will be explained and discussed.

2-1 Geoelectrical forward modeling and inversion

The majority of the profiles used in this project (both for training and classification) are synthetic. This has several advantages. The most important one is that the true geometry and parameters of layers in the subsurface are known and can be used to evaluate the accuracy of the predictions made by the classifiers. Since there is full control of how the profiles are defined, they can be designed to test how the classifier will perform when confronted with different subsurface scenarios.In order to generate synthetic profiles, two steps were necessary: forward modeling and inversion. They will be discussed in following section. PyGIMLi has been used to perform both of those steps [Rücker et al., 2017]. For the parameters used to create each profile please see Section 3-2-1 and Tables 3-1 and 3-2.

ERT

In this project the ERT is used as a method to survey the subsurface. It is a method commonly used in different areas including not only geophysics, but medicine as well [Dyhoum et al., 2014]. There were many studies successfully applying this method to similar problems (for example [Ward et al., 2014]). It can be applied in diverse environments and is efficient in the terms of data acquisition. [Maurer, 2006].

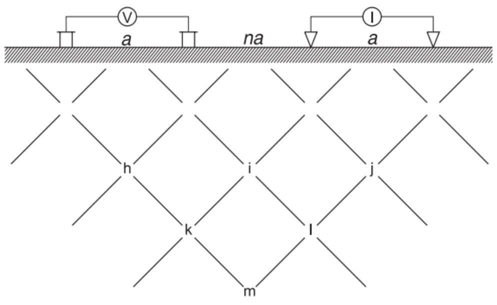


Figure 2-1.1: Dipole-Dipole array Butler, 2005. I - transmitter current; V - received voltage; a - dipole size; na - receiver-transmitter separation (multiple of dipole size); h, i, j, k, l, m - intersections of the grid.

In principle the inversion of the measured voltages can be described with the following equation:

$$\left[\ln\left(\mathbf{V}^{obs}\right) - \ln\left(\mathbf{V}^{\mathbf{ini}}\right)\right] = \mathbf{G}\left[\ln(\boldsymbol{\rho}) - \ln\left(\boldsymbol{\rho}^{\mathbf{mod}}\right)\right] \tag{2-1}$$

where ρ is distribution of the resistivity in the subsurface, V is voltage measured at electrodes (mod - modeled, obs - observed, measured) and $\mathbf{G}$ is the Jacobian matrix [Maurer, 2006].
There are different electrode layouts available. They differ in their properties and can be selected based on requirement of specific survey. The two commonly used layouts are Wenner and Schlumberger [Butler, 2005], but for this project dipole-dipole was selected because it is capable of delivering superior results compared to Wenner array [Chambers et al., 2002]. With this survey layout, both penetration and horizontal resolution are very good [Butler, 2005]. It makes this method suitable for this project as its aim is to investigate the distribution of resistivity in the 2D subsurface profiles. Figure 2-1.1 shows a conceptual illustration of the dipole-dipole array. Two electrodes, spaced by a are used to inject a current, and the resulting potential is measured using a second pair of electrodes that is na from the injection dipole. [Reynolds, 2011]

Resistivity of the subsurface

An ERT survey aims to estimate the resistivity distribution of the subsurface using voltage measurements at the surface or within boreholes. Based on this it allows to delineate the interfaces between layers of different electrical properties (for example between highly resistive sand and conductive water). The exact relationship between different materials that can be found in the subsurface and the resistivity is complex and depends on many variables. There are two ways of how electrical current can be transported through the subsurface and both of them are associated with different physical parameters: electronic process and electrolytic process [Philip Kearey, 2002]. The electronic process depends on electric capabilities of minerals (mostly metals) to conduct electricity by passage of electrons. The electrolytic process depends on porosity of the layer and the pore fluid to transport the ions through it. For the purposes of this project the second one is more important. From the point of view of the process the much more important parameter is the resistivity of the pore fluid (most commonly water) than the parameter of the rock material. Both resistivities have to be recognized in order to delineate between the subsurface material and the part of the subsurface that is infiltrated by water. The presence of water will decrease the resistivity and create a contrast that can be measured by ERT. To some extent a greater contrast will make this task easier as there will be less doubt assigning points to either of those layers. The resistivities of the rock, the pore fluid and the porosity are connected in the following empirical formula:

$$\rho_0 = a \cdot \phi^{-m} * S^{-n} \cdot \rho_f \tag{2-2}$$

It was first introduced by Archie, 1942 and was modified by [W. O. Winsauer, 1952]. ϕ is the porosity of the layer, ρ_f the resistivity of the pore fluid. "m" is a cementation exponent, which is related to the effective grain size (connected with shape of the pores and porosity) [Hubbard, 2005]. The constants "a" and "n" are empirical [Glover, 2016]. There are many papers, publications and research projects discussing the relationship between the pore fluid, the porosity and the matrix material (e.g.: [Bai et al., 2013; Datsios et al., 2017; Lech et al., 2020; Pandey et al., 2015]). This very complex phenomenon plays a limited role in this project as the main contrast in the real data is known to be related to changes in water content, and a lithological change. Additionally, the majority of this thesis focuses on synthetic data. For this project two assumptions are important. The first assumption is that in porous layers infiltration of water will strongly lower the total resistivity of the infiltrated part of the layer, creating an interface between the low and high resistivity areas. This is confirmed by different studies including [Delforge et al., 2021; Goldman, 2010]. The second assumption is that there is a big (orders of magnitude) difference between the conductive water and the resistive matrix material. This is confirmed by many sources for example: [Reynolds, 2011].

2-1-1 Forward Modeling

After the geometry of the profile is defined, a simulation of an ERT survey is performed. The positions of the electrodes can be defined manually and the type of ERT survey can be selected. For details on the used survey configuration please refer to Chapter 3-1-1. Figure 2-1.2a shows the defined geometry of the subsurface for an example profile. Mesh and the environment model used for the forward modeling can be found in the Figure 2-1.2b.

Finite Element

The ERT problems usually do not have a simple, closed form solution or have one only under very restrictive assumptions or requirements [Dyhoum et al., 2014]. Therefore a numerical solution has to be used [Ramirez et al., 2000]. There are two commonly used methods to achieve this: finite elements (FE) and finite differences (FD). FD is more efficient for regular geometry but FE offers more flexibility [Ramirez et al., 2000] which can be beneficial for more complex topography and can improve both the computational efficiency and the quality of result [Günther et al., 2006a; Zienkiewicz et al., 2005].

PyGIMLi uses the finite element method (FE) for forward modelling (FM) [Günther et al., 2006a]. The FE method originates from mechanical problems in which complex structures have been reduced to the elements that behave in a similar way (for example beams) [Igel, 2016]. In this project an irregular, triangular mesh with varying cell sizes is used. It is generated automatically by PyGIMLi based on the specified geometry and mesh quality parameters. Some tests have been performed using a regular grid, but it increased the computational effort significantly and decreased the quality of the result. This is consistent with observation made by [Günther et al., 2006a].

2-1-2 Inversion

PyGIMLi inverts the data using a Gauss-Newton scheme with the misfit defined as [Günther et al., 2006b]:

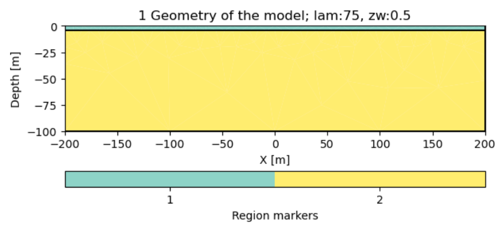
$$\Phi_d(\mathbf{m}) = \sum_{i=1}^{N} \left| \frac{d_i - f_i(\mathbf{m})}{\epsilon_i} \right|^2 = \|\mathbf{D}(\mathbf{d} - \mathbf{f}(\mathbf{m}))\|_2^2 \tag{2-3}$$

In this equation m_j contains physical properties of each cell (in this case it will be the logarithm of the resistivity of each cell), d is vector with data and D is defined as $\mathbf{D} = \operatorname{diag}(1/\epsilon_i)$ where ϵ_i is an error associated with the data. The goal is to minimize ϕ_d by minimizing the difference between the data d and the model response $f(m)$. ϵ is an error associated with each point and is used to weight the data. This is an ill posed problem, so regularization has to be applied:

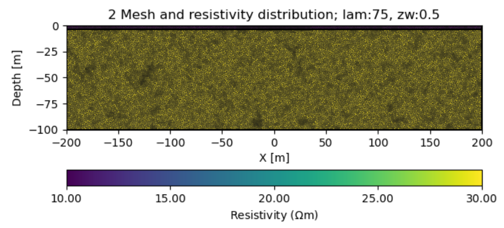
$$\Phi = \Phi_d + \lambda\Phi_m \rightarrow \min \tag{2-4}$$

where λ is a regularization parameter that weights measurements and model [Jiang et al., 2020]. The choice of its value has consequences. It is a trade-off between how well the data is fitted and how smooth / rough the model is [Günther et al., 2006b], [Tikhonov and Arsenin, 1977]. For more details on λ please see Section 4-1-3.

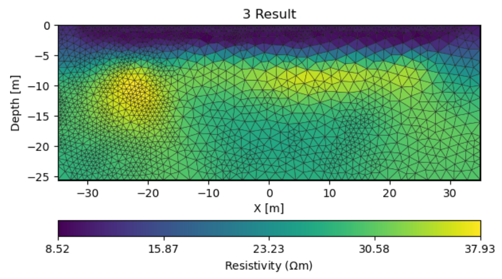
The methods described in this section apply to both synthetic and real data.



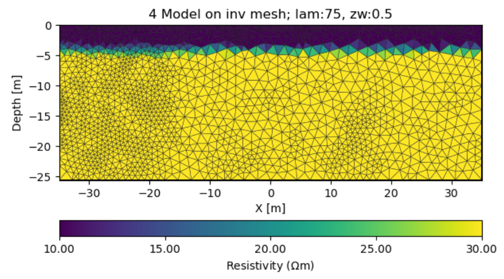
(a) Geometry of the subsurface - colors indicate different layers



(b) Mesh used for forward modeling and resistivity distribution



(c) Inverted result of the ERT measurement simulation



(d) Input model resampled to the inversion mesh

Figure 2-1.2: Sample of the synthetic profile created with PyGIMLi

2-2 Machine learning

In this section the different machine learning techniques used in this project will be introduced and briefly discussed.

2-2-1 Supervised and unsupervised learning

The methods used in this project can be split into two groups based on how they learn from the given data. There are different ways of learning but for this project only two are important: supervised and unsupervised [Bishop, 2006]. The first one relies on features in the provided dataset which contains information about the expected result of the prediction (here it is an expected class) for each sample. Based on this given "answer" and the data itself, the classifiers will try to develop a way of assigning each sample to one of the provided classes. The unsupervised learning requires no expected answers to be provided. Instead unsupervised learning methods (clustering) will try to group points into classes based on the self similarity of the samples in each class.

2-2-2 Clustering

Clusters can be defined in a very intuitive way: "groups of data points whose inter-point distances are small compared to points outside of the cluster" [Bishop, 2006]. Clustering represent an unsupervised way of learning. In this project, two clustering techniques will be used: KMeans and Mean Shift. They will be discussed in following sections.

KMeans

KMeans is a simple and widely used technique. Its goal is to develop a set of cluster centers that would represent subgroups in the data set [Müller, A. C., & Guido, S., 2017]. Mathematically the principle of this technique can be described with the following formulation:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 \tag{2-5}$$

where x is the sample point and μ_k is an average of a cluster K; N and n indicate the points and K and k indicate the specific cluster. r_{nk} indicates whether the point n belongs (1) or not (0) to the specific cluster k. This is a distortion measure, that KMeans is minimizing [Bishop, 2006]. The selection of the positions of the clusters and points belonging to each of them are changed iteratively. It uses euclidean distance to measure the distances between samples [Hartigan, 1975]. Each iteration consists of two phases: expectation and maximization. In the first one r_{nk} is changed while μ_k is fixed, in the second one it is reversed. In both phases J is minimized [Bishop, 2006].

MeanShift

The MeanShift is a centroids based method. In each iteration they are shifted towards "a region with maximum density of points" based on the mean shift vector, which is defined as follows [Pedregosa et al., 2011]:

$$m\left(x_i\right) = \frac{\sum_{x_j \in N(x_i)} K\left(x_j - x_i\right) x_j}{\sum_{x_j \in N(x_i)} K\left(x_j - x_i\right)} \tag{2-6}$$

where x indicate a centroid [Comaniciu and Peter, 2002] and K is defined as follows [Cheng, 1995]:

$$K(x) = \begin{cases} 1 & \text{if } \|x\| \le \Delta \\ 0 & \text{if } \|x\| > \Delta \end{cases} . \tag{2-7}$$

where Δ is defined as the characteristic function.

2-2-3 Classifiers

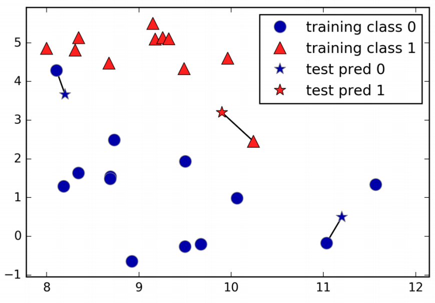
Classifiers are a ML techniques that belongs to the supervised learning group. In this project the following classifiers will be used: k nearest neighbours (KNN), stochastic gradient descent (SGD), support vector machines (SVM), deep neural networks (DNN), adaptive boost (Adaboost) and gradient boosting classifier (GBC). They will be introduced in following sections.

K Nearest Neighbours

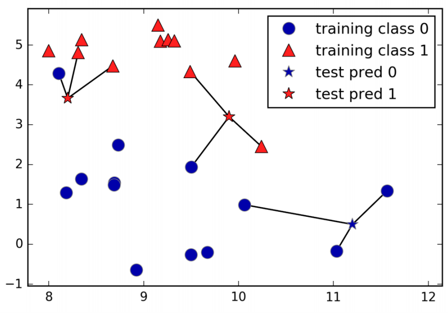
K Nearest Neighbours (KNN) is one of the simplest classifiers [Müller, A. C., & Guido, S., 2017]. Nevertheless, it is widely applied to a wide range of different problems (e.g.: hyperglicemia diagnosis [Sarwar, 2017] or spam detection [Salih and Dhannoon, 2021]). In its basic form, this method is used to predict the probability density $p(x)$ but it can be extended to a classification problem [Bishop, 2006]. The principle of this algorithm can be written mathematically as how probable it is that each point belongs to a specific class. For the purpose of this project a simpler concept is sufficient. In the basic form of one-nearest-neighbour each centroid will be matched with the closest point to it within the data set. In the prediction phase, each new point given to the classifier will be compared with all training points and it will be matched with the one that is the most similar to it. Next, the point will be assigned the same class as the training point is was matched to. However, if the decision is based only on one point from the training dataset, the outcome of the prediction may not be reliable. For example the outliers could strongly influence the result of the prediction. Consider a prediction of the class of a single point that is surrounded by training points that belong to the same class but the closes point to it is an outlier representing a different class. In this situation the incorrect class might be assigned because the outlier is the closest training point to the point for which the class is being predicted. In spite of multiple trainings point with correct class being available in the vicinity of the point used for prediction. In order to avoid this problem one neighbour can be replaced with any arbitrary number of them indicated by k (hence the name - *k nearest neighbours*). The final class will be assigned based on which class had the most occurrences among the k neighbour points. This concept is illustrated in Figure 2-2.1
The KNN technique has several drawbacks. The KNN is a so called ”lazy learner”. Unlike, for example SVM, it does not learn from the data by developing a margin between classes or any new conclusion. Instead, its predictions are based purely on comparing the new point with all samples used to train the classifiers. This means, that it may be challenging for it to generalize. If the prediction dataset contains points, that will differ from the training dataset the result may be inaccurate or incorrect. [Jivani et al., 2016]

Support Vector Machines

Support vector machines (SVM) try to classify data by developing a margin. It can be defined as a maximal distance between the decision boundary and any of the given samples. There is penalty for the points that are left on the ”wrong side of the margin” and the classifier tries to minimize it. [Boser et al., 1992] The strength of the idea of the margin (or the optimal hyperplane as it was defined in the publication that introduced this concept [Vapnik and Kotz, 2006]) is that it has to consider only a subsection of the training samples in order to develop the margin. Those points are the support vectors [Cortes and Vapnik, 1995]. A simple example of the margin in a 2D space can be found



(a) .



(b) Real measurements - classifiers.

Figure 2-2.1: KNN principle: (a) one-nearest-neighbour; (b) k-nearest-neighbours where $k = 3$; illustration from Müller, A. C., & Guido, S., 2017

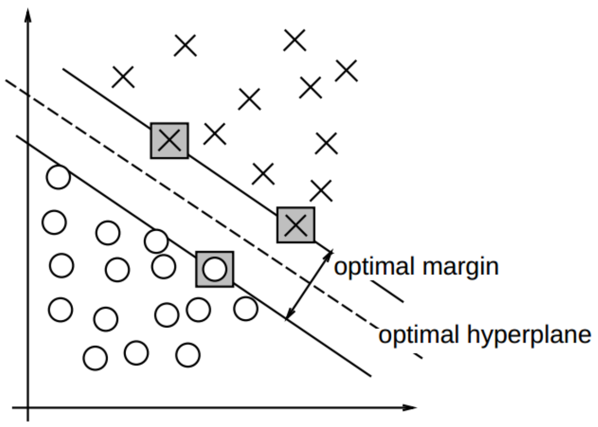


Figure 2-2.2: The idea of SVM margin from Cortes and Vapnik, 1995.

in the Figure 2-2.2.

The SVM directly supports only two class problems. For the purpose of this project this is sufficient. They can be extended to support multiple classes with one-vs-the-rest approach [Bishop, 2006]. Another strength of the SVM is their capability to identify and eliminate outliers [Boser et al., 1992].

Stochastic Gradient Descent

The stochastic gradient descent (SGD) is based on a gradient descent algorithm. Its strength is its efficiency. The computational effort associated with the standard gradient descent algorithm increases with the number of samples. It can quickly become unfeasible

to take the next step within the gradient if there are many samples available [Goodfellow et al., 2016]. Another strength of the SGD is that it is a sequential learning algorithm. It is not necessary to present the whole population of data at once to the classifier. [Tsuruoka et al., 2009] This can be useful to monitoring projects as SGD can be updated in real time.
Mathematically, the SGD classification is formulated in a following way (after [Pedregosa et al., 2011]):

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f\left(x_i\right)\right) + \alpha R(w) \tag{2-8}$$

with L indicating a loss function (measures errors, misfit) and R is a regularization function that penalizes errors and complexity, x and y are training sets, and α is a hyperparameter controlling regularization strength ($\alpha > 0$) and $f(x)$ is a scoring function:

$$f(x) = w^T x + b \tag{2-9}$$

where w is a vector containing model parameters and b is an intercept. The behaviour of the SGD can be altered by selecting a different loss functions L. For example if L is defined as soft-margin, SGD will behave like an SVM. At each iteration the parameter vector w will be updated as follows [Bishop, 2006]:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n \tag{2-10}$$

with τ indicating the iteration number. η is a learning rate parameter, which has to be selected carefully as setting it to a too large value will prevent it from converging and selecting a too small value may increase the time required to converge. Scikit-learn defines η for classification as follows:

$$\eta^{(t)} = \frac{1}{\alpha\left(t_0 + t\right)} \tag{2-11}$$

where $t = n_{samples} * n_{iter}$ and indicates the time step, t_0 is determined by the algorithm [Bishop, 2006].

Deep Neural Network

The Deep Neural Network classifer (NN) in scikit-learn uses multilayer perceptrons (MLP) [Pedregosa et al., 2011]. In principle MLP consists of an input layer with neurons representing the input features, multiple hidden layers and an output layer. In each neuron in the network the transformed values are passed to it from a previous layer based on weighted summation. Next, the values interact with an activation function. Different types of them are available. Their purpose is to simulate the response of neurons [Huang et al., 2020]. Those two steps are repeated in each layer. Finally the output layer maps the values to the output features.

A neural network with one hidden layer can be mathematically described as follows:

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{2-12}$$

where y_k is an output feature, σ is an activation function, w indicate weights of connections between neurons and x is a set of input values. The content of vector w (weights, parameters) is what is developed during training. [Bishop, 2006] This is done by minimizing an error function that depends on vector w:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{y\left(\mathbf{x}_n, \mathbf{w}\right) - t_n\right\}^2 \tag{2-13}$$

its minimum can be found where:

$$\nabla E(\mathbf{w}) = 0 \tag{2-14}$$

as the dependence of E on w is usually highly non-linear, there may not be an analytical solution. Therefore mathematical methods have to be used. [Bishop, 2006] In order to reach a set of parameters that would fulfil equation 2-14 different methods can be used. Scikit-learn implements three of them: SGD, Adam and L-BFGS. [Pedregosa et al., 2011] The most appropriate one can be selected by a hyperparameter grid search (for more details please see Section 3-3-7).

Adaboost

The Adaboost name originates from its capability to adapt to the errors of weak classifiers. [Freund and Schapire, 1997] Adaboost belongs to the boosting algorithms. Boosting is a process in which multiple prediction rules (rules of a thumb, weak classifiers) that fail to deliver a reliable prediction on their own (but are better than random guess) are combined to a rule that is very accurate. [Drucker, 1997; Freund and Schapire, 1997]

Adaboost is one of the best classifiers for two case problems, but this is not the case anymore when extended to multicase problems. [Freund and Schapire, 1997] This limitation is related to an assumption of the error of each weak classifier being smaller than 0.5. [J. Zhu et al., 2006] If a classifier has to decided whether a data point belongs to class A or B, with no other option, as long as the data points are equally distributed between classes, this condition will be satisfied. [Freund and Schapire, 1997] However, when classes C, D and following are added, or when the data points belong predominantly to one of the two classes, this assumption may not be valid anymore. In multi class problems the error rate associated with random guessing is defined as:

$$err = (K-1)/K \tag{2-15}$$

where K is number of classes. [J. Zhu et al., 2006] In its original form the weighting coefficient (β) is calculated with the following formulation:

$$\beta^{(m)} = log(\frac{1 - err^{(w_k)}}{err^{(w_k)}}) \tag{2-16}$$

where w_k indicates a weak classifier. In order to extend the Adaboost capabilities to a multi class case, the following modification was introduced by [J. Zhu et al., 2006]:

$$\beta^{(m)} = log(\frac{1 - err^{(w_k)}}{err^{(w_k)}}) + log(K - 1) \tag{2-17}$$

where K indicates the number of classes. This modified approach used by Scikit-learn is called Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME) [Pedregosa et al., 2011].

Gradient Boosting Classifier

Similarly to Adaboost, the gradient boosting classifier (GBC) belongs to a family of ensemble methods which combine multiple weak classifiers to obtain better result than any of them would achieve on their own. This is done by boosting - each next weak classifier is given a chance to learn and improve from previous classifiers mistakes.

2-2-4 Committee

The aim of the committee is to combine predictions made by different clusters or classifiers. It resembles to some extent council of precogs from "Minority Report" by Phillip K. Dick [Dick, 1956]. It is done by a democratic vote over predicted class for each point separately. There is not a minimum vote threshold required to make a decision. The prediction with the highest number of votes defines the assigned class prediction.

Chapter 3

Experiment

3-1 Procedure and setup

3-1-1 Forward Modeling and inversion

In this Section I will discuss how synthetic data is being created and how real measurements are inverted. For all steps discussed in this section PyGIMLi is used.

ERT survey simulation

The first step of forward modeling is to define the geometry of the environment in which ERT survey will be simulated. In order to minimize boundary effects, the forward modelling domain has to be significantly larger than the extent of the ERT survey. In this project the environment has been defined without any topography (surface is entirely flat and horizontal) and it is 400 meters long and 100 meters deep. The number of interfaces and their positions are defined for each profile separately. The same applies to the resistivities of layers.
After defining the environment, the survey itself has to be defined. It consists out of one profile, that is 64 meters long and its center is placed in the middle of the environments surface (profile reaches 32 meters to the left and to the right of the surface). The estimated depth of investigation is between 18 to 20 meters. 44 electrodes are used, with a spacing of 1.5 meters. Dipole-dipole has been selected as measurement scheme (please see Section 2-1 for more details). Based on the geometry of the environment, mesh is generated. Next step was to execute the ERT survey. In order to add some realism to this simulation, artificial noise was added. Noise is controlled by two parameters: its level (which was set to 1%) and absolute voltage error (set to 1^{-6} Volts).

Inversion

The measurements obtained from the forward modeling (the simulated ERT survey) now have to be inverted in order to obtain a resistivity profile. Choice of parameters controlling the inversion is not trivial. The selected parameters for profiles can be found in Table 3-1 and 3-2. Inverted data often contain some outlier, that can make it very hard to interpret profile. In order to solve this problem, the result of the inversion is trimmed. 10% of highest and lowest vales are removed. This is an arbitrary threshold and it has been selected experimentally. At this point the input model will be interpolated to the inversion mesh in order to compare it with the result of the inversion.

Assignment of classes

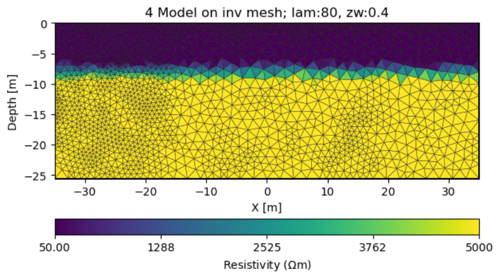
This section will briefly discuss how classes are assigned. It is an important step as they will be later used to train the classifiers. Classes are assigned based on the resistivity values from the input model interpolated to the inversion mesh. The interpolation is important, because it allows to match each cell in the inverted model to a resistivity that it should have if there was no noise, the inversion was absolutely perfect and ERT survey extracted all information from the subsurface about its true resistivity. This way classifier can indirectly compare the imperfect data with a real model. It is important that the classifiers learn to predict on imperfect data as real data will contain noise and other imperfections.

Binary classes This approach is only appropriate for profiles with one interface and two layers. Each layer (associated with its resistivity) was considered as one class and assigned label 0 or 1, accordingly. Each point in the input model was assigned into the class, that has identical resistivity. For points that have resistivity values different that do not match any of the classes (due to inaccuracies introduced by interpolation), the class with closest resistivity value will be assigned to the point.

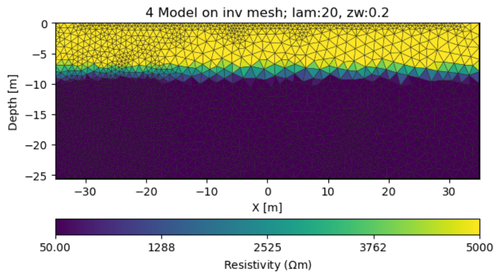
Multiple classes If there are multiple layers in the profile, a multiple classes approach is more suitable. However it can work as well if there are less layers than classes. Classes were assigned using the normalized resistivity values. The range between 0 to 1 were split into multiple subranges. The number of those subranges will match the number of classes. Each point was assigned that class that matches its resistivity value. The values at the boarder of the classes are always assigned to the higher class.

3-2 Data

In this Section data sets used for testing will be discussed. Overall the profiles used can be split into two categories: synthetic and real. Most of this study has been performed



(a) Example of a normal profile.



(b) Example of inverted profile.

Figure 3-2.1: Example of the normal and inverted profiles.

with synthetic profiles as they are fully controlled and all of their parameters can be defined, hence they are known and there is no uncertainty about them.

3-2-1 Synthetic data

Synthetic profiles can be split into two categories: normal and inverted. They differ in the order of the layers. In normal profile less resistive layer is on top (just below the subsurface, at the upper side of the interface). The inverted profiles are the opposite: less resistive layer is at the bottom (further away from the subsurface, below the interface). This is the naming convention used in following chapters. The profiles are summarized in the Table 3-1 for the normal profiles and Table 3-2 for the inverted profiles.
There are three resistivity contrast in each group 50 vs 500 Ωm, 500 vs 5000 Ωm and 50 vs 5000 Ωm. For the inverted profiles those contrasts are inverted, with higher resistivity being assigned to the upper layer (see Figure 3-2.1 for examples of both types of profiles). For each contrast there are 14 depths of the interface ranging from 2 m below the surface down to 15 m below the surface in intervals of 1 m. There are 42 profiles in each group, making it 84 profiles in total. For more details on how they were generated please see Sections 3-1-1, 2-1-1 and 2-1-2.

Overview of features of each profile

Each profile is stored in a csv file. It is an exported pandas dataframe in which each row corresponds to a point (sample) and each column is a feature. Each point originates from a cell in the inversion mesh. The most important features are resistivity (RES) and normalized resistivity (RESN). Both are based on the resistivity distribution obtained from inversion. To RES only decimal logarithm has been applied and RESN has been additionally normalized. Additional features are sensitivity (SEN) and spatial position of each point (X, Y and Z coordinates relative to the center of surface). They can be used as input for both training and prediction.
In order to train supervised classifiers each point needs to have an expected class assigned. There are three values available for this purpose in each profile. CLASS is a

Table 3-1: Overview of the parameters (resistivity and interface depth), that define synthetic profiles used for training and prediction. Resistivity is given in $\Omega * m$ and depth is given as meters below the surface. $\rho 1$ refers to upper layer in the subsurface and $\rho 2$ refers to lower layer.

Name	Number of layers	ρ 1	ρ 2	interface depth
Hor1_0001	1	50	500	2
Hor1_0002	1	50	500	3
Hor1_0003	1	50	500	4
Hor1_0004	1	50	500	5
Hor1_0005	1	50	500	6
Hor1_0006	1	50	500	7
Hor1_0007	1	50	500	8
Hor1_0008	1	50	500	9
Hor1_0009	1	50	500	10
Hor1_0010	1	50	500	11
Hor1_0011	1	50	500	12
Hor1_0012	1	50	500	13
Hor1_0013	1	50	500	14
Hor1_0014	1	50	500	15
Hor1_0015	1	500	5000	2
Hor1_0016	1	500	5000	3
Hor1_0017	1	500	5000	4
Hor1_0018	1	500	5000	5
Hor1_0019	1	500	5000	6
Hor1_0020	1	500	5000	7
Hor1_0021	1	500	5000	8
Hor1_0022	1	500	5000	9
Hor1_0023	1	500	5000	10
Hor1_0024	1	500	5000	11
Hor1_0025	1	500	5000	12
Hor1_0026	1	500	5000	13
Hor1_0027	1	500	5000	14
Hor1_0028	1	500	5000	15
Hor1_0029	1	50	5000	2
Hor1_0030	1	50	5000	3
Hor1_0031	1	50	5000	4
Hor1_0032	1	50	5000	5
Hor1_0033	1	50	5000	6
Hor1_0034	1	50	5000	7
Hor1_0035	1	50	5000	8
Hor1_0036	1	50	5000	9
Hor1_0037	1	50	5000	10
Hor1_0038	1	50	5000	11
Hor1_0039	1	50	5000	12
Hor1_0040	1	50	5000	13
Hor1_0041	1	50	5000	14
Hor1_0042	1	50	5000	15

Table 3-2: Overview of the parameters (resistivity and interface depth), that define synthetic profiles used for training and prediction. Resistivity is given in $\Omega * m$ and depth is given as meters below the surface. $\rho 1$ refers to upper layer in the subsurface and $\rho 2$ refers to lower layer.

Name	Number of layers	Rho 1	Rho 2	interface depth
Hor1_inv_0001	1	500	50	2
Hor1_inv_0002	1	500	50	3
Hor1_inv_0003	1	500	50	4
Hor1_inv_0004	1	500	50	5
Hor1_inv_0005	1	500	50	6
Hor1_inv_0006	1	500	50	7
Hor1_inv_0007	1	500	50	8
Hor1_inv_0008	1	500	50	9
Hor1_inv_0009	1	500	50	10
Hor1_inv_0010	1	500	50	11
Hor1_inv_0011	1	500	50	12
Hor1_inv_0012	1	500	50	13
Hor1_inv_0013	1	500	50	14
Hor1_inv_0014	1	500	50	15
Hor1_inv_0015	1	5000	500	2
Hor1_inv_0016	1	5000	500	3
Hor1_inv_0017	1	5000	500	4
Hor1_inv_0018	1	5000	500	5
Hor1_inv_0019	1	5000	500	6
Hor1_inv_0020	1	5000	500	7
Hor1_inv_0021	1	5000	500	8
Hor1_inv_0022	1	5000	500	9
Hor1_inv_0023	1	5000	500	10
Hor1_inv_0024	1	5000	500	11
Hor1_inv_0025	1	5000	500	12
Hor1_inv_0026	1	5000	500	13
Hor1_inv_0027	1	5000	500	14
Hor1_inv_0028	1	5000	500	15
Hor1_inv_0029	1	5000	50	2
Hor1_inv_0030	1	5000	50	3
Hor1_inv_0031	1	5000	50	4
Hor1_inv_0032	1	5000	50	5
Hor1_inv_0033	1	5000	50	6
Hor1_inv_0034	1	5000	50	7
Hor1_inv_0035	1	5000	50	8
Hor1_inv_0036	1	5000	50	9
Hor1_inv_0037	1	5000	50	10
Hor1_inv_0038	1	5000	50	11
Hor1_inv_0039	1	5000	50	12
Hor1_inv_0040	1	5000	50	13
Hor1_inv_0041	1	5000	50	14
Hor1_inv_0042	1	5000	50	15

Table 3-3: Split data for training and prediction

Training batches			
Mixed		Normal	Inverted
Normal	Inverted		
Hor1_0001	Hor1_inv_0001	Hor1_0001	Hor1_inv_0001
Hor1_0003	Hor1_inv_0003	Hor1_0003	Hor1_inv_0003
Hor1_0005	Hor1_inv_0005	Hor1_0005	Hor1_inv_0005
Hor1_0007	Hor1_inv_0007	Hor1_0007	Hor1_inv_0007
Hor1_0009	Hor1_inv_0009	Hor1_0009	Hor1_inv_0009
Hor1_0011	Hor1_inv_0011	Hor1_0011	Hor1_inv_0011
Hor1_0013	Hor1_inv_0013	Hor1_0013	Hor1_inv_0013
Hor1_0015	Hor1_inv_0015	Hor1_0015	Hor1_inv_0015
Hor1_0017	Hor1_inv_0017	Hor1_0017	Hor1_inv_0017
Hor1_0019	Hor1_inv_0019	Hor1_0019	Hor1_inv_0019
Hor1_0021	Hor1_inv_0021	Hor1_0021	Hor1_inv_0021
Hor1_0023	Hor1_inv_0023	Hor1_0023	Hor1_inv_0023
Hor1_0025	Hor1_inv_0025	Hor1_0025	Hor1_inv_0025
Hor1_0027	Hor1_inv_0027	Hor1_0027	Hor1_inv_0027
Hor1_0029	Hor1_inv_0029	Hor1_0029	Hor1_inv_0029
Hor1_0031	Hor1_inv_0031	Hor1_0031	Hor1_inv_0031
Hor1_0033	Hor1_inv_0033	Hor1_0033	Hor1_inv_0033
Hor1_0035	Hor1_inv_0035	Hor1_0035	Hor1_inv_0035
Hor1_0037	Hor1_inv_0037	Hor1_0037	Hor1_inv_0037
Hor1_0039	Hor1_inv_0039	Hor1_0039	Hor1_inv_0039
Hor1_0041	Hor1_inv_0041	Hor1_0041	Hor1_inv_0041

default class assigned to not normalized values (RES), CLASSN is assigned based on normalized resistivity (RESN) and Labels are string classes based on CLASSN.

Split between training and prediction

Data is split between training and prediction as it is shown in Table 3-3. After training of the classifiers with one of the training batches, all profiles that have not been used for training will be used for prediction. In order to provide both training and prediction phases with representative collection of profiles, the split is not made randomly. All profiles with odd number in their name are used for training and all profiles with even number are used for prediction.

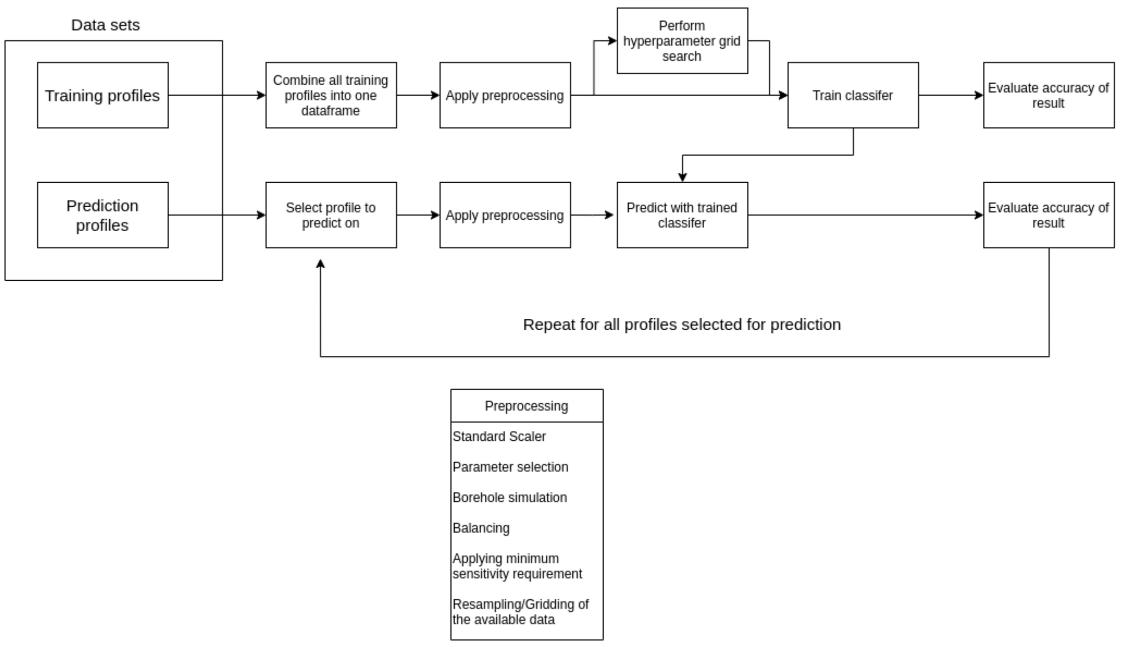

Data sets
Training profiles
Prediction profiles
Combine all training profiles into one dataframe
Apply preprocessing
Perform hyperparameter grid search
Train classifer
Evaluate accuracy of result
Select profile to predict on
Apply preprocessing
Predict with trained classifer
Evaluate accuracy of result
Repeat for all profiles selected for prediction
Preprocessing
Standard Scaler
Parameter selection
Borehole simulation
Balancing
Applying minimum sensitivity requirement
Resampling/Gridding of the available data

Figure 3-2.2: Procedure for training and prediction phases

3-2-2 Real data

14 profiles have been used. They were measured twice per month from November 2019 to June 2020 (apart of the first and last months from which only one measurement was included). Those profiles have been measured as a part of an ongoing research project at the Berkeley Lab. The profiles were measured in the San Francisco Bay area (west side of the Berkeley Hills). The ERT survey was carried out with a dipole-dipole array. 112 electrodes at 0.6 meter separation has been used. The length of the profile is 66.6 meters. [Uhlemann et al., 2021]. All used measurements were run at 14:30. They were inverted with λ 60 and z_weight 0.2.

3-2-3 Classification

Figure 3-2.2 shows the workflow of the training and prediction phases. In principle the training and prediction phases are very similar. Most of the steps are shared between them.

Training

First the modelled and inverted profiles are read from the files. In the training phase all profiles are combined and stored in a single dataframe. The classifiers require all

the points to be given as input ”at once”. Before training or prediction is performed, data has to be preprocessed. It includes different data manipulations, both using scikit-learn tools and related to requirements of the classifiers and external data manipulation focused on improving the performance of the classifiers by altering the available data. Those steps and order in which they are performed will be briefly introduced in this Section. For more details regarding each of those steps please see Section 3-3.
The first step of the prepossessing comprises deciding which features will be used for the classifier (please see Section 3-3-8 for more details). This step has consequences for both training and prediction as all selected features will be used for both training and prediction and will influence the performance of the classifier.
Next, the input features have to be transformed according to their type (this applies to both features and their labels). This is done with two scikit-learn functions ”Standard-Scaler” and ”OneHotEncoder”. Please see Section 3-3-1 for more details about when and how they are used). This step will be put into a pipeline together with the classifier. Pipeline is a part of scikit-learn toolbox and ensures that all steps will be performed in the same, reproducible way for all applications of the classifier. In the prediction phase the same pipeline will be used, so steps discussed in this paragraph will be applied to the prediction phase as well. The pipeline does not have to be passed directly (within the same run of the script) to the prediction phase. After the training is completed the pipeline will be saved using joblib [Joblib-Development-Team, 2020] and can be loaded to be used in the prediction phase in another run of the script. Because of this, classifiers can be reused for evaluation of multiple sets of profiles without a need to retrain them.
The following steps are optional, as they are not preparing the data to match the requirements of the classifiers. However, they have some potential to improve the accuracy of the results. Those steps will be applied to available data, one at time, and it will be evaluated, whether the result benefits from them. The first step that can be taken is to change the population of points in terms of their spatial distribution and other qualities. This can be done by resampling the data (more details in Section 3-3-5). It will change the spatial distribution of the points. Another possibility is to remove points with low sensitivity (more details in Section 3-3-2). It is better to execute this step after resampling, as the inaccuracies of the interpolation may increase if the spatial distribution of points is less dense. Next step would be balancing the classes. For horizontally layered profiles it is appropriate to run this step now, but for other types of profiles it may be better if this step was delayed to after the next step (borehole simulation is applied). Please see Section 3-3-3 for more details. All steps discussed in this steps are applied to a dataframe containing all points from all profiles assigned to training, however those steps are applied to each profile separately. Otherwise some information could be lost, or get unproportionally boosted. Balancing of the classes can serve as a good example. It should be done for each profile separately as the distribution of the points between classes depends on the position of the interface. Therefore applying balancing to the whole population of points at once, could change the weight of each profile within the training dataset.
Next step, the borehole simulation, is applied only in the training phase. There are two

reasons why it is beneficial to train on a subset of input data that resembles a borehole. Firstly, it would be closer to a real world application if training was done on the borehole data and prediction on the inverted ERT measurement (in form of a full profile). Secondly it reduces the computational cost significantly, without having a negative impact on the result.
A last step that can be taken is to apply weights to data points of the training data set. Here I have chosen to use the sensitivity, which is provided by PyGIMLi as part of the synthetic modelling, to weight the data points, where small sensitivities relate to small weights and large sensitivities to large weights. This was chosen because large sensitivities relate to areas of the model that are well constrained by the data. For more detail please see Section 3-3-2.
The final step, that can be performed only during the training phase, is hyperparameter tuning (described with more detail in Section 3-3-7). It will deliver the best parameter set for each classifier, which performed best while training. The classifiers with their respective best set of parameters will be passed in form of a processing pipeline to the prediction phase.
Last part of training phase is evaluation of the accuracy for each training profile separately. Section 3-4 discusses how the accuracy is evaluated.

Prediction

In terms of data preprocessing and accuracy evaluation the prediction phase is similar to the training phase. Therefore only the differences will be discussed in this Section.
The main difference is how the data is fed to the classifier. Unlike in the training phase, prediction on each profile is performed separately. The steps, that are not bundled together in the pipeline are applied in the same way, but for one profile at time.
What is different is which steps are being applied. The borehole simulation and hyperparameter tuning are not used during prediction. Borehole simulation does not offer any benefit to the prediction phase as discussed in previous Section. For the hyperparameter tuning, it is already included in for of the optimized classifier contained inside of pipeline, therefore there is no need to repeat this step again. In case the pipeline is not passed directly between training and prediction phases, it will be loaded from one of the previous training phases.
After the prediction, each profile is evaluated using the same techniques as in the training phase (please see Section 3-4 for more detail).

3-3 Improvements to classification

3-3-1 Scikit-learn tools

First two techniques discussed in this section (Standard Scaler and One Hot Encoder) are part of the scikit-learn package. The other ones have been developed and implemented as part of this project.

Standard scaler

Many classifiers require data to approximately resemble ”standard, normally distributed data” [Pedregosa et al., 2011]. In order to achieve this a Standard Scaler is used. It is a function from scikit-learn’s preprocessing toolbox and can be included in the processing pipeline. It transformers numerical features by first subtracting their mean value and than scales it by dividing each value with the standard deviation of the feature it belongs to. Each feature is preprocessed separately.

One Hot Encoder

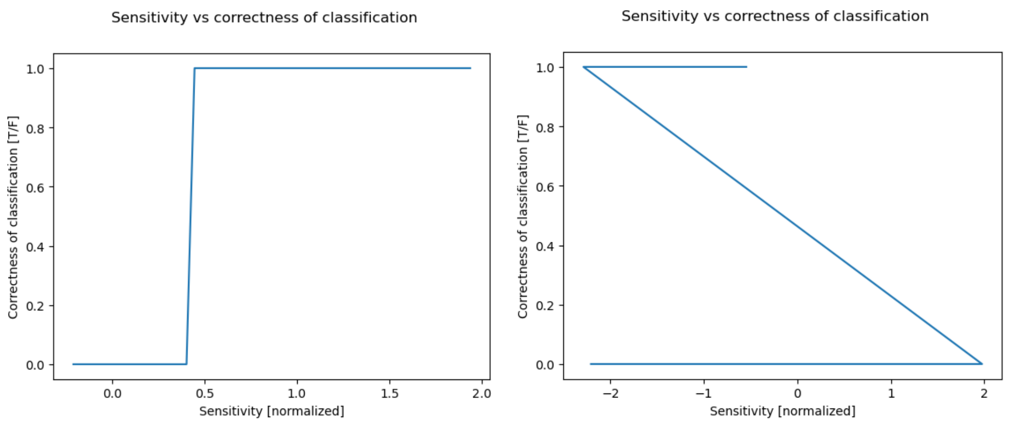
One Hot Encoder converts text labels into a form, that is more understandable to the classifiers. It creates columns. Its number matches the amount of unique labels found in the dataset. In each column there is a 0 or 1 value marking whether the label represented by the column is assigned to the point or not.

3-3-2 Minimum sensitivity

The sensitivity of the the measurements varies throughout the profile. Due to this variation, not all data points carry the same importance and quality of information with them. PyGIMLi returns sensitivity as a float array with one entry for each cell. Because of the decimal logarithm, the values can be both negative and positive. Therefore in order to ensure, that the cut-off threshold is consistent between profiles with different ranges of sensitivity values, it has to be normalized. This is done as follows:

$$sen_{norm} = \frac{abs(sen) + abs(min(sen))}{abs(max(sen)) + abs(min(sen))} \tag{3-1}$$

This step should be applied to both training and prediction steps. However intuitively, it can be said, that it is more important for training. The idea is to avoid confusing the classifier by removal of cells, that carry less information.
However selecting a threshold for the cut-off is not a trivial task. It will be discussed now, by comparing two examples. In the Figure 3-3.1 two plots of accuracy of classification depending on their sensitivity are shown. Subfigure 3-3.1a shows a plot for a profile, for which there can be obtained a clear value for the cut-off sensitivity threshold. For this specific profile it would be around 0.45. All lower values are miss-classified, therefore they do not contain good enough information or the classifier is not capable of using them. Subfigure 3-3.1b shows an example where the relationship between sensitivity and training accuracy is complex and it is impossible to pick a unique cut-off value. Incorrectly classified points are spread over the whole range of sensitivity values. All correctly classified point have the sensitivity value below zero. This indicate generally a poor performance of the classifier for this profile.



(a) Correctness of classification for profile Hor1_A_0011 with GBC classifier.

(b) Correctness of classification for profile Hor3_C_0012 with SVM classifier.

Figure 3-3.1: Correctness of classification depending on sensitivity. On Y-axis 1 indicates, that point is classified correctly and 0 indicates incorrectly classified points.

3-3-3 Class balancing

Imbalanced classes in the training data are a common problem of ML applications. There are many papers discussing this problem, for example [T. Zhu et al., 2017]. Unfortunately for the ERT profiles this is often not the case. First, consider the most basic geometry: one horizontal interface in the whole subsurface with strong contrast between layers. If the classifier is trained to identify two classes in this profile, a good result can be expected. Even if amount of points in each class will differ, they both should be fairly well represented. However it quickly gets more complicated if there are more classes or more interfaces. Considering a case of a profile with one interface and 4 classes it is clear that the two classes assigned to either of the two layer will have a population of points much higher than two remaining classes assigned to the transition zone between layers. If the interface is well resolved, the population difference between classes can even reach multiple magnitudes. In this situation, it is possible, that for the less populated classes, there may be not enough information available for the classifier. Further consequences of this problem will be discussed in Section 3-4.

This problem can be solved in two ways: by up and downsampling. In upsampling, the less abundant classes will be resampled to increase their population. Downsampling, will reduce the population of the more abundant classes. Both solutions have some benefits and disadvantages. Upsampling will interpolate new points between existing ones. This may add some numerical errors to the dataset. However, if the data has been inverted on relatively fine grid, the error should be acceptable. Downsampling will reduce the pool of available data. In extreme case, downsampling may lead to using only a very small fraction of the data.

In this project classes will be balanced by upsampling using scikit function ”resample”. This decision can be justified for two reasons. The first one is how it would influence data if combined with other steps taken to improve the results. Downsampling combined with simulation of the borehole measurement (please see Section 3-3-4 for more details) could lead to only very few points being used for classification. Second reason, why upsampling has been selected is related to what kind of data is used. Both training and prediction will be using synthetic data calculated on fine grids. Therefore interpolation between centers of the cells should not lead to strong numerical errors, as new points will be generated very close to already existing ones.

3-3-4 Borehole simulation

This step can be used to simulate classifier training on data obtained from boreholes. There are two reasons why this step should be applied. First one is related with what kind of data can be obtained from the field measurements. The resistivity distribution in the subsurface can be directly measured either during excavation, or by measurement in boreholes. An example of this approach is discussed in [Ramirez et al., 2000]. Therefore verifying if this kind of training data (in terms of both number of the points available for training and their spatial distribution) would suffice to achieve satisfying accuracy is important. The other reason why this step is applied to all profiles is related to computational effort. As discussed in Section 3-2-3, all points used for training have to be delivered to the classifier "at once" in one dataframe. Depending on how dense the mesh in each profile is and how many profiles are supplied to the classifier, the computational effort may quickly become unfeasible. With smaller training data sets it is possible to use full profiles, but the amount of points that the classifier has to process will quickly turn into a limiting factor. Limiting the number of points used, by simulating borehole improves the performance significantly.
The borehole measurement will be simulated by limiting the population of points used for training to those with two, thin, vertically elongated rectangles. One of them is placed near the centre of the profile and the other one is closer to the left end of the profile. Although the reduction of the computational effort is significant, this method of training has its limitations and disadvantages as well. The main concern is how well will the subset of data represent profiles with more complex structure? For profiles, that have flat topography and horizontal layers, it may make little to no difference as the subset of the points should look the same or very similar throughout the profile. However if classifier was supposed to train using spatial parameters as well in order to recognize tilted layers, or structures like folds, reducing the population of points to those available in two narrow boreholes may make it impossible to deliver sufficient information.

3-3-5 Resampling

The main goal of this step is not directly related with the accuracy of prediction, but with computational effort of training. Dataset can be resampled in order to lower the

amount of points and still give the classifier access to the whole profile. The exact way of doing it can be adjusted according to the available information of the subsurface. For example the points can be resampled with bigger spacing if it is know, that layers are thick, or spacing can be increased only in one direction if the subsurface is known to be layered horizontally.
However resampling has disadvantages. It has its own computational effort and in some situation, this additional effort may consume the improvement of the computation time of the training. Apart from this, if the resampled profile uses mesh with big cells, there may be some inaccuracies introduced due to the interpolation between cells.

3-3-6 Weighting of samples

Some classifiers support passing sample weight as input. With those sensitivity can be used to inform classifier how informative each sample should be.

3-3-7 Hyperparameters tuning

The behaviour and performance of each classifier is controlled by a set of hyperparameters. They are specific to each type of classifiers. Choosing optimal ones requires a good understanding of both the classifier itself and the dataset it will be trained on. As a sufficient level of understating may not always be available, it is tempting to let the parameters be selected automatically.
The scikit-learn package offers tools, that allow to automatize the selection of the parameters for each classifier. Function GridSearchCv will investigate performance of different combinations of parameters within a range defined by the user.
For each classifier, only the best set of parameters set by GridSearchCV will be taken into consideration and only its performance will be evaluated and compared with other classifiers. It is important to notice, that the selected best combination of parameters may differ between experiments run with different configurations and datasets used for training.

3-3-8 Including additional information

Apart from resistivity, there are two more features on which classifiers could train and make predictions: spatial position and sensitivity. Giving classifier access to spatial position of each point can lead to a serious problem. Instead of classifying points mostly by resistivity, the classifier may focus on its positions. In order to avoid this problem, only depth will be used as feature. Some more details on this problem can be found in Section 3-4-1. The reason to include sensitivity is to give the classifier access to information how reliable each point is. Compared to the way of incorporating this value into training discussed in Section 3-3-6, this approach gives more freedom to the classifier in terms of how to use this information and what conclusions will be drawn from it. This freedom can actually turn into a problem when classifier decides to focus too much on the sensitivity while making prediction. This will be discussed in later sections.

3-4 Evaluation of classification accuracy

In this Section various ways of evaluating accuracy, or more generally, quality of the prediction made by clusters and classifiers is discussed. The choice of appropriate evaluation technique is crucial, as each of them answers a very specific question and some may not be fit for evaluating the prediction, that is expected from the classifier.

3-4-1 Conventional techniques

Accuracy score

The most basic way of evaluating the prediction accuracy is the accuracy score. It counts all the points which were assigned to the correct class and compares it to the whole population of points. This technique is implemented in scikit-learn as "accuracy_score" and is part of Scikit-learn's metrics toolbox. It can be mathematically described (in it's default normalized form) as:

$$\text{accuracy\_score}(y_{\text{pred}}, y_{\text{true}}) = \left(\frac{1}{n_{\text{points}}} \sum_{i=0}^{n_{\text{points}}-1} 1\,(y_{\text{pred}} = y_{\text{true}})\right) * 100\% \tag{3-2}$$

Where y_pred indicates the array of predicted classes in form of a vector with size of [n_points, 1], y_true is an array of true classes assigned to each point before the classification.

This simple way of evaluating the quality of prediction, but it has numerous advantages to it. It directly relates the content of the prediction to a summarized score. There is no complicated algorithm or mathematical procedure behind it and it is easy and clear in terms of interpretation. It directly answers question: "What fraction of points has been assigned to the correct class?". The second advantage is, that it only depends on one factor. Only points with correctly assigned classes influence the result. Therefore it is easy to visualize how accurate the resulting value is.

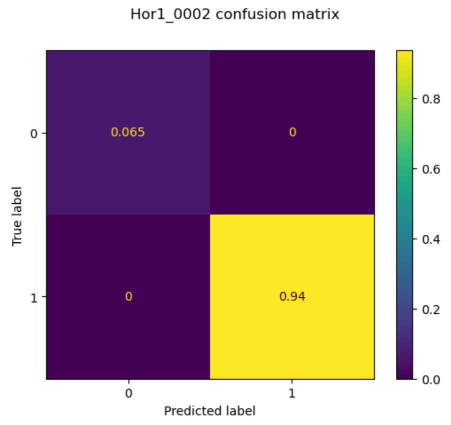
The simplicity of this method results in some limitations as well. If the dataset is strongly imbalanced. The accuracy score might be misleadingly high. As an example, consider a subsurface model with one interface at a depth of 2 meters and total depth of 20 meters, where the upper layer is assigned class 0 and the bottom one class 1. If the classifier fails to recognize points belonging to the upper layer and it will instead assign 100% of the points to class 1, it will still result in an accuracy of 90%. This seemingly good result will completely hide, that 100% of the population of class 0 has been incorrectly assigned to class 1. Furthermore for the specific goal of this project, the accuracy of class assignment is not the most meaningful parameter. There are situations in which prediction with poor accuracy score can still achieve a very good result in terms of detecting the interface. This problem will be discussed in more detail in Chapter 4. This method will be used as supportive for the results interpretation in Chapter 4.

Confusion matrix

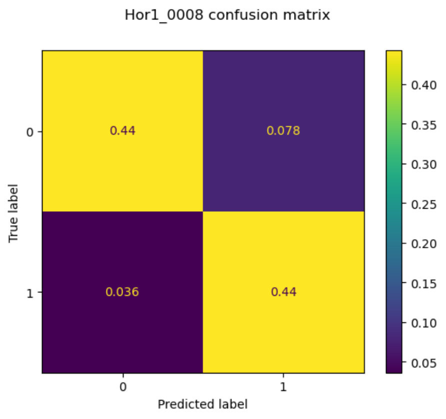
Unlike accuracy score, this method does not output one specific value. It returns a matrix showing what fraction of points has been assigned to the correct classes, which were assigned to the incorrect ones and to which class they were missassigned. In scikit-learn convention the rows present the true classes, and the columns the predicted classes. [Pedregosa et al., 2011], this is a notation found commonly in literature [Powers, 2008]. The biggest advantage of this evaluation method is that it offers more than just a single number and can be used to identify which classes are most challenging for the classifier. It delivers information about falsely assigned classes and how they relate to the real ones. Two examples of confusion matrix can be found in the Figures 3-4.1. The Figure 3-4.1a shows an example of a confusion matrix describing the result of a test where all points have been assigned correctly, the only non-zero values are found on the diagonal from top-left to the bottom-right of the matrix. Values are normalized relatively to the number of all points considered by the classifier. The Figure 3-4.1b shows an example of a confusion matrix where numerous points have been assigned to incorrect classes.
The confusion matrix can be a very good, supportive metrics for the interpretation and the evaluation of the result. Its limitation is similar to the one of accuracy score.

Feature importance

Feature importance is not strictly speaking a method of evaluation of the result. It can be used to verify, whether classifier has made prediction based on correct features.
This method is simple and self explanatory. If the classifier has been trained and is prediction on more than one feature (corresponding to the columns in pandas dataframe used an input), this diagram will show how meaningful each feature was for the specific prediction. It cannot be used a single technique to evaluate the quality of prediction, but it can help to look critically at results. It can happen, that seemingly good result has been achieved using mostly the information, that should not be the most important. For example: for the classifier is given RES/RESN (RES - resistivity, RESN - resistivity normalized), SEN (SEN - sensitivity) and Y (Y - depth), the expected hierarchy of those features is either RES-SEN-Y, or RES-Y-SEN (with preference for the first one). Any other order will indicated, that resistivity is not the most meaningful feature, hence the classification has occurred mostly on spatial position of the points (as SEN is related/dependent). This means even though the classifier may reach a high accuracy, it was a conclusion based on wrong assumptions, hence the result cannot be considered as trustworthy.
The Figure 3-4.2 compares an example of feature importance hierarchy of a trustworthy



(a) Confusion matrix for test Cla-A14-Hor1_0002



(b) Confusion matrix for test Cla-A11-Hor1_0008

Figure 3-4.1: Two examples of the confusion matrix. On the vertical axis the true classes are presented, and at the horizontal one the predicted ones. The value in each cell indicates which part of the samples belongs to the specific true/predicted class combination (range from 0 to 1, where 0 means that there are none sample it that combination and 1 means that all samples belong to the specific combination).

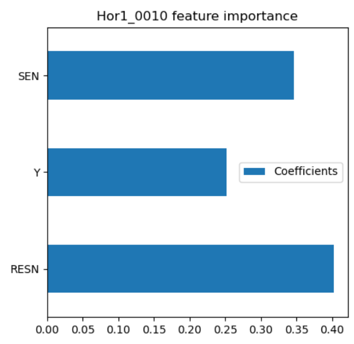
result (Figure 3-4.2a) and one in which classifier failed to obtain data in meaningful proportions (Figure 3-4.2b).

3-4-2 Interface detection accuracy

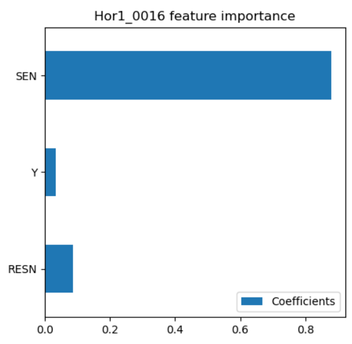
Interface detection accuracy is not a part of scikit-learn. It has been designed and implemented specifically for this project and with its goal in the focus. What this project aims at is automatic detection of interfaces between layers in ERT profiles, therefore accuracy of a classifier or influence of specific set of its parameter on it should take this into consideration.
As discussed in previous Sections, a high score obtained with classical methods may not always indicate, that the goal has been achieved. In some special cases it can even be the opposite. Considering this, developing a tool, that will evaluate the prediction in terms of its fitnes for the goal can be a good approach. Here we define the root mean square error (RMSE) of the true and predicted interface as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \tag{3-3}$$

where n indicates number of samples, Y_i is value of a sample i and $\hat{Y}_i$ is mean value.



(a) Feature importance of Cla-A11-Hor1_0010



(b) Feature importance of Cla-A11-Hor1_0016

Figure 3-4.2: Two examples of the feature importance plots. On the vertical axis there are feature acronyms (RES - resistivity; SEN - sensitivity, Y - depth). (a) shows a feature importance plot for a prediction in which the hierarchy of features is correct, the resistivity is recognized as the most important parameter as it should be; (b) shows a feature importance plot for a prediction in which the conclusions were drawn mostly from sensitivity feature, which is not correct as it was supposed to be only a supportive information to the main one - resistivity.

This introduces one significant difference between this method and the ones discussed so far. Accuracy score, confusion matrix and feature importance considered the direct output from the classifier and compared it with an expected answer. With "interface detection accuracy" there is an extra step between classification and interface detection. The algorithm will try to find an interface using points with classes assigned by the classifier and their spatial position. Than the depth of the detected interface will be compared with true depth of the interface.
The strength of this method is that it directly evaluates the quality of the prediction with respect to the goal of the project. The main goal of this project is to interpret where is the interface in the ERT data and this scorer can evaluate how well this goal has been achieved for each profile. On the other hand, identification of the interface is not a trivial task. Geology in the subsurface can be diverse, topography may vary between profiles. Hence, the performance of the detection algorithm may vary. It may introduce additional error and noise to the results. Still, since each profile will be subjected to the same interface detection procedure, with the same thresholds it can deliver a good way of comparing different classifiers, as long as its limitations and assumptions are taken into consideration while interpreting the result.

How is the interface detected?

As no interface detection function, package or toolbox has been found, that would fit the needs of this project, it had to be developed. In this section the algorithm will be

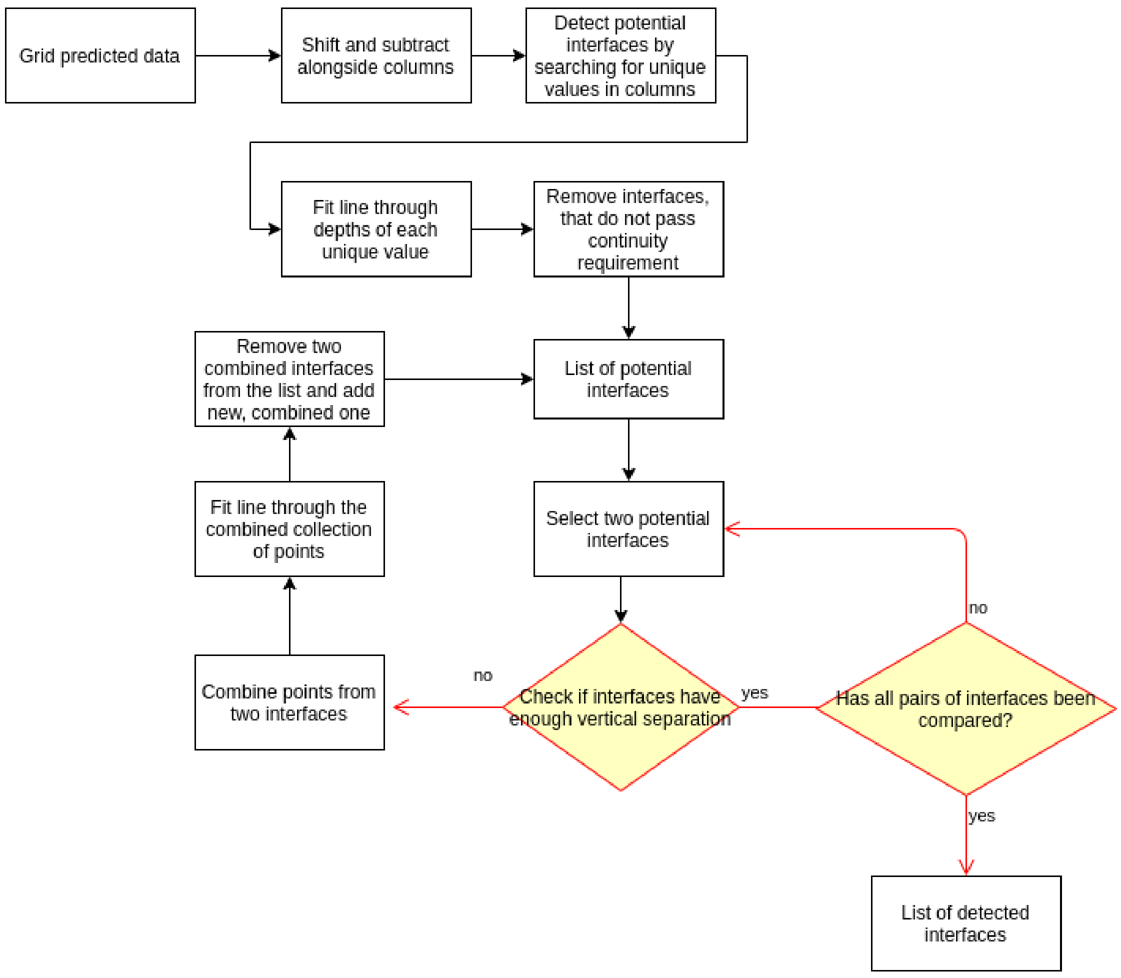

Grid predicted data
Shift and subtract alongside columns
Detect potential interfaces by searching for unique values in columns
Fit line through depths of each unique value
Remove interfaces, that do not pass continuity requirement
Remove two combined interfaces from the list and add new, combined one
List of potential interfaces
Fit line through the combined collection of points
Select two potential interfaces
no
Combine points from two interfaces
no
Check if interfaces have enough vertical separation
yes
Has all pairs of interfaces been compared?
yes
List of detected interfaces

Figure 3-4.3: Overview of the interface detection algorithm

introduced and assumptions that it is based on will be discussed. A general outline of the algorithm is illustrated in the Figure 3-4.3.

The first step is to resample data into a grid. Both forward modeling and inversion use unstructured meshes (please see Section 2-1-1 for more details). Gridding the data can be a computationally expensive for larger grids and could introduce some interpolation errors. The latter problem can be ignored for the datasets used in this project as they use very fine meshes. However in some cases it might have some impact on the overall quality. Despite the disadvantages of gridding, it has to be performed because following steps require matrix to be of a regular shape, specifically it is required that there are vertical columns in the data structure. It is required in order to shift the matrix and subtract it from the original one in one of the following steps.
This algorithm works with a 2D array of size [n_different_y_values, n_different_x_values]

containing numerical class identifier in each cell (for binary class problems it would be 0 and one, for n-class problem it would be a collection of integers from 0 to n-1). In order to ensure, that multiple interfaces within one profile can be detected, it is necessary to change values within the array. As the next step relies on the difference between values assigned to each class, interfaces between different classes should create different values. Classes labeled with consecutive integer would not fulfil this requirement, e.g.: the interface between classes 0 and 1 will result with a difference of 1 or -1 depending on the position of each class, the interface between classes 1 and 2 will have the same value.
Because of this, to each consecutive class number a "difference factor" will be added. This factor will depend on the desired difference between classes e.g.: the desired difference between classes 0 and 1 is 1, so the "difference factor will be zero", for classes 1 and 2 the desired difference will be 2, so the "difference factor" of 1 has to be added to class 2. This array will be copied, shifted downward by one unit alongside the vertical axis and subtracted from the original one. Now in the resulting array non zero values will be found only there where cells [m, n] and [m+1, n] have different class assignments. All other cells will contain zero.
At this point each non-zero cell (associated with its x and y coordinates) defines a point, that could potentially be a part of the interface. With perfect data at this point a good estimate of the interface would be available. Yet, as geophysical data is rarely perfect, there are more steps, that need to be taken. The following steps will be performed separately for each collection of points of the same value.
The first step is to fit a straight line through the points, this way both depth and slope can be obtained (as this project considers mostly horizontal layers, slope will be neglected). At this point the first check will be performed over the interface candidate - the continuity check.
During the development of this algorithm it turned out, that in more complex profiles, there may be multiple short and local interfaces detected. However it is assumed, that a valid interface is continuous throughout the whole width of the profile. On the other hand, there are some reasons, why even well detected interfaces may not be entirely continuous, therefore a continuity threshold was defined. In order to pass as an interface, at least 40% of the columns have to be identified as an interface. The value has been selected in an experimental way. It is high enough, that erroneous, short and local interface candidates are removed, but low enough that valid profiles are preserved.
In the next step each interface candidate will be compared with all other ones in order to check if any pair violates the separation requirement. If the number of classes does not match the number of layers there might occur a transition zone in which 3 classes will be found simultaneously (two classes from neighbouring layers and one "transitional" class), this can be linked to smooth interfaces in the inverted data. Therefore a requirement of minimum distance of 2 meters between two valid interfaces has been introduced. This value has been selected experimentally and is arbitrary. If a pair of interface candidates fails this test, their points will be combined and new interface candidate will be created in their place. This step is repeated until all pairs of candidates pass the separation test.

At this point interface candidates turn into detected interfaces.
The final step is to compare the points of the detected interfaces with the true depth (in case there are multiple, the closest match will be taken into consideration). Next the root mean squared error (RMSE) will be calculated. This value will be passed further and will be used to compare performance of the classifier for prediction with different profiles.
While interpreting accuracy estimation evaluated by this method its limitations have to be taken into consideration. As this algorithm has been developed and implemented strictly for the purposes of this project, it has not been tested on a wide range of profiles. The interfaces detected by it may not be optimal ones. Due to this the potential for error in this method is greater compared to the ones discussed in previous sections, as it would combine errors of the classifier with errors of the interface detector. However for the profiles tested it has proven to be reliable. An example of how accurately the interface can be detected is shown in Figure 3-5.1.

3-5 Software and source code

The digital experiment performed in this project is implemented and executed using following open source software: Python [Van Rossum and Drake Jr, 1995], PyGIMLi [Rücker et al., 2017], Matplotlib [Hunter, 2007], Scikit-learn [Pedregosa et al., 2011], Pandas [Reback, 2020], SciPy [Virtanen et al., 2020] and Joblib [Joblib-Development-Team, 2020].
The source code developed to simulate the experiments and process the data can be found here.

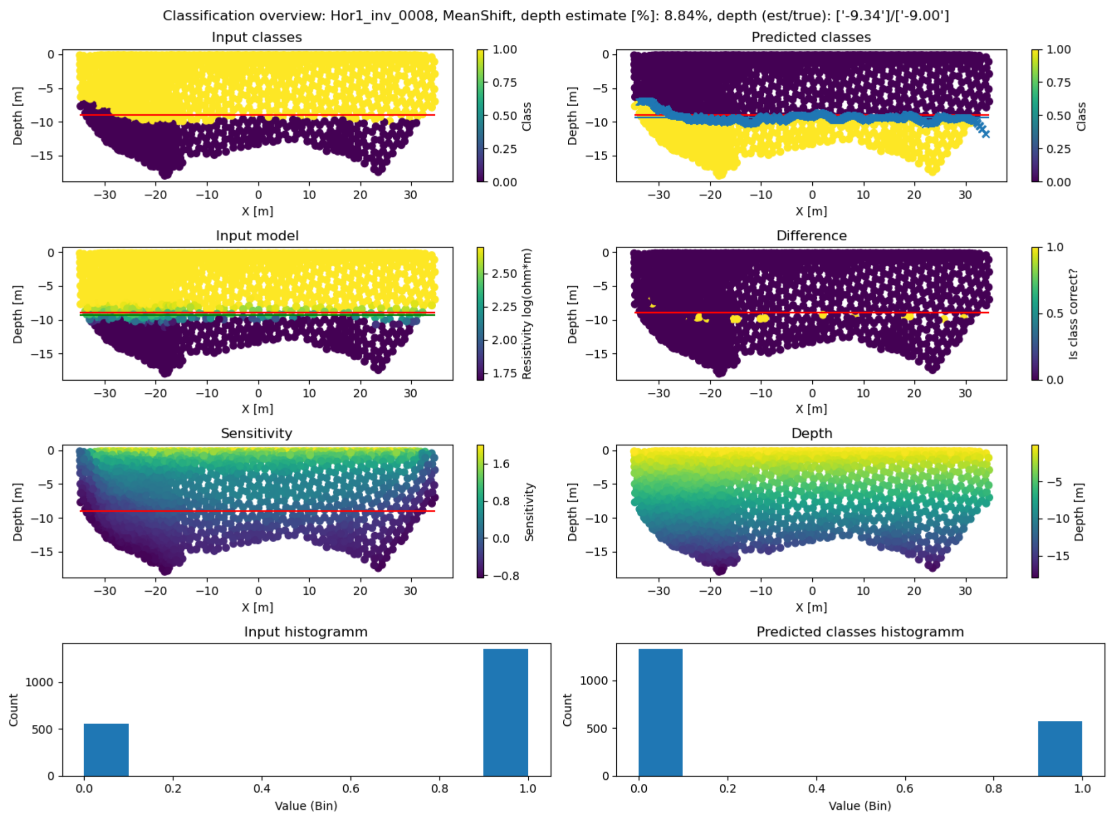


Figure 3-5.1: An example of an accurately detected interface (Experiment U17, profile Hor1_inv_0008, cluster: Mean Shift). The detected interface is marked by blue crosses in "Predicted classes" subplot. Red line indicates the true position of the interface.

Chapter 4

Results and discussion

Different clusters and classifiers have been tested with many combinations of type of training data and with different data processing methods disabled or enabled. For an overview of the best and worst results for the cluster and classifier analysis please see Table 4-1 and Table 4-2, respectively. Both of those tables contain results (evaluation of training and prediction accuracies) and information about the configuration with which those results have been achieved. The configuration is described in the tables with the following convention: Yes indicates that the specified method or data type has been used and No indicates the opposite. Column "Ref" indicates the configuration that differs by only one parameter (as a reference). It indicates that the configuration of specified experiment has been used as as starting point to investigate another parameter. For example experiment A36 is identical to experiment A21 in all parameters apart from application of sample weight. Each experiment changed one parameter in order to make an evaluation of the effect of this change. In order to reduce the amount of tests (performing experiments with all possible parameter combinations would be not feasible in terms of time and effort required to analyse and evaluate the results), specific variables were tested one at a time, and the best result of this test was used in following tests of a different parameter. The columns "score" and "interface" within the accuracy section of the table refer to the accuracy estimation methods described in Sections 3-4-1 and 3-4-2, respectively. The tested parameters can be split in two categories: (1) related to the input data, and (2) to data processing. The first ones define which features and what type of data is given to the classifier. While resistivity is always given, the effect of sensitivity and depth is tested. The resistivity can be normalized and/or have decimal logarithm applied to it. The latter option is applied always by default. The data processing option have been discussed in the previous chapter, and comprise: balancing, minimum sensitivity, hyperparameter tuning, and sample weight. As described in Section 3-2-1, "training type" refers to the type of resistivity contrast in the model (either

low to high, high to low, or mixed models).
In the next sections the following nomenclature will be used. The name of each experiments consists of a letter indicating the group to which it belongs, followed by a consecutive number of the experiment within each group. There are two groups of tests: experiments run with classifiers and clusters. Experiments employing classifiers are indicated by A (for cl**A**ssifiers), and experiments employing clusters with U (for cl**U**sters). For example, the name experiment name U25 indicates that this is the 25th experiment employing cluster analysis. If two experiments are referred to as a "pair", this mean that their configuration differs by one parameter only and that their comparison will show the impact of this parameter on the accuracy of prediction. For example, pair A14-A20 can be used to estimate influence of balancing on the result. In experiment A14 balancing is not used (No), while in A20 it is used (Yes).

4-1 Synthetic data

4-1-1 Clusters

Overview of the performed experiments

In total 48 experiments with using different input data and preprocessing steps have been conducted for the cluster analysis. An overview of the 10 best and the 10 worst results and parameters can be found in Table 4-1. The accuracies of the prediction of all experiments can be see in Figure 4-1.1.
It is important to consider that in cases with two classes, cluster have a strong tendency to consistently assigned opposite classes. As the goal of this study is to detect the interface in the subsurface the importance of correct class assignment is lesser than the correct detection of the interface. The latter can be achieved with classes assigned correctly and them being flipped. In first case the accuracy score will show high values close to 100%, in the second case the accuracy score will be very low and close to 0%. In both situations it is possible that the interface will be detected correctly hence both very high scores and very low ones can be considered as good.
In the Figure 4-1.1, the right half of the diagram shows a repetitive, almost periodic pattern. This seems to be correlate with the use of SEN (sensitivity) and Y (depth) as features, where higher accuracy is achieved when those data are not used as features in the training step.

4-1-2 Classifiers

Overview of the performed experiments

In total 42 experiments were conducted for the classifiers, testing the impact of different parameters on the prediction accuracy. Table 4-2 shows an overview of the 10 best and

the 10 worst results and parameters. The prediction accuracies of all experiments is shown in Figure 4-1.2.
As can be seen in Figure 4-1.2, experiments A13 to A25 generally provided the best results. For those experiments the accuracy score, which indicates the fraction of points that were classified correctly, is high, and interface detection accuracy shows the lowest values.

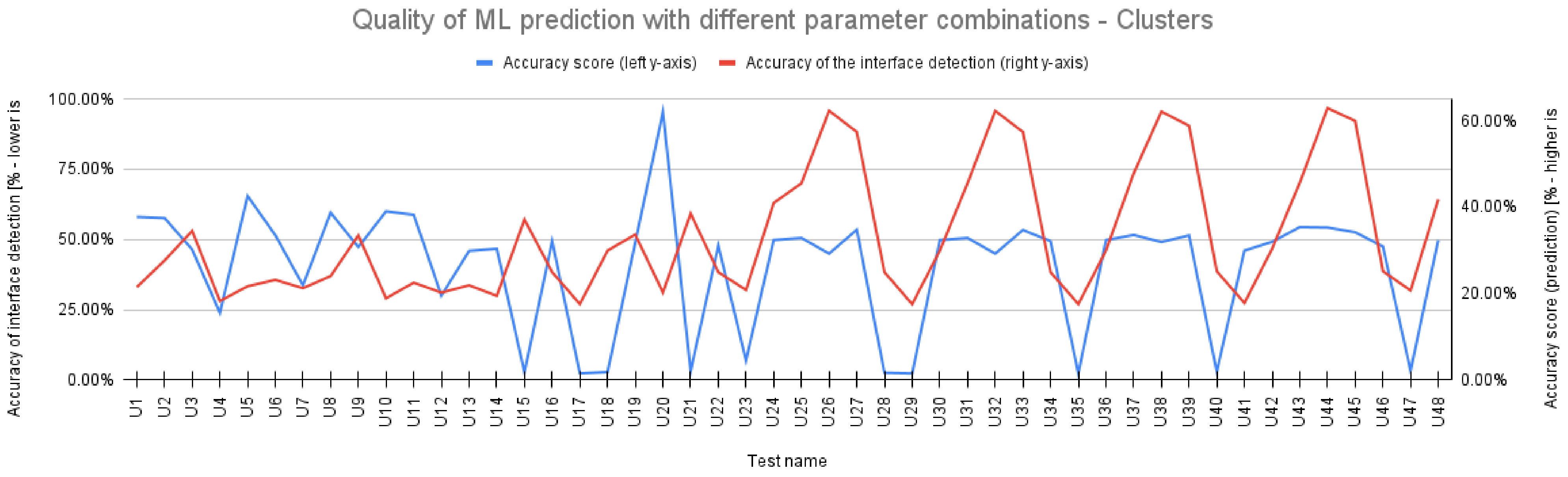


Figure 4-1.1: Accuracy of different experiments run with Clusters

Table 4-1: Overview of 10 best and 10 worst experiments with clusters. Sorted by accuracy of interface detection. Bal.: balancing; Min. Sen.: minimum sensitivity; Opti.: hyperparameter tuning; Sam. wght.: sample weight; P1: profiles with undetected interfaces; n test: number of tests (total); P2: profiles with undetected interfaces. Line with "-" indicate skipped experiments.

Test name	Ref	Data		Resistivity type		Train. type	Bal.	Min. Sen.	Opti.	Sam. wght.	Accuracy (training)		Accuracy (prediction)		Test name	P1	n tests	P2
		SEN	Y	Norm	log10						score	interface	score	interface				
U17	-	No	No	Yes	Yes	Normal	No	Yes	Yes	No	2.57%	15.94%	2.28%	17.54%	A17	1	126	0.79%
U29	U17	No	No	No	Yes	Normal	No	Yes	Yes	No	2.50%	14.85%	2.26%	17.54%	A29	1	126	0.79%
U35	U17	No	No	No	Yes	Normal	No	Yes	No	No	2.57%	15.94%	2.28%	17.54%	A35	2	126	1.59%
U41	U17	No	No	No	Yes	Normal	No	Yes	Yes	Yes	50.40%	16.97%	46.05%	17.80%	A41	2	126	1.59%
U4	-	No	No	No	Yes	Mix	No	Yes	Yes	No	24.01%	18.47%	23.90%	18.26%	A4	21	84	25.00%
U10	-	No	No	No	Yes	Mix	Yes	Yes	Yes	No	48.16%	22.77%	60.01%	18.90%	A10	23	84	27.38%
U14	-	No	No	Yes	Yes	Normal	No	No	Yes	No	48.75%	23.17%	46.71%	19.47%	A14	0	126	0.00%
U12	-	No	No	No	Yes	Inv	Yes	Yes	Yes	No	38.38%	18.14%	30.04%	20.22%	A12	46	126	36.51%
U20	-	No	No	Yes	Yes	Normal	Yes	No	Yes	No	95.32%	34.46%	95.43%	20.24%	A20	2	126	1.59%
U23	-	No	No	Yes	Yes	Normal	Yes	Yes	Yes	No	23.47%	6.98%	6.98%	20.82%	A23	2	126	1.59%
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
U43	U37	Yes	Yes	Yes	Yes	Mix	Yes	Yes	Yes	Yes	51.87%	56.05%	54.44%	45.66%	A43	2	84	2.38%
U37	U4	Yes	Yes	Yes	Yes	Mix	No	Yes	Yes	Yes	51.30%	53.50%	51.62%	47.80%	A37	1	84	1.19%
U27	U6	Yes	Yes	Yes	Yes	Inv	No	Yes	Yes	No	49.86%	41.32%	53.48%	57.40%	A27	0	126	0.00%
U33	U6	No	No	No	Yes	Inv	No	Yes	No	No	49.86%	41.23%	53.38%	57.40%	A33	2	126	1.59%
U39	U6	Yes	Yes	Yes	Yes	Inv	No	Yes	Yes	Yes	51.11%	38.66%	51.46%	58.83%	A39	0	126	0.00%
U45	U39	Yes	Yes	Yes	Yes	Inv	Yes	Yes	Yes	Yes	91.92%	42.33%	52.59%	59.95%	A45	0	126	0.00%
U38	U5	Yes	Yes	Yes	Yes	Normal	No	Yes	Yes	Yes	51.44%	44.12%	49.09%	62.11%	A38	0	126	0.00%
U32	U5	Yes	Yes	Yes	Yes	Normal	No	Yes	No	No	83.50%	39.54%	44.96%	62.28%	A32	0	126	0.00%
U26	U5	Yes	Yes	Yes	Yes	Normal	No	Yes	Yes	No	83.44%	39.41%	44.97%	62.31%	A26	0	126	0.00%
U44	U38	Yes	Yes	Yes	Yes	Normal	Yes	Yes	Yes	Yes	9.00%	38.75%	54.27%	62.91%	A44	0	126	0.00%

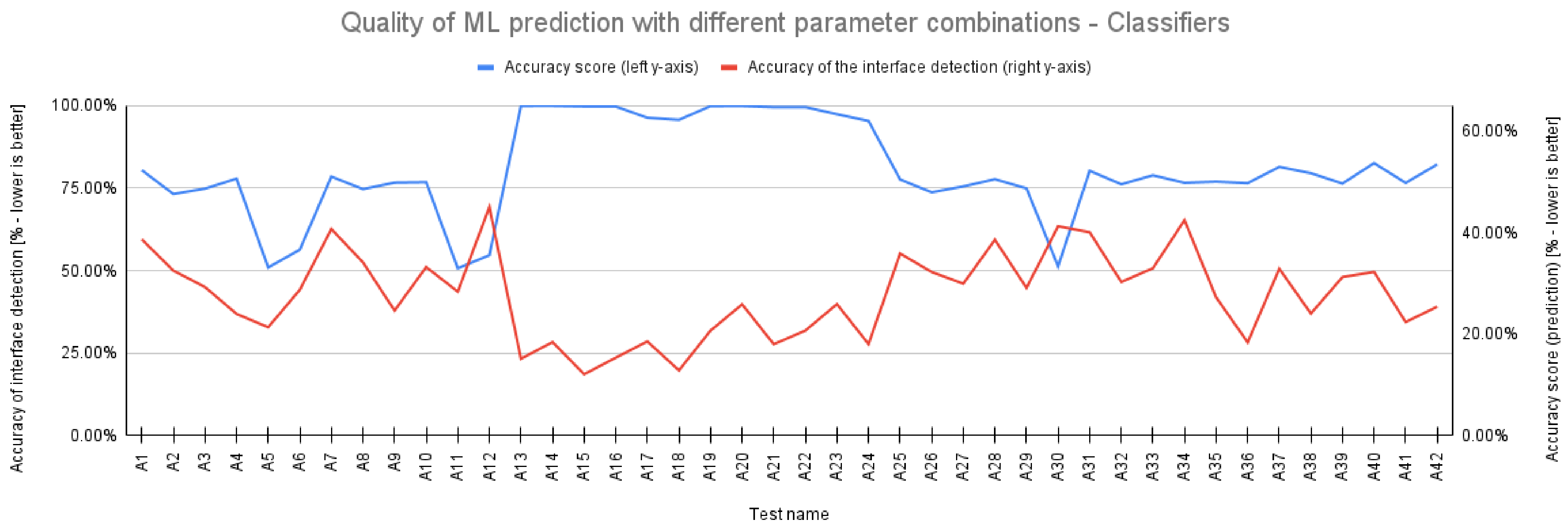


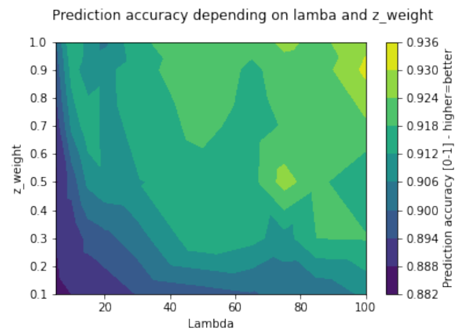
Figure 4-1.2: Accuracy of different experiments run with classifiers.

Table 4-2: Overview of 10 best and 10 worst experiments with classifiers. Sorted by accuracy of interface detection. Bal.: balancing; Min. Sen.: minimum sensitivity; Opti.: hyperparameter tuning; Sam. wght.: sample weight; P1: profiles with undetected interfaces; n test: number of tests (total); P2: profiles with undetected interfaces. Line with "-" indicate skipped experiments.

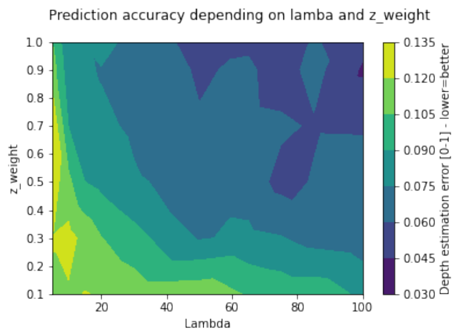
Test name	Ref	Data		Resistivity type		Train. type	Bal.	Min. Sen.	Opti.	Sam. wght.	Accuracy (training)		Accuracy (prediction)		Test name	P1	n tests	P2
		SEN	Y	Norm	log10						score	interface	score	interface				
A14	-	No	No	Yes	Yes	Normal	No	No	Yes	No	99.98%	18.42%	99.97%	19.60%	A14	6	378	1.59%
A20	-	No	No	Yes	Yes	Normal	Yes	No	Yes	No	99.94%	25.86%	99.93%	19.62%	A20	6	378	1.59%
A16	-	Yes	Yes	Yes	Yes	Mix	No	No	Yes	No	99.90%	15.35%	99.77%	21.09%	A16	6	252	2.38%
A19	-	No	No	Yes	Yes	Mix	Yes	No	Yes	No	99.95%	20.69%	99.90%	21.12%	A19	6	252	2.38%
A13	-	No	No	Yes	Yes	Mix	No	No	Yes	No	99.97%	15.14%	99.96%	21.16%	A13	6	252	2.38%
A22	-	Yes	Yes	Yes	Yes	Mix	Yes	No	Yes	No	99.89%	20.69%	99.59%	21.29%	A22	18	252	7.14%
A15	-	No	No	Yes	Yes	Inv	No	No	Yes	No	99.94%	12.09%	99.80%	26.04%	A15	0	378	0.00%
A21	-	No	No	Yes	Yes	Inv	Yes	No	Yes	No	99.90%	18.01%	99.58%	26.37%	A21	0	378	0.00%
A17	-	Yes	Yes	Yes	Yes	Normal	No	No	Yes	No	99.86%	18.55%	96.38%	27.96%	A17	18	378	4.76%
A23	-	Yes	Yes	Yes	Yes	Normal	Yes	No	Yes	No	99.91%	25.90%	97.45%	28.62%	A23	12	378	3.17%
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A36	A21	No	No	Yes	Yes	Inv	Yes	No	Yes	Yes	91.94%	18.37%	76.55%	59.88%	A36	74	378	19.58%
A33	A15	No	No	Yes	Yes	Inv	No	No	Yes	Yes	87.81%	32.98%	78.92%	59.91%	A33	65	378	17.20%
A9	-	No	No	No	Yes	Inv	Yes	No	Yes	No	91.30%	24.63%	76.71%	60.29%	A9	71	378	18.78%
A27	A15	No	No	Yes	Yes	Inv	No	No	No	No	86.13%	29.93%	75.56%	61.26%	A27	62	378	16.40%
A6	-	Yes	Yes	No	Yes	Inv	No	No	Yes	No	96.78%	28.72%	56.35%	61.46%	A6	24	378	6.35%
A30	A21	No	No	Yes	Yes	Inv	Yes	No	No	No	96.34%	41.24%	51.34%	61.87%	A30	19	378	5.03%
A10	-	Yes	Yes	No	Yes	Mix	Yes	No	Yes	No	91.83%	33.21%	76.85%	64.51%	A10	12	252	4.76%
A12	-	Yes	Yes	No	Yes	Inv	Yes	No	Yes	No	96.21%	45.04%	54.68%	67.82%	A12	50	378	13.23%
A5	-	Yes	Yes	No	Yes	Normal	No	No	Yes	No	97.92%	21.35%	50.97%	76.78%	A5	40	378	10.58%
A11	-	Yes	Yes	No	Yes	Normal	Yes	No	Yes	No	97.88%	28.37%	50.73%	81.41%	A11	9	378	2.38%

4-1-3 Dependency on the inversion parameters

Inversion parameters have an impact on the results of the training and the prediction. The inversion parameters matter because they control the quality and accuracy of the inversion result, which can improve or limit the performance of the classifiers. For almost any tool or procedure, it is a valid rule that the better the input data, the better the result of it. From this a hypothesis can be drawn, that each profile should be inverted with a specific set of parameters that will increase the accuracy of the inversion. However, there are some problems associated with this approach. The goal of this project is to investigate capabilities of ML techniques to detect interfaces. Inversion is a part of this study, but not the main one. As the duration of this project is limited, it is not feasible to search for optimal set of parameters for each profile separately. On the other hand, this kind of investigation may not be possible for real measurements because the true subsurface is not always know. Therefore it may be impossible to make a conclusive statement about which parameters fit a specific profile best. Keeping this in mind, it seems reasonable to assume that not all profiles provided to the classifiers are going to be inverted with the best possible set of parameters.
Nevertheless, to investigate the impact of various inversion parameters on the accuracy of the classification, different parameters were tested. Figure 4-1.3 shows how different combinations of the inversion parameters z_weight and λ change the accuracy of the prediction made by the same set of classifiers. Each experiment with one pair of z_weight and λ is represented in those diagrams by only one point. The value that it shows is an average of the accuracy of each classifier and the accuracy of each classifier is an average of the accuracy of the prediction made on all tested profiles, hence it shows an aggregated response of various classifiers. The following classifiers were used in this study: Adaboost, KNN, GBC, SGD, SVM and committee (see Section 2-2-4).
Comparing results shown in Figures 4-1.3a and 4-1.3b shows that the differences in the accuracy score are very small. Between the highest and the lowest value there are less than 5% difference. Still a slight preference for higher values for z_weight and λ can be inferred from these results. A potentail explanation for this might be an improved continuity of more smoothed interfaces.
A similar tendency can be found in Figure 4-1.3b. It shows the interface detection accuracy. Better accuracies are focused to higher values for both z_weight and λ. However the range of the accuracy is significantly larger. With a difference between best and worst score of around 10% it seems that the inversion parameters have a stronger influence over the accuracy of the interface detection compared to the accuracy score.
According to Figures 4-1.3a and 4-1.3b the best results are achieved with values of z_weight in the range between 0.7 to 1.0 and λ between 80 to 100.
Finally, for the following analysis two different pairs of z_weight and λ have been selected for the normal and inverse models. For the normal models it is 0.2 and 60, respectively, and for inverted ones 0.4 and 40. Those pairs are not falling within the best range defined by this parameter study, but they have shown a balance to provide detailed subsurface models, while limiting artifacts.



(a) Results of the sensitivity study evaluated with accuracy score.



(b) Results of the sensitivity study evaluated with interface detection accuracy.

Figure 4-1.3: Sensitivity study over influence of inversions parameter of z_weight and λ on accuracy of prediction. Evaluated by: (a) accuracy score (3-4-1) and (b) accuracy of the interface detection (3-4-2).

4-1-4 Impact of the training data

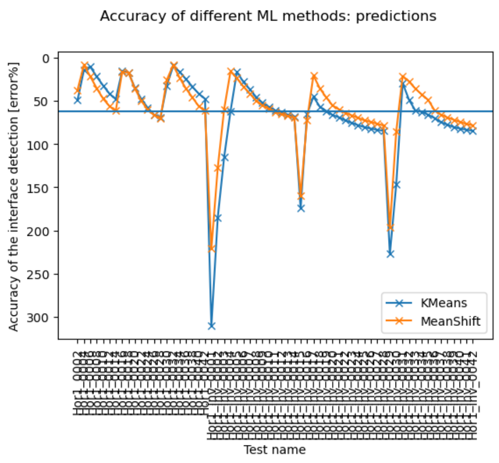
The classifiers and clusters can be trained on normal, inverted or mixed data (please see Section 3-2-1 and 3-2-1 for more details).
Experiments can be grouped into groups of three. Within each group there are experiments run with clusters or classifiers trained on different sets of training profiles (trained on mixed, normal, or inverse profiles) but with the same methods applied to them (configuration). There are two full groups in the top 10 results: A13-A14-A15 and A19-A20-A21. In terms of accuracy in those groups training on normal models results in the best score and training on inverted models results in the worst score. Considering all experiments and not considering differences resulting from other parameters, generally training on inverted models performs consistently worse than on normal or mixed models, which tend to perform equally well.
Among experiments run with clusters the best results are achived when they are trained on normal profiles (see Table 4-1). Using mixed types of profiles seems to deliver good result as well, but is clearly outperformed by the normal models. Inverted training profiles are represented only by one good score.
This noticeable difference in preference between clustering and classification can be explained by differences in how the data is assessed and how classes are assigned within the data set. For the classification, a lower class number is always assigned to an area with lower resistivity, while clusters rely more on internal similarity and coherence of the points contained in each cluster. Therefore, for the clusters there is only a small difference between normal and inverted models. The points in the resistive and conductive layers will share the same similarity regardless of their spatial position. The only difference can be related to the quality of the inversion. Lower layers will be resolved more poorly compared to the upper ones due to a decreasing sensitivity with depth. It may seem that this would suggest a preference for training on the mixed dataset. This

may not necessarily be true, for two reasons. Using mixed datasets will include errors specific to both normal and inverted profiles. To some extent it may be beneficial, as the cluster or classifier will have a chance to learn those imperfections too and adjust to them. But on smaller and more self similar datasets, this benefit may not outweigh the complexity introduced by it. Considering normal and inverted profiles, normal models provide better results than the experiments using inverted models. This is likely being caused by the inversion process. For the inverted models the more resistive layer will be on top. This may result in smaller sensitivity of cells deep in the profile (in lower more conductive layer).
Classifiers face a different problem. Since the points are assigned to classes based on their resistivity, not depth or spatial position, training them on mixed dataset may create additional complexity. Especially if depth is included as a feature for both training and prediction. Using only one type of data for training may result in clearer division between classes and still provide enough generalization to assign points correctly. This explanation seems to be supported by the higher accuracy that was achieved when using resistivity as the only input feature, compared to the experiments when either or both of depth and sensitivity were used.

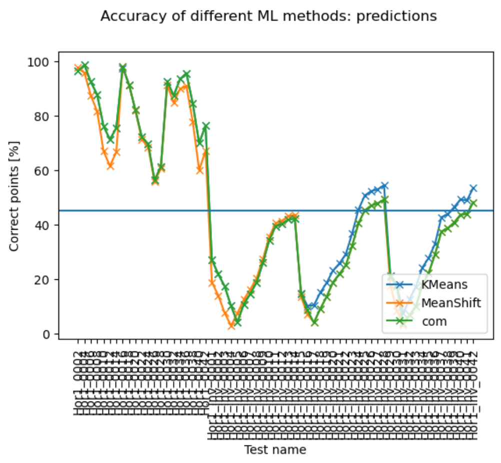
4-1-5 Impact of number of parameters

The impact of the number of features included (whether SEN and Y are used) differs between clusters and classifiers.
The clusters show a very well defined response. Among the best 10 results there are none that would benefit from information given by SEN or/and Y. Rather on the opposite, the accuracy may decrease significantly when SEN and Y are included as features. For example, for pair U17-U26, the error of interface detection increased from 17.56% to 62.31%. U17 can be found in Figure 4-1.13 and U 26 4-1.4. Interestingly, the tendencies of accuracy achieved by U17 and U26 for the first 20 profiles are exactly opposite. For U17 the accuracy increases with depth and for U26 it decreases.

A possible explanation to why clusters perform better using only resistivity as a feature is that there is sufficient amount of similarity and coherence between points only in terms of their resistivity. If this is the case, adding more information may create more complexity. The clusters may try to bring together points that have similar SEN and Y values but different RES, for example two points at similar depth. Those points can seem similar in terms of those two parameters (SEN and Y), but strongly differ in their resistivity if they are on different sides of an interface. In this situation one of them may be misclassified, which would not have happened if only RES was taken into consideration. A similar tendency, i.e. to prefer training and prediction on the resistivity only, can be found among the classifiers. However unlike clusters, they have achieved good accuracy scores with those features too. The third best result of classifiers have been achieved using both SEN and Y. That experiment (A17, Figure 4-1.6) performed significantly worse than its pair (A14, Figure 4-1.9). Different result can be found in



(a) Clusters U26 - interface detection accuracy.



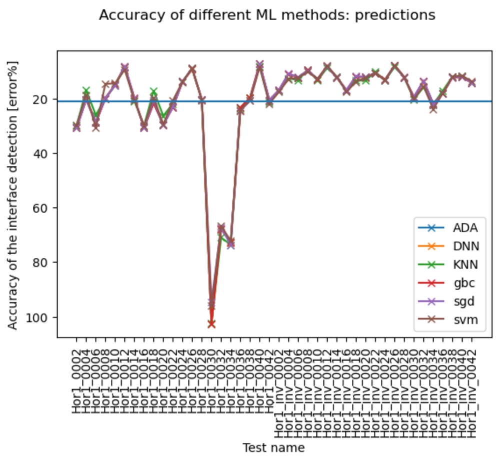
(b) Clusters U26 - prediction accuracy.

Figure 4-1.4: Accuracy estimations for classifiers test U26: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

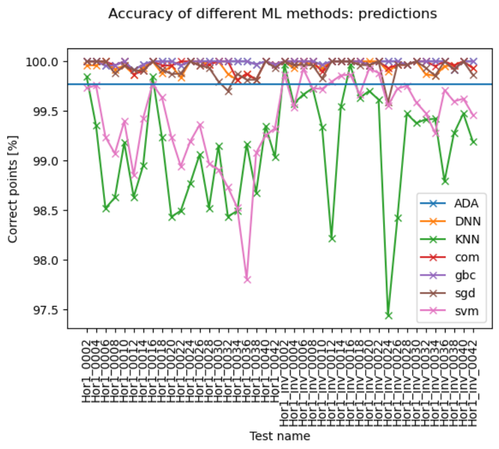
the pair A13-A16. Their performance is very similar, the difference in the interface detection accuracy is 0.07 percentage point. A far greater difference can be found in pair A20-A23 (Figures 4-1.10 and 4-1.7 respectively). Generally the results achieved by both of them are similar if not the same. The only difference is an outlier - KNN. It is caused by misclassified points at the bottom of the profile that created a false interface. But as Figure 4-1.8 shows, the depth of the true interface was estimated with a good accuracy. Generally it can be said that classifiers have a preference for not including SEN or Y features. This is similar to the results of clusters but contrary to them, classifiers can achieve good results while including additional features. There are some exceptions (e.g.: KNN in A13-A16 pair). But comparison of pairs shows clear preference.

4-1-6 Balancing

For both clusters and classifiers class balancing does not provide any benefit to the learning.
This result may be slightly counter intuitive as the class inbalance is a common problem in ML classification. However, after the training profiles are combined together, the number of points in each class may be sufficient for the classifier to learn about all classes. The profiles used in this project have diverse interface depths and since those different depths are split evenly between training and prediction phases, the whole collection of profiles may be balanced. Hence, balancing should not have a negative effect on the overall accuracy, and we would predict that the best results should be

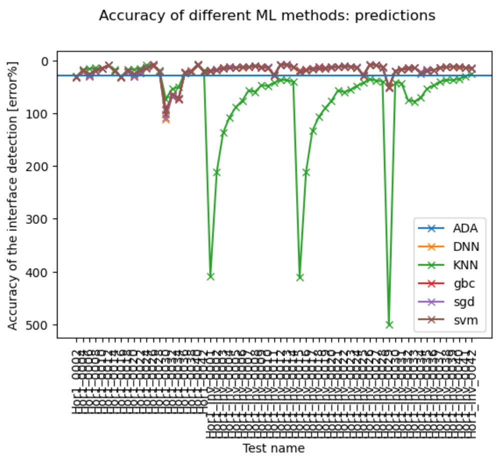


(a) Classifiers A16 - interface detection accuracy.

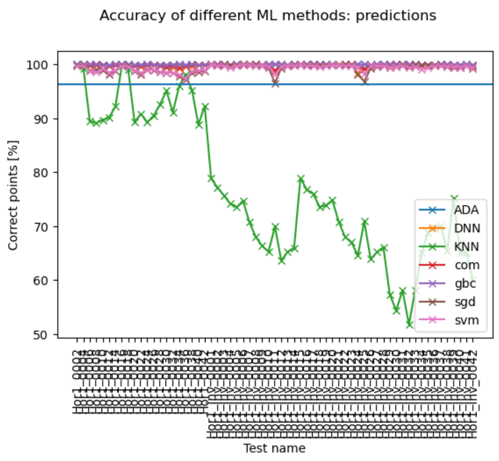


(b) Classifiers A16 - prediction accuracy.

Figure 4-1.5: Accuracy estimations for classifiers test A16: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

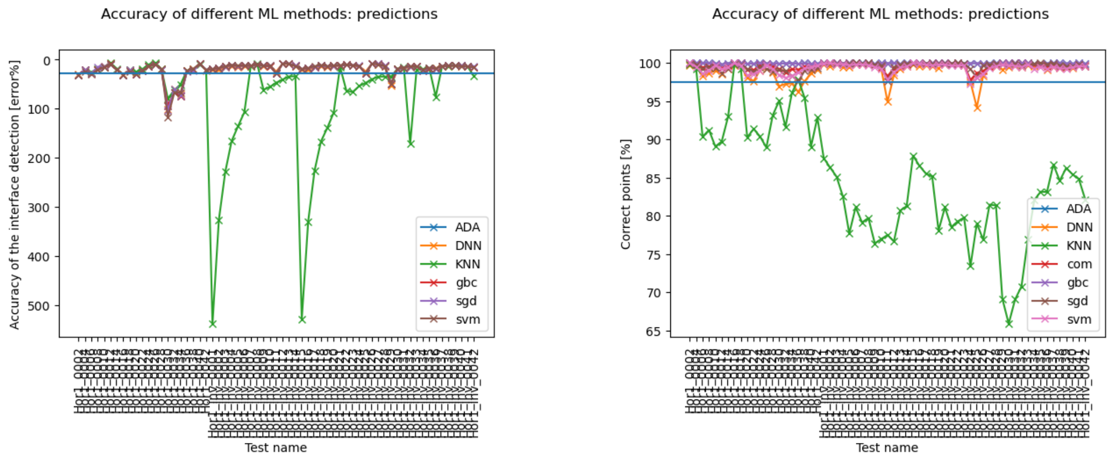


(a) Classifiers A17 - interface detection accuracy.



(b) Classifiers A17 - prediction accuracy.

Figure 4-1.6: Accuracy estimations for classifiers test A17: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).



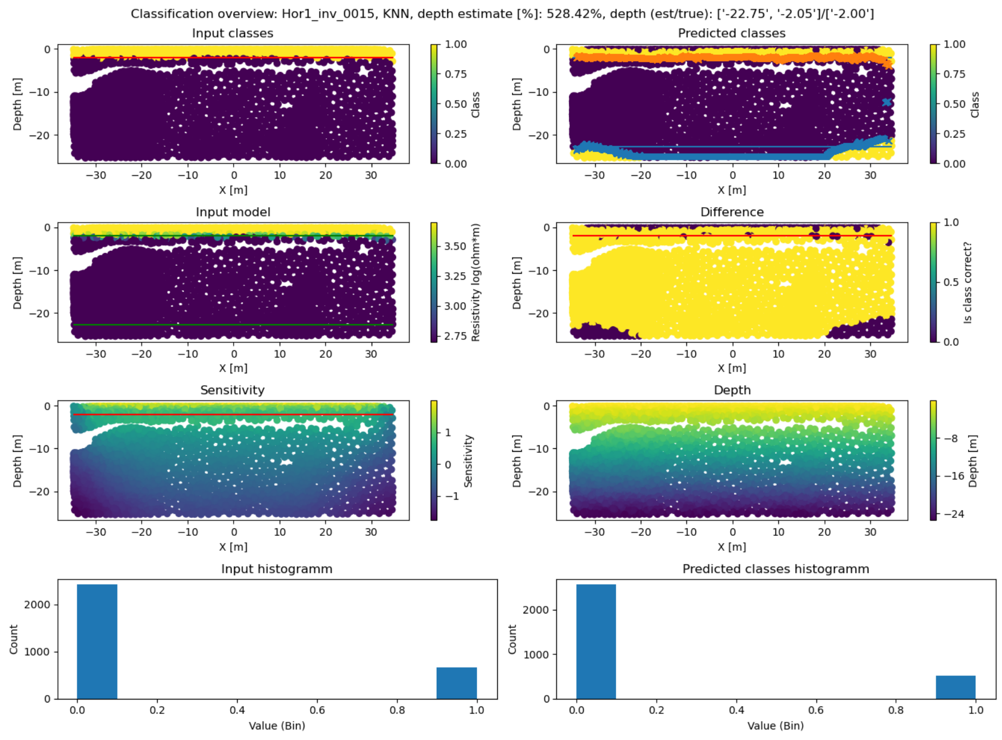
(a) Classifiers A23 - interface detection accuracy.

(b) Classifiers A23 - prediction accuracy.

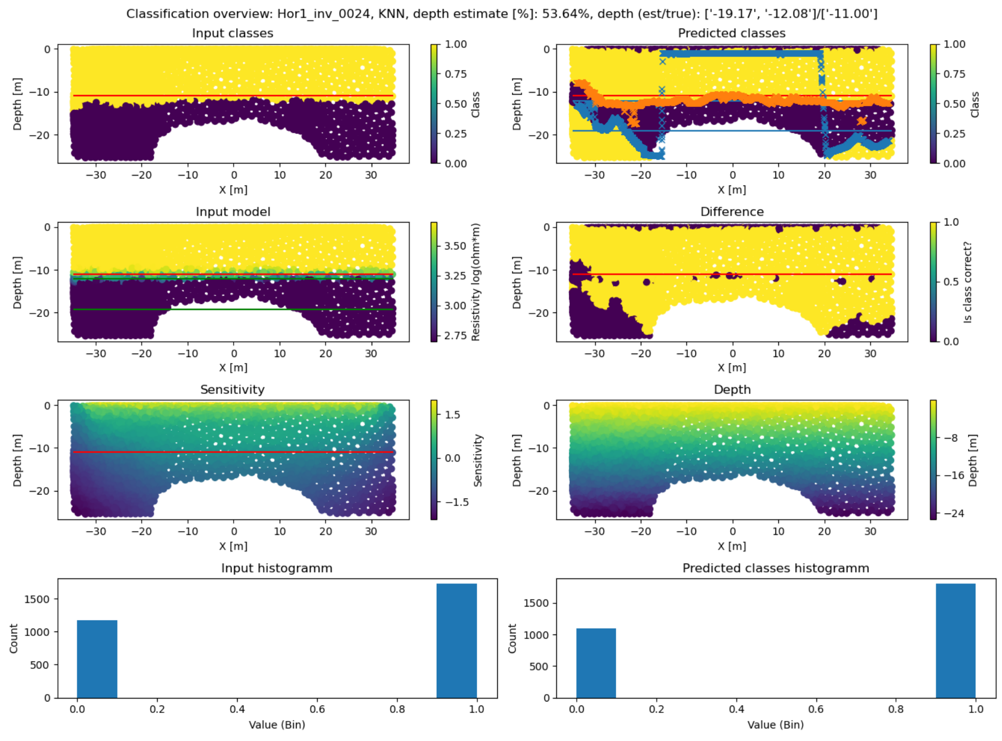
Figure 4-1.7: Accuracy estimations for classifiers test A23: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

achieved either with or without class balancing. This is the case for the classifiers. Exactly 5 out of 10 results do use balancing and they are spread evenly among the best results. Table 4-2 highlights these results, as it shows pairs of experiments where the only difference is whether or not class balancing was applied. Those pairs are: A14-A20, A16-A22, A13-A19, A15-A21, A17-A23 (in each pair first experiment is executed without the balancing and second with it). In all pairs but A13-A19, the results obtained without balancing are slightly better. Those differences are small. They range between 0.02 percentage points (A14-A20) to 0.2 percentage points (A16-A22). Comparing in detail the accuracies of experiments A14 (Figure 4-1.9) and A20 (Figure 4-1.10) shows no visible difference for the interface detection accuracies. There are slight differences between classifiers in terms of the accuracy score, but they have minimal influence on the final, averaged score. Slightly easier to notice is the difference between experiments A13 and A19. A19 (with balancing) detects interface more accurately by 0.04 percentage points, which is almost negligible. Comparing Figures 4-1.11a and 4-1.12a shows that for A19 the lines for different classifiers are less overlapping. It can mean that balancing of classes introduced some degree of confusing as this effect is not visible in experiment A13 run with the same configuration but without balancing. This is specifically visible between profiles Hor1_0030 to Hor1_0034. Overall the differences are minor. Regarding the accuracy score, in experiment A19 the average value is slightly better compared to A13. Based on this it can be concluded that for experiments with classifiers balancing has no influence. Neither positive nor negative one.

Whether balancing has a positive, negative, or no impact on clustering is more difficult

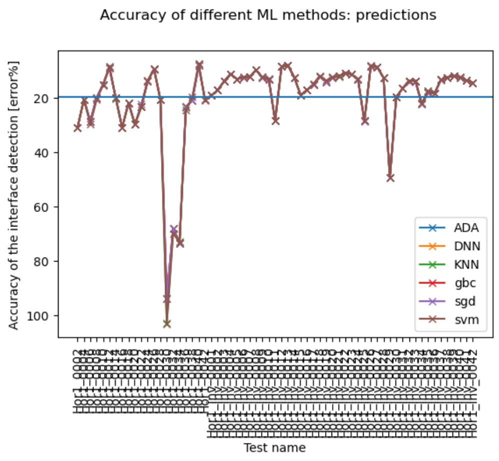


(a) Classifiers A23 - Hor1_inv_0015.

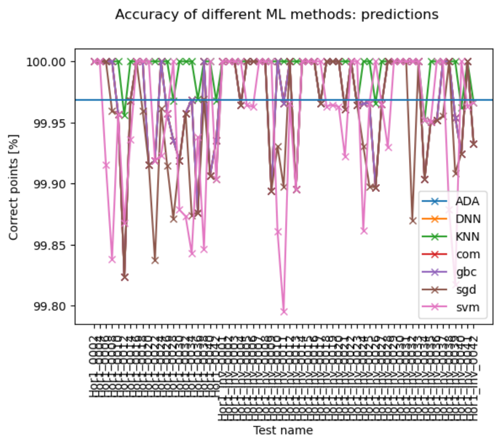


(b) Classifiers A23 - Hor1_inv_0024.

Figure 4-1.8: Experiment A23: (a) profile Hor1_inv_0015 and (b) profile Hor1_inv_0024

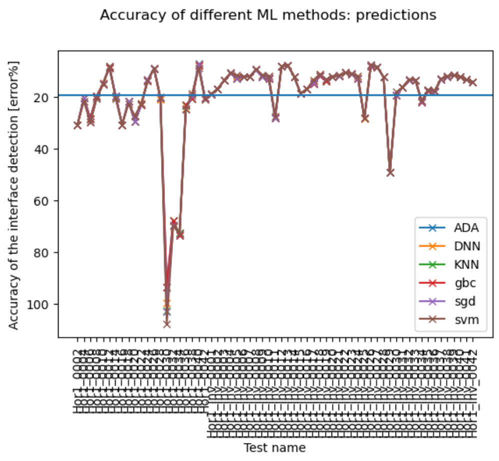


(a) Classifiers A14 - interface detection accuracy.

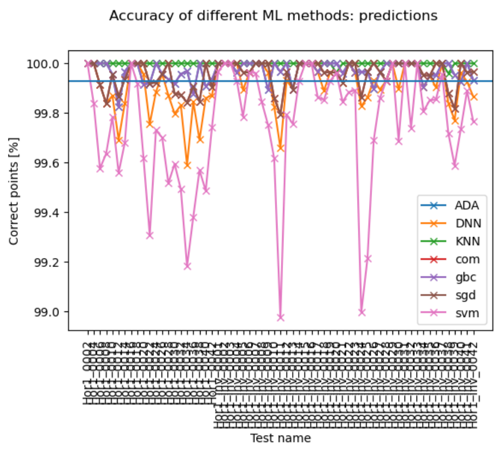


(b) Classifiers A14 - prediction accuracy.

Figure 4-1.9: Accuracy estimations for classifiers test A14: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

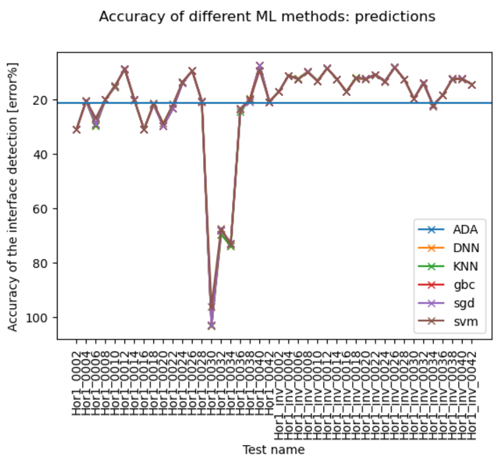


(a) Classifiers A20 - interface detection accuracy.

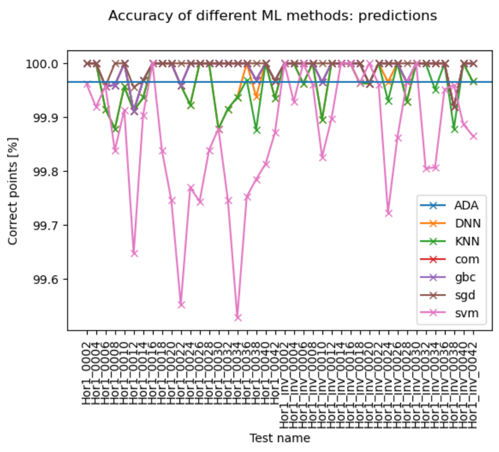


(b) Classifiers A20 - prediction accuracy.

Figure 4-1.10: Accuracy estimations for classifiers test A20: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

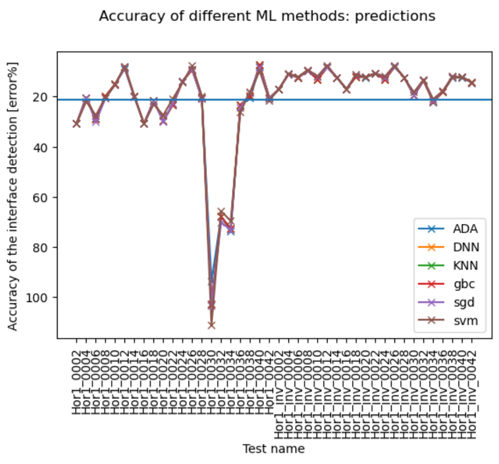


(a) Classifiers A13 - interface detection accuracy.

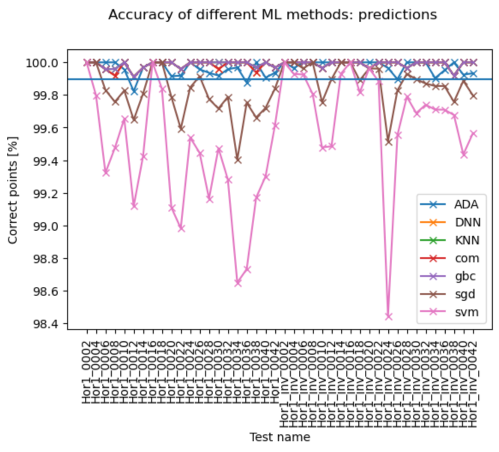


(b) Classifiers A13 - prediction accuracy.

Figure 4-1.11: Accuracy estimations for classifiers test A13: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).



(a) Classifiers A19 - interface detection accuracy.



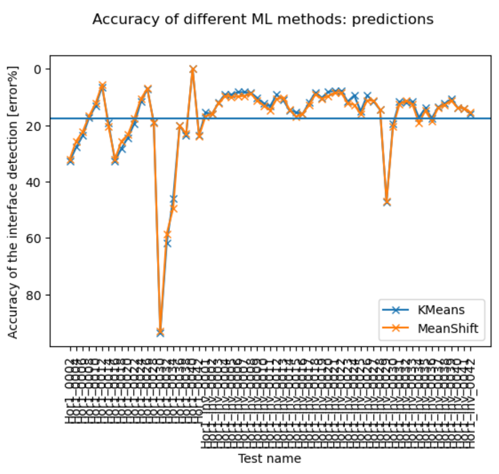
(b) Classifiers A19 - prediction accuracy.

Figure 4-1.12: Accuracy estimations for classifiers test A19: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

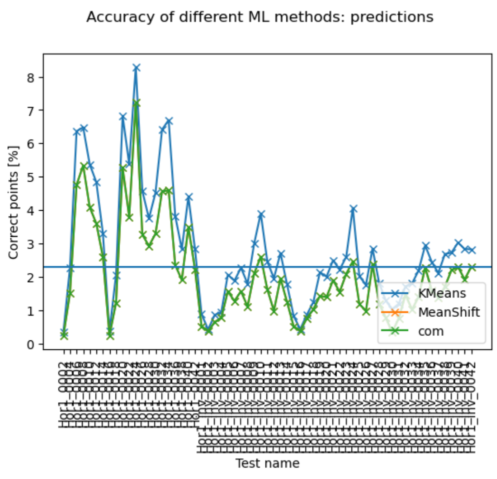
to judge. 4 out of the 10 best results use balancing, but their results are worse than those that are not using balancing (Please see Table 4-1). As for classifiers, we can compare pairs of results with the only difference whether class balancing was applied. These pairs are: U4-U10, U14-U20 and U17-U23. The differences in the accuracies within pairs are larger than for classifiers. For the first pair it is 3.28 percentage points, which is the largest difference. The smallest difference is for pair U29-U41 (0.26 percentage points). In the first pair (U17-U23) it can be observed that adding balance has a strongly negative impact on MeanShift. Figures 4-1.12a and 4-1.14a show the decrease in interface detection accuracy reaching up to 15%. There are some profiles for which the decrease was negligible (for example profiles Hor1_0006 to Hor1_0018). Similar effects can be observed for experiments U4 and U10 (Figures 4-1.15a and 4-1.16a). However, for some profiles balancing actually improves the accuracy. For example, profile Hor1_0030 has a better prediction with KMeans and MeanShift if balancing is applied. The improvement is significantly stronger for the latter one. Applying balancing increased the number of profiles with undetected interfaces by one for the prediction with KMeans and decreased it by one for MeanShift. Even harder to observe are differences in pair A14-A20. There is no change to the result of KMeans, but the MeanShift result is better in the experiment without balancing. Figures 4-1.20 shows one more interesting consequence of applying balancing. In Figure 4-1.20b (experiment U20 - with balancing) the classes have been assigned correctly contrary to the results from experiment U14, which ran without balancing (Figure 4-1.20a). This is not a separate, exceptional case. Figures 4-1.18b and 4-1.19b show this as well. Figures 4-1.15a, 4-1.16a, 4-1.13a, 4-1.14a show that inversion of the assigned classes is not a consistent result of applying balancing to the data, though it happens in all of the pairs.
Overall it seems, that balancing does have a weak negative effect at the accuracy of clusters prediction. This is specifically true for MeanShift. KMeans seems to be mostly unaffected.

4-1-7 Minimum sensitivity

The application of minimum sensitivity threshold yields results that require careful investigation. The results for clusters and classifiers are the exact opposites.
Clusters clearly benefit from a minimum sensitivity threshold (Table 4-1). There are two pairs in top 10 results: U14-U17 (Figures 4-1.18 and 4-1.13) and U20-U23 (Figures 4-1.19 and 4-1.14). In the first pair, better accuracy was achieved with minimum sensitivity applied. Interestingly, minimum sensitivity has removed the problem of inverted class assignment in experiment U14. This has improved the accuracies achieved by the worst outliers and some of the relatively good profiles. The biggest improvements can be found among the normal profiles. Two inverted profiles in which no interfaces has been detected were improved enough to make detection possible. On the other hand, in one of the normal profiles a previously detected interface was lost. Overall the difference in accuracy between U14 and U17 is 1.93 percentage point. The opposite can be observed in pair U20-U23. In this pair, U20, where no minimum sensitivity was applied, offers

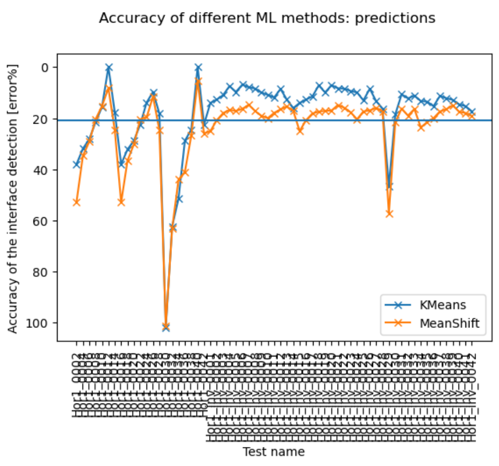


(a) Clusters U17 - interface detection accuracy.

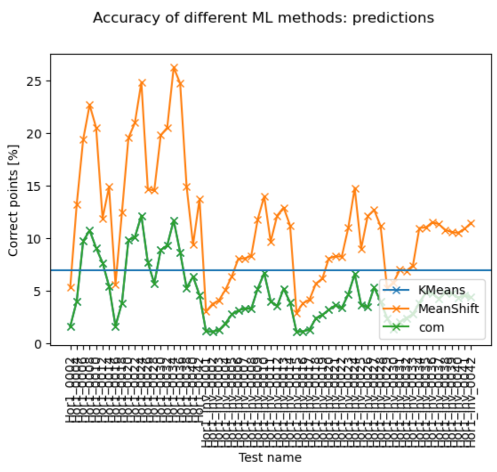


(b) Clusters U17 - prediction accuracy.

Figure 4-1.13: Accuracy estimations for clusters test U17: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

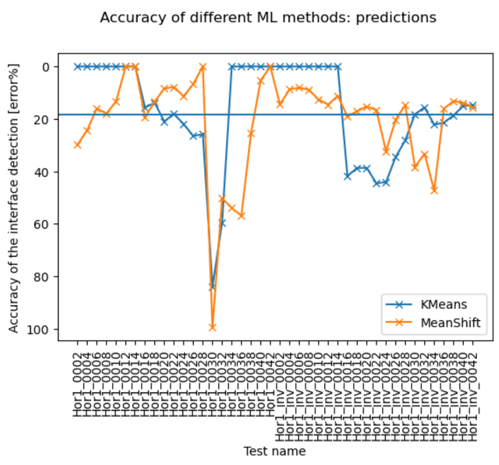


(a) Clusters U23 - interface detection accuracy.

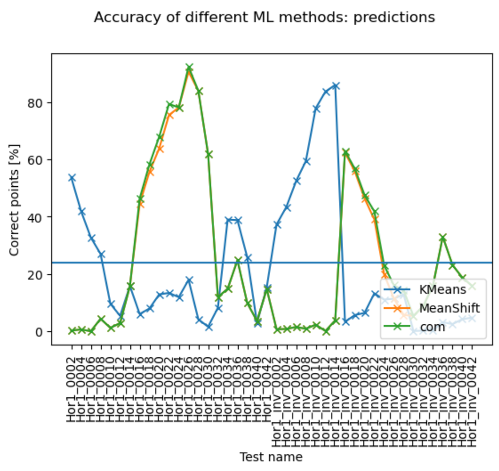


(b) Clusters U23 - prediction accuracy.

Figure 4-1.14: Accuracy estimations for clusters test U23: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

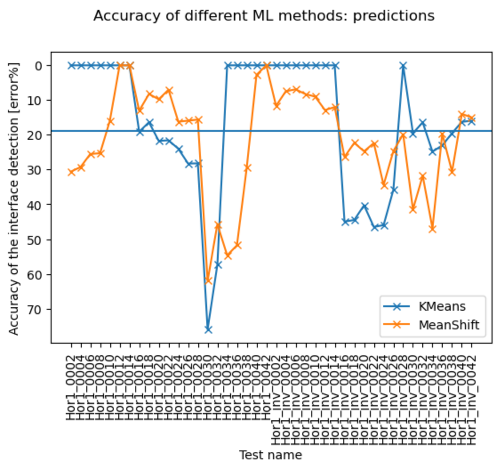


(a) Clusters U4 - interface detection accuracy.

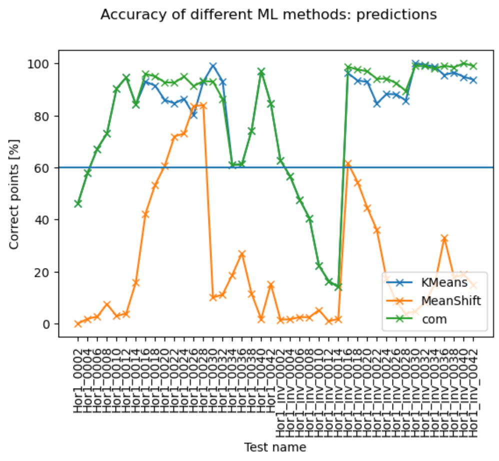


(b) Clusters U4 - prediction accuracy.

Figure 4-1.15: Accuracy estimations for clusters test U4: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

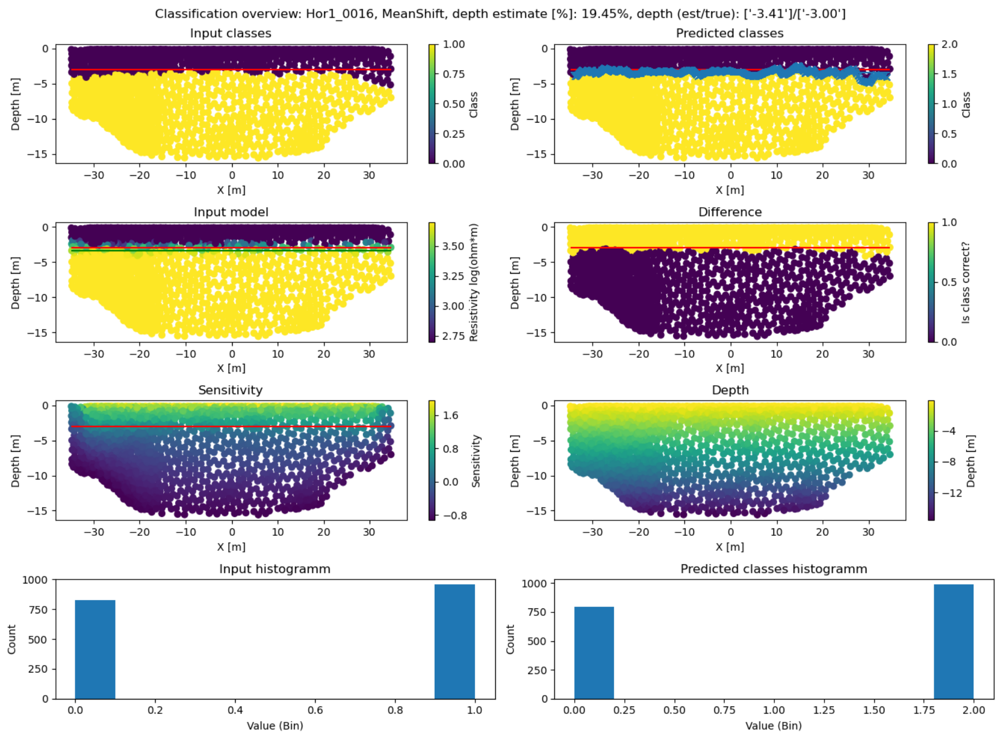


(a) Clusters U10 - interface detection accuracy.

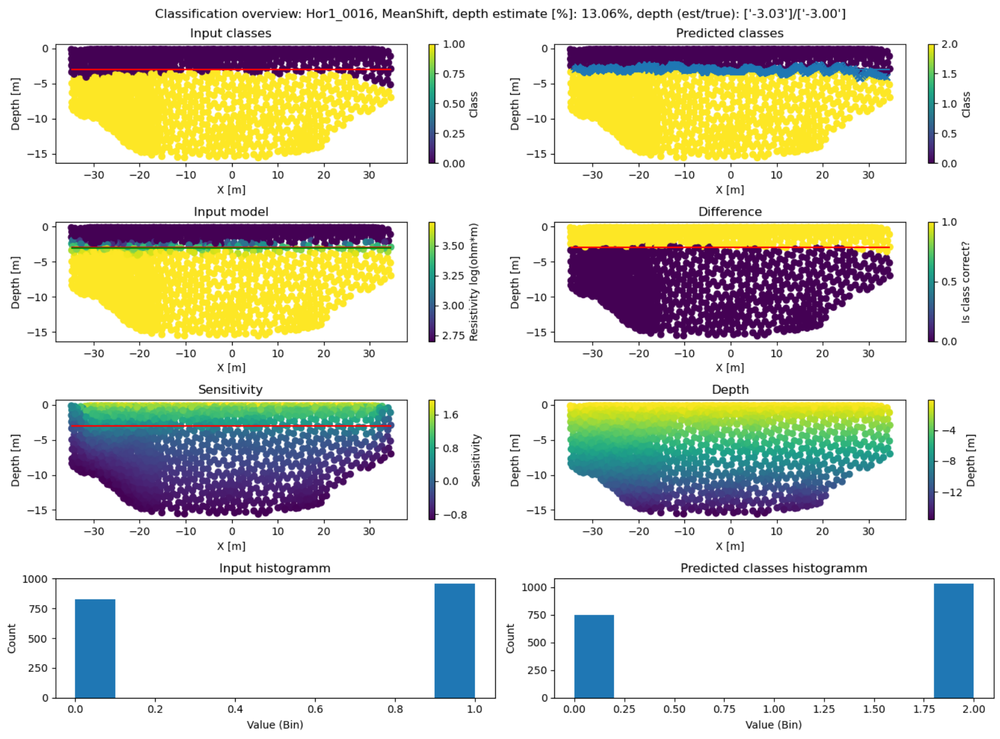


(b) Clusters U10 - prediction accuracy.

Figure 4-1.16: Accuracy estimations for clusters test U10: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

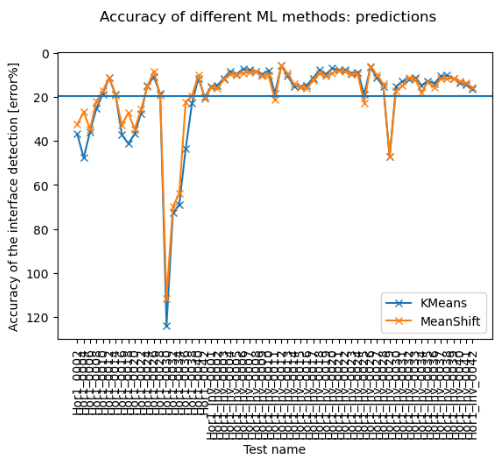


(a) Clusters U4 - interface detection - Hor1_0016.

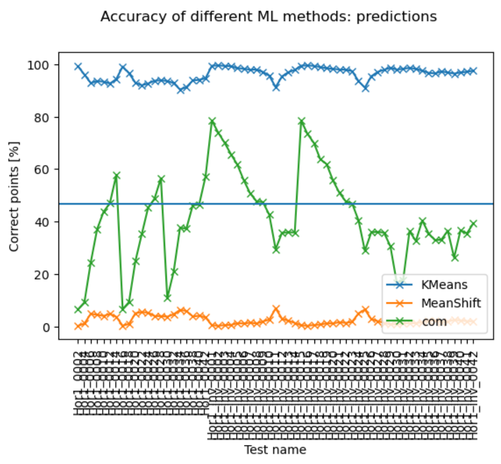


(b) Clusters U10 - interface detection - Hor_0016.

Figure 4-1.17: Profile Hor_0016 prediction overview from experiments (a) U4 and (b) U10.

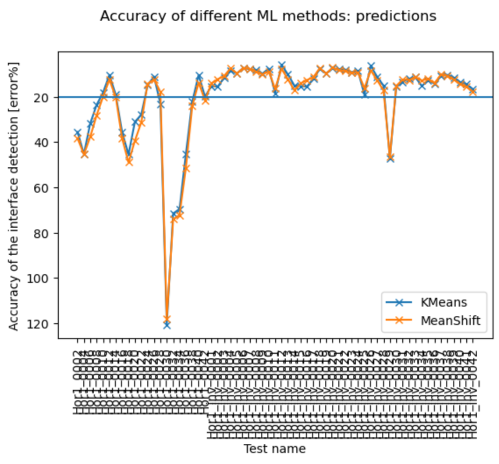


(a) Clusters U14 - interface detection accuracy.

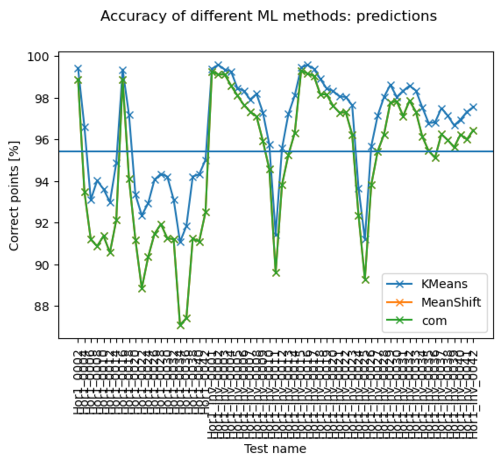


(b) Clusters U14 - prediction accuracy.

Figure 4-1.18: Accuracy estimations for clusters test A14: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

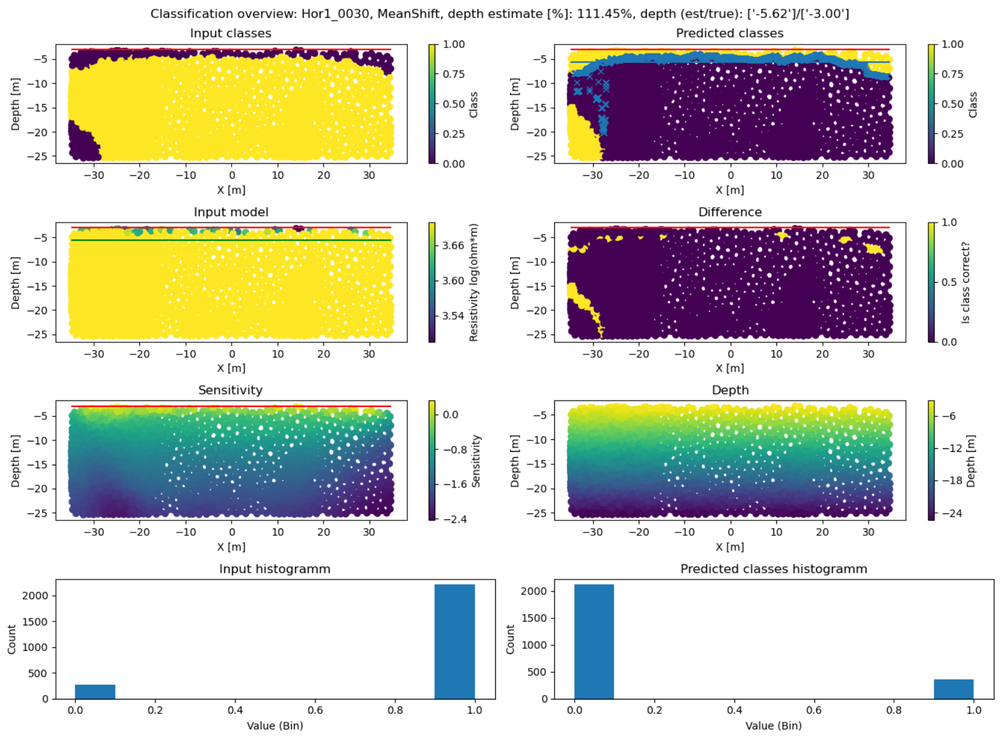


(a) Clusters U20 - interface detection accuracy.

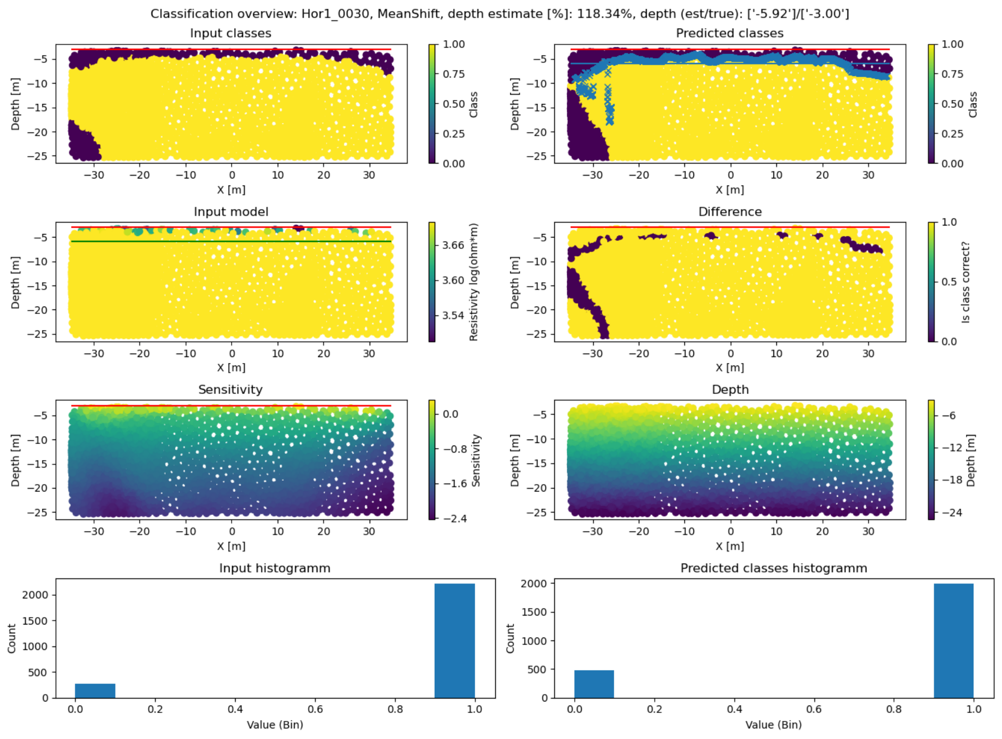


(b) Clusters U20 - prediction accuracy.

Figure 4-1.19: Accuracy estimations for clusters test U20: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).



(a) Clusters U14 - interface detection - Hor1_0030.



(b) Clusters U20 - interface detection - Hor_0030.

Figure 4-1.20: Profile Hor_0030 prediction overview from experiments (a) A14 and (b) A20.

better a slightly better accuracy. Considering that between two pairs in top 10 only in one of them experiment without minimum sensitivity delivered better results and other 8 experiments have pairs (experiments without minimum sensitivity applied) outside of top 10 (worse performance) it can be said that clusters quite strongly benefit from minimum sensitivity. A possible explanation for this is that if less informative values (and less reliable ones) are removed it is easier for the cluster to group the remaining points based on their self similarity. A similar explanation has been proposed in another section discussing the influence of the number of features on the accuracy of clusters (please see Section 4-1-5).
For experiments run with classifiers the influence of the minimum sensitivity threshold is also clear. None of the experiments in top 10 applied minimum sensitivity. For the top 10 results, pairs where the sensitivity threshold was applied performed significantly worse. Investigation of pair A14-A38 shows clearly that applying minimum sensitivity impacted performance of the classifiers in a negative way. SGD lost its capability of detecting the interface almost in half of the profiles and all other classifiers accuracy decreased.

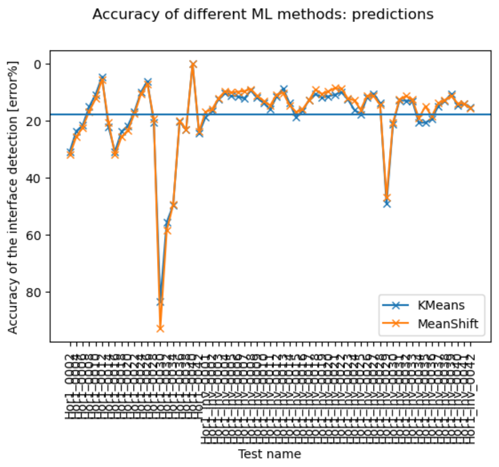
4-1-8 Hyperparameter tuning

Tables 4-1 and 4-2 show clearly that hyperparameter tuning improves accuracy of the prediction.
Comparing results achieved by pairs A14-A26, the error is reduced by more than half as a result of hyperparameter tuning. It improves not only the chances of detection of the interface and its accuracy, but also the consistency of the performance of the classifiers. Figure 4-1.9 shows that the results achieved by the classifiers overlap.

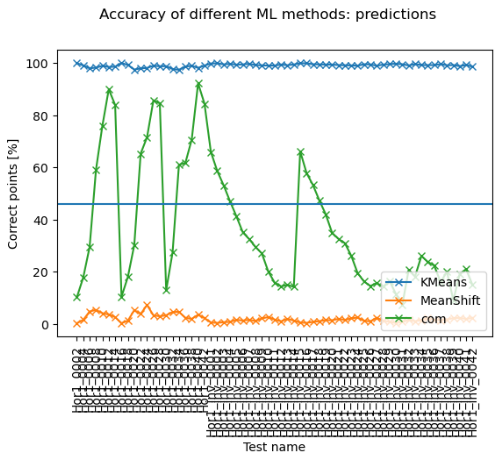
The results achieved by clusters are similar. Hyperparameter tuning clearly improves the accuracies of prediction with clusters. There is one outlier in the top 10 but it is similar in its configuration to the best experiment. Apart from experiment U35, all other ones run without hyperparameter tuning are scattered in the lower half of the ranking. It is clear that hyperparameter tuning helps both clusters and classifiers achieve their best performance.

4-1-9 Sample weighting

The influence of sample weighting is mostly negative on the accuracy of both clusters and classifiers.
Among classifiers there is no experiment in the top 10 that benefited from applying sample weighting.
For clusters, only one experiment achieved a high accuracy (U41), but it still performed worse than its pair (U17), although the difference is very small (0.26 percentage point).



(a) Clusters U41 - interface detection.



(b) Clusters U41 - interface detection.

Figure 4-1.21: Accuracy estimations for clusters test U41: (a) interface detection accuracy (lower value is better); (b) accuracy score (higher value is better). On y-axis there are names of profiles, the interface depths increases from left to right (within both normal profiles and inverted ones (longer names with inv in it)).

Comparing results of U17 (Figure 4-1.13) and U41 (Figure 4-1.21) shows very little difference. Only two profiles have been predicted slightly better without sample weighting: Hor1_inv_0024 and Hor1_inv_0026 but only for prediction with KMeans. For some profiles (e.g.: Hor1_inv_0004 to Hor1_inv_0010) applying sample weighting slightly decreased the accuracy. But there are profiles which benefited from it: e.g. Hor1_0030.
Those local improvements are not enough to shift the results in favour of sample weighting and it can be said that it consistently decreases the accuracy of both clusters and classifiers.

4-1-10 Borehole simulation

The exact impact of the borehole simulation has not been tested. As it was discussed in Section 3-3-4 the high computational cost of training on full datasets made it not feasible to run experiments without borehole simulation applied. In earlier experiments, run with far smaller data sets it was verified that applying borehole simulation has no negative effect on the results of the experiments. In some cases it even improved them. However there is a problem that should be taken into consideration.
The issue is the way how borehole is defined for the purpose of the simulation. In a perfect situation, borehole would indeed be a rectangular, narrow and vertically elongated rectangle. However, this is usually not the case. Boreholes are rarely straight, there can be a difference between borehole depth and so called true vertical depth (TVD) [Ellis and Singer, 2007]. This leads to a question how well can

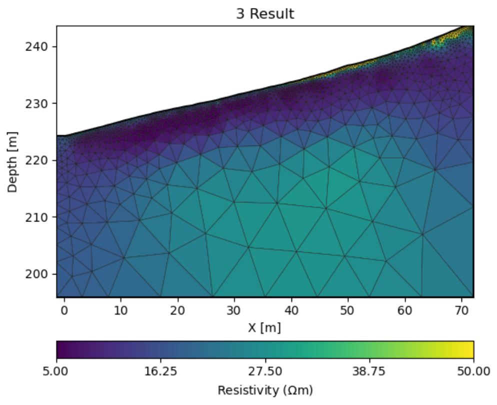
this simplified, perfect definition of a borehole approximate training on a data from a real one, where there may be some inaccuracies in the depth of the measured resistivities.

4-1-11 Summary

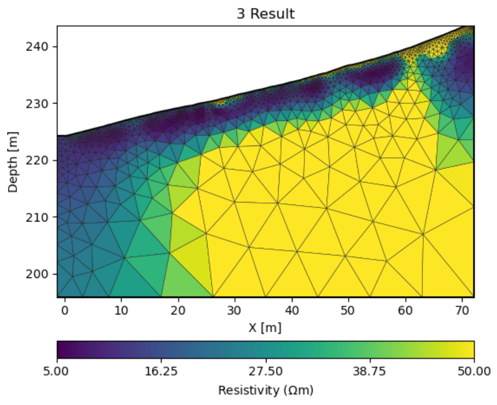
It can be said that clusters and classifiers need to be handled slightly differently. Starting with the similarities: both perform best if trained on mixed or normal profiles and both benefit from hyperparameter tuning and perform worse when samples are weighted with sensitivity. Both prefer to use only resistivity for training and prediction but classifiers are less strict in this preference than clusters. For the latter ones it has weak, negative influence on the results. The biggest differences has been observed in two areas. Clusters clearly benefit from applying minimum sensitivity whereas it decreases the accuracies achieved by classifiers. The normalization of the resistivity has the opposite effect. A beneficial one for classifiers and negative cluster. In terms of accuracy of the interface detection there is very little difference between classifiers. Accuracy score shows more profound differences between performance of different classifiers. Mostly negative outliers are Adaboost and KNN, but it has very limited impact on their ability to detect the interfaces.
Comparing the best experiments in terms of interface detection accuracy leads to an interesting observation. In all of them the results achieved by different classifiers/cluster overlap. Locally the overlap may not be perfect, or there can be some singular outliers, but those are rare exceptions. From this there can be a conclusion drawn that selecting specific classifier/cluster type may be less important than processing and improving the data in an optimal way and optimizing the selection of hyperparameters for the specific classifier.

4-2 Real data

Only a limited subset of experiments was performed on real data. The selection of the experiments configurations was based on the most promising parameter combinations obtained from the synthetic data.
Experiments discussed in this section follow similar naming convention to the ones used for the synthetic data. The difference is that the letter ”R” was added before the experiment’s number to indicate that it uses data from real measurements, e.g.: UR1 - first experiment with real measurements and clusters, AR1 - first experiment with real measurements and classifiers.
For more details on the survey during which the profiles have been measured please see Section 3-2-2. Figure 4-2.1 shows two examples of the inverted real profiles.



(a) An example of an inverted real profile. Measured at 2020-04-30 at 14:30.



(b) An example of an inverted real profile. Measured at 2020-06-07 at 14:30.

Figure 4-2.1

4-2-1 Experiment results

Table 4-3 shows the results of experiments with real data using clustering, while Table 4-4 shows the results of experiments run with classifiers. Values in those tables are sorted by the accuracy score of the prediction. This is different than in Tables 4-1 and 4-2 which were sorted by accuracy of interface detection in the prediction phase. Here the accuracy score is used because the interface detection score is a far less reliable metrics than for the synthetic experiments. As discussed in Section 3-4-2, the interface detection currently only supports horizontal interfaces and hence it is compared to a given value. Both of those are limitations for the real data as the true depth of the interface is unknown and interface is a slope. Still, this metrics can be used to compare experiments with similar configurations that use classifiers or clusters. Figures 4-2.2a and 4-2.2b show an overview of the achieved accuracies with cluster and classifiers respectively.
The results achieved by clusters have to be considered slightly differently as the ones for classifiers. As clusters assign classes randomly, in a binary class problems accuracy scores of 0% and 100% may be equally good in terms of the interface detection.

Table 4-3: Overview of experiments on real measurements with clusters. Sorted by accuracy of interface detection. Ref.: indicates which test from synthetic experiments shares the same configuration; Bal.: balancing; Min. Sen.: minimum sensitivity; Opti.: hyperparameter tuning; Sam. wght.: sample weight; P1: profiles with undetected interfaces; n test: number of tests (total); P2: profiles with undetected interfaces. Line with "-" indicate skipped experiments.

Test name	Ref	Data		Res. type		Train. type	Bal.	Min. sen.	Opti.	Sam. wght.	Accuracy (training)		Accuracy (prediction)		Remarks
		SEN	Y	Norm	log10						score	interface	score	interface	
UR8	U17	No	No	Yes	Yes	Norm	No	Yes	Yes	No	0.64%	20.10%	1.85%	15%	
UR23	U34	No	No	Yes	Yes	Norm	No	Yes	No	No	0.64%	20.10%	1.85%	15.00%	
UR22	U33	No	No	Yes	Yes	Mix	No	Yes	No	No	1.12%	19.52%	3.43%	15.21%	
UR13	U14	No	No	Yes	Yes	Mix	No	No	Yes	No	1.36%	12.75%	4.00%	14.59%	
UR25	-	No	No	Yes	Yes	Mix	No	No	Yes	Yes	1.39%	12.69%	4.07%	14.58%	
UR15	U15	No	No	Yes	Yes	Inv	No	No	Yes	No	0.87%	15.32%	4.32%	14.98%	
UR3	U18	No	No	Yes	Yes	Inv	No	Yes	Yes	No	21.04%	30.70%	13.84%	0%	wrong contrast
UR18	U3	No	No	No	Yes	Inv	No	No	Yes	No	4.19%	14.71%	20.28%	14.91%	
UR17	U2	No	No	No	Yes	Norm	No	No	Yes	No	13.79%	25.43%	22.36%	25.43%	
UR16	U1	No	No	No	Yes	Mix	No	No	Yes	No	10.95%	21.19%	24.37%	14.80%	
UR6	U11	No	No	Yes	Yes	Inv	Yes	Yes	Yes	No	66.26%	15.21%	35.96%	0%	wrong contrast
UR20	A5	Yes	Yes	Yes	Yes	Norm	No	Yes	Yes	No	11.59%	20.75%	46.05%	0%	
UR9	U18	No	No	Yes	Yes	Inv	No	Yes	Yes	No	49.13%	15.31%	49.13%	15.30%	
UR14	U14	No	No	Yes	Yes	Norm	No	No	Yes	No	49.81%	14.25%	50.39%	14.58%	
UR10	U9	No	No	Yes	Yes	Mix	Yes	No	Yes	No	49.67%	17.07%	50.84%	15.04%	
UR01	U16	No	No	Yes	Yes	Mix	No	Yes	Yes	No	51.89%	16.81%	51.32%	0%	wrong contrast
UR4	U9	No	No	Yes	Yes	Mix	Yes	Yes	Yes	No	48.45%	18.96%	51.32%	0%	wrong contrast
UR5	U10	No	No	Yes	Yes	Norm	Yes	Yes	Yes	No	43.94%	19.31%	51.32%	0%	wrong contrast
UR19	A4	Yes	Yes	Yes	Yes	Mix	No	Yes	Yes	No	49.33%	43.56%	51.32%	0.00%	
UR21	A6	Yes	Yes	Yes	Yes	Inv	No	Yes	Yes	No	50.16%	22.05%	51.32%	0%	
UR26	-	No	No	Yes	Yes	Norm	No	No	Yes	Yes	50.41%	14.07%	52.45%	14.58%	
UR02	U17	No	No	Yes	Yes	Norm	No	Yes	Yes	No	63.21%	23.52%	53.95%	0%	wrong contrast
UR12	U11	No	No	Yes	Yes	Inv	Yes	No	Yes	No	49.24%	27.67%	54.10%	15.15%	
UR27	-	No	No	Yes	Yes	Inv	No	No	Yes	Yes	49.02%	15.28%	55.49%	14.96%	
UR24	U35	No	No	Yes	Yes	Inv	No	Yes	No	No	49.13%	15.31%	56.69%	15.30%	
UR7	U16	No	No	Yes	Yes	Mix	No	Yes	Yes	No	46.88%	19.52%	56.85%	15.21%	
UR11	U10	No	No	Yes	Yes	Norm	Yes	No	Yes	No	64.16%	19.52%	70.23%	15.13%	

Table 4-4: Overview of experiments on real measurements with classifiers. Sorted by accuracy of interface detection. Ref.: indicates which test from synthetic experiments shares the same configuration; Bal.: balancing; Min. Sen.: minimum sensitivity; Opti.: hyperparameter tuning; Sam. wght.: sample weight; P1: profiles with undetected interfaces; n test: number of tests (total); P2: profiles with undetected interfaces. Line with "-" indicate skipped experiments.

Test name	Ref	Data		Res. type		Train. type	Bal.	Min. sen.	Opti.	Sam. wght.	Accuracy (training)		Accuracy (prediction)		Remarks
		SEN	Y	Norm	log10						score	interface	score	interface	
AR6	A15	No	No	Yes	Yes	inv	No	No	Yes	No	99.99%	11.41%	99.81%	14.57%	
AR4	A13	No	No	Yes	Yes	mix	No	No	Yes	No	100.00%	12.66%	99.80%	12.66%	
AR9	A21	No	No	Yes	Yes	inv	Yes	No	Yes	No	99.99%	15.97%	99.74%	14.57%	
AR16	A25	No	No	Yes	Yes	mix	No	No	No	No	99.96%	12.70%	99.63%	14.58%	
AR10	A40	No	No	Yes	Yes	mix	Yes	Yes	Yes	No	99.99%	16.70%	99.56%	14.80%	
AR5	A14	No	No	Yes	Yes	Normal	No	No	Yes	No	99.98%	13.90%	99.52%	14.58%	
AR7	A19	No	No	Yes	Yes	mix	Yes	No	Yes	No	99.97%	17.10%	99.52%	17.10%	
AR17	A26	No	No	Yes	Yes	Normal	No	No	No	No	99.96%	13.98%	99.50%	14.58%	
AR18	A27	No	No	Yes	Yes	inv	No	No	No	No	99.90%	11.37%	99.48%	14.57%	
AR11	A41	No	No	Yes	Yes	Normal	Yes	Yes	Yes	No	99.92%	19.15%	99.46%	14.80%	
AR12	A42	No	No	Yes	Yes	inv	Yes	Yes	Yes	No	99.95%	13.76%	99.44%	14.81%	
AR8	A20	No	No	Yes	Yes	Normal	Yes	No	Yes	No	99.91%	19.29%	99.13%	14.58%	
AR19	A16	Yes	Yes	Yes	Yes	mix	No	No	Yes	Yes	99.97%	12.65%	93.32%	14.89%	
AR13	A16	Yes	Yes	Yes	Yes	mix	No	No	Yes	No	99.97%	12.65%	89.28%	14.89%	
AR15	A18	Yes	Yes	Yes	Yes	inv	No	No	Yes	No	99.94%	11.38%	83.80%	9.57%	
AR20	A17	Yes	Yes	Yes	Yes	Normal	No	No	Yes	Yes	99.98%	13.98%	81.29%	11.75%	
AR21	A18	Yes	Yes	Yes	Yes	inv	No	No	Yes	Yes	99.94%	11.38%	78.15%	10.24%	
AR14	A17	Yes	Yes	Yes	Yes	Normal	No	No	Yes	No	99.91%	14.23%	73.44%	9.33%	
AR2	A14	No	No	Yes	Yes	Normal	No	No	Yes	No	84.04%	33.07%	51.66%	0.00%	wrong contrast
AR3	A15	No	No	Yes	Yes	inv	No	No	Yes	No	89.07%	23.85%	51.66%	0.00%	wrong contrast
AR1	A13	No	No	Yes	Yes	mix	No	No	Yes	No					wrong contrast

4-2-2 Hyperparameter tuning

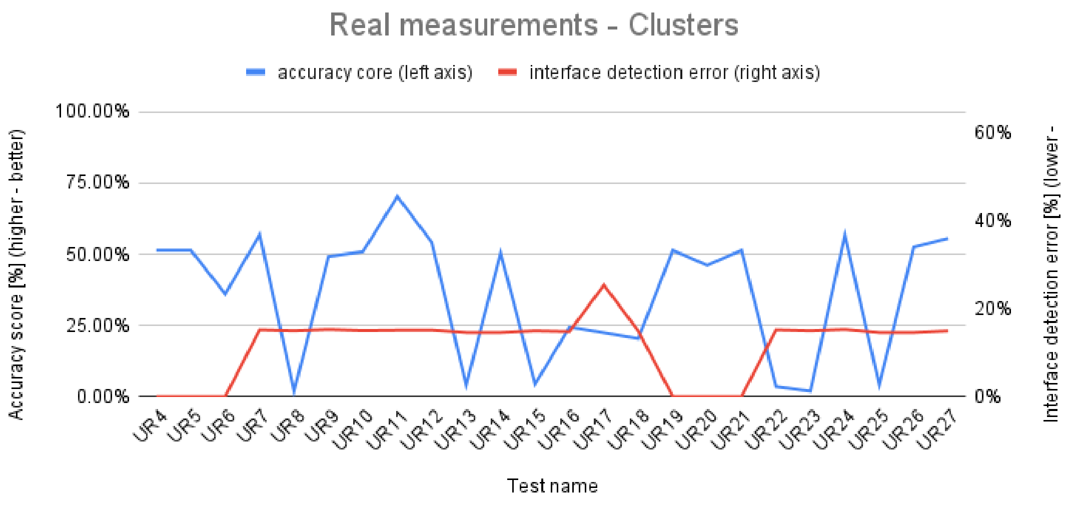
As discussed in one of the previous sections, the hyperparameter tuning is expected to have a positive effect on the accuracy. However, for clusters, the best results were achieved without hyperparameter tuning (UR22 and UR23). Their seemingly low scores are actually related to flipped classes. On the other hand, the seemingly best score of 70.23% (which results in the worst accuracy) is achieved with no hyperparameter tuning as well. There were no other experiments run without hyperparameter tuning, therefore no strong conclusions can be drawn in regard to whether it benefits the clustering or not. The hyperparameter tuning seem not to have as strong and beneficial impact at the achieved accuracies as it was the case for the experiments with clusters run with the synthetic data.
Classifiers have only three experiments run with no hyperparameter tuning as well. Unlike by clusters, they don't score the top score but their performance is above average. Comparing this to the observations made for the clusters, it can be said that in experiments with real measurements hyperparameter tuning has a stronger impact on the classifiers than on clusters.

4-2-3 Sample weighting

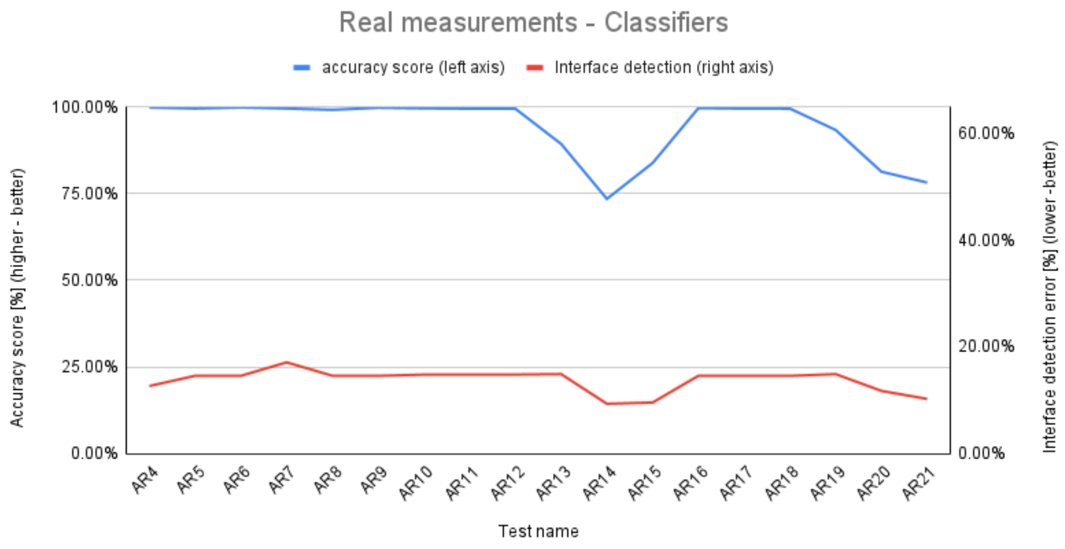
In general the results are fairly comparable with synthetic experiments.
In synthetic experiments with cluster the experiments with sample weighting applied were spread over the whole score table. There are not enough experiments with sample weighting executed with real measurements to replicate this kind of distribution, but two experiments in the top half of the score table and one in bottom part suggest that with more similar experiments run, the tendency would be similar to one in synthetic experiments.
Classifier experiments with real data shows a strong similarity to results of synthetic experiments. Both with real and synthetic data the experiments with sample weighting are clearly in the lower end of the results range. It can be said with some confidence that the experiments with real measurements has confirmed the tendency observed in synthetic ones.

4-2-4 Minimum sensitivity

The effects of applying the minimum sensitivity between experiments on synthetic and real data are the most consistent and similar.
In experiments with clusters and real measurements the runs with minimum sensitivity applied are spread over the whole results table. Generally the distribution of those experiment in the classification results with real measurements is similar to the synthetic experiments. The representation of the experiments executed with minimum sensitivity is slightly higher among the results run with synthetic data compared to the ones with



(a) .



(b) Real measurements - classifiers.

Figure 4-2.2: Real measurements - experiment results.

real measurement. But this difference can be explained with different number of experiments run with minimum sensitivity enabled or disabled with synthetic and real data. In experiments with synthetic data, classifiers have a clear preference for not using minimum sensitivity. Predicting on real data provides the same result. It is less pronounced as there are fewer experiments with it applied, but the best scores are achieved without it. However, it should be stressed that comparing groups AR7-AR8-AR9 (without sample weight applied) and AR10-AR11-AR12 (with sample weight applied) it is hard to reach a strong conclusion. The accuracy scores order those test as follows (from best to worst): AR9-AR10-AR7-AR11-AR12-AR8. No minimum sensitivity wins, but scores the worst result as well. There may be a need for a more detailed investigation of this parameter.

4-2-5 Balancing

The results of experiments run with real data and clusters are consistent. Experiments without balancing perform better. Among the best scores are none that include this step.
Classifiers reach the same conclusion as in the synthetic experiments, where the impact of balancing on the result is limited. In experiments with real measurements the results obtained with balancing applied are not spread as evenly as in synthetic ones but they are very close to it. Comparing pairs (AR6-AR9, AR4-AR7 and other ones) shows consistently that balancing lowers the achieved accuracy. However the differences are rather small. In pair AR6-AR9 the accuracy with balancing is lower by 0.7 percentage point.

4-2-6 Contrast in training data vs contrast in prediction data

At first the clusters and classifiers used to perform prediction on real measurements were trained on the same datasets that were used for synthetic experiments. The training data set included profiles with contrasts 50-5000 Ωm and 500-5000 Ωm. Whereas the profiles on which prediction was performed had much lower contrast of 5-50 Ωm. This approach turned out to lead to low accuracy scores as neither clusters nor classifiers were capable of distinguishing between two classes after this training. Instead they assigned all points in the profile to the same class. Hence, there were no interfaces detected.
The accuracies achieved in the experiments trained and predicted on different resistivity contrasts are low. There were six experiments with mismatching contrasts (between training and prediction phases) run with clusters and three with classifiers. Their results can be found in the result tables with a remark "wrong contrast". In order to mitigate this problem and to verify whether the lack of matching contrast in the training data set was the problem a special training dataset was prepared with the resistivity contrast expected in the real profiles. This solved the problem of all points in the profiles being assigned the same class. Thanks to this achieved results have improved from being very

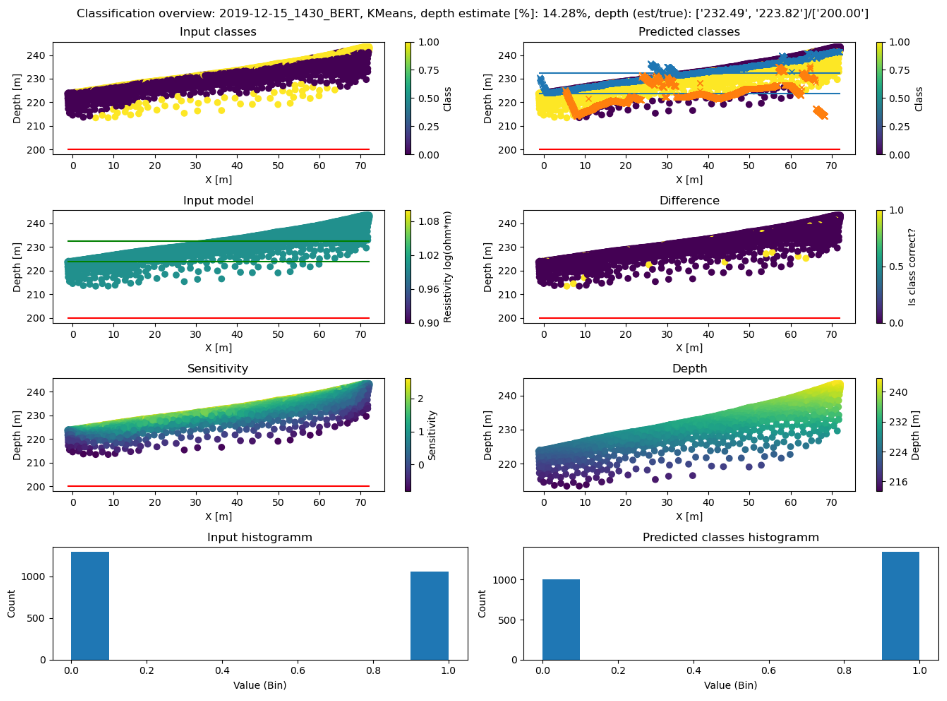
poor to being comparable with accuracies of the synthetic experiments. The results are slightly lower compared to the results of experiments run with synthetic profiles. The difference between best accuracy score of classifiers in experiments with synthetic and real data is 0.16 percentage point. Considering the differences in topography between synthetic and real profiles and the real interface being a slope, this is a promising result. The improvement was so strong that experiments trained on mismatching resistivity contrasts had to be removed from the overview figures in order to maintain their readability.
There are two conclusions that can be drawn from those observations. First, there is a limitation to what clusters and classifiers can achieve and it seems to be related to a difference in the resistivity contrast between training and prediction data. Clusters and classifiers perform poorly when they need to predict on data with contrasts they have not been made familiar with. This is somewhat surprising, since the training was performed on normalized resistivities, and hence a contrast of 5 Ωm - 50 Ωm should be the same as a contrast of 50 Ωm - 500 Ωm. Yet, the results indicate that the correct order of magnitude of the resistivity contrast is needed to achieve accurate results. Hence, the approach has only limited generalizability, and hence prior information is required to achieve good results. This will be further discussed in next Chapter. What is promising is that neither the clusters nor classifiers seemed to be confused by difference in topography and spatial orientation (slope) of the interface between profiles used for training and prediction.

4-2-7 Other observations

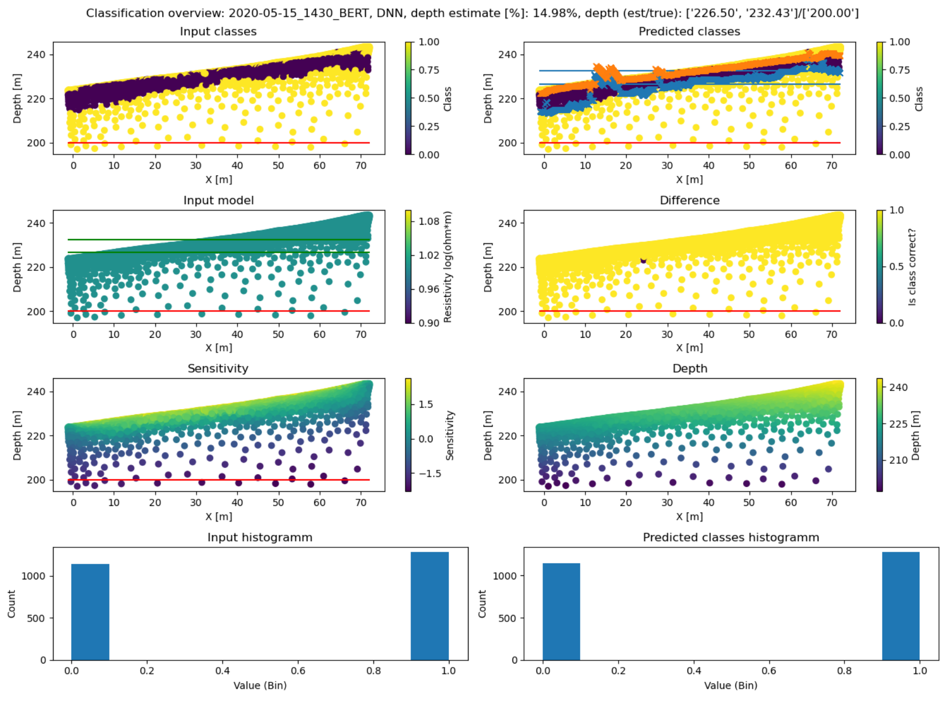
The experiments with real measurements offer some insight to how clusters and classifiers will react to differences between training and prediction data in terms of topography and spatial orientation of the interface. The clusters and classifiers have been trained on synthetic profiles with flat topography and flat horizontal interfaces. Those two are strong assumptions that fail to reflect complexity of real measurements. Apart from some very specific areas (like extremely flat salt plains) those assumptions will not hold. It is very promising that even though the profiles have a noticeable topography and a sloping interface the clusters and classifiers seem not to struggle with it. If this holds true and spatial orientation of the interface and the overall topography play little role (or their impact is limited) it may make training of the clusters and classifiers significantly easier and computationally less expensive. Otherwise training datasets would have to be diverse not only in terms of resistivity contrast or the position of the interface but topography and orientation of the slope will add to it as well. Figure 4-2.3 shows that even with topography and sloping interface the prediction can achieve a good result.

4-2-8 Relation with geology

The hydrogeological structure that can be observed in Figure 4-2.3 is called a perched aquifer. It occurs when water from precipitation penetrates the subsurface and is stopped



(a) Prediction made with clusters experiment UR23 measurement date: 2019-12-15.



(b) Prediction made with clusters experiment AR6 measurement date: 2019-05-15.

Figure 4-2.3: Example of prediction on real data with topography and sloping interface. (a) Clustering, experiment UR23; (b) classification, experiment AR6. Due to a bug plots showing input data (left column, middle row) are incorrect, this has no impact on the prediction results as classes are assigned according to the correct values. Red line shows an arbitrary interface depth and is not related to the expected results in any way, its meaning is only as a placeholder value (required by a script to run).

at some point by an impermeable layer. It corresponds to the lower interfaces in the mentioned figure. The upper interface is the perched water table. The high rainfall will impact both of them. The deeper interface will become deeper as water will seep into the subsurface until it reaches the impermeable layer where it accumulates. This will move the bottom interface to the higher area in the subsurface. The top aquifer will grow thicker with increased precipitation so its bottom boundary will move downwards. Both of those processes are interconnected and show an increase of water saturation in the subsurface which makes the slope less stable; a higher water table will rise pore pressures, which in turn increases the effective stress and lowers the stability of the slopes material. The top of the perched aquifer is more relevant to evaluation of the stability of the slopes. But it is useful that ERT and ML together can track changes of both at the same time.
The Figure 4-2.4 show the change of the interface depth (as a depth below the surface) compared to rainfall and temperature at the profile site. The depth has been estimated manually based on the inverted profiles. It was not done automatically due to a current implementation of the interface detector that lacks support for slopes. However it has been confirmed by comparison between the inverted profiles and result of the prediction that with a interface detector that support slopes, they would be detected with high accuracy.
What can be observed in the figure is that it takes about three months for the rainfall water to change the water table. On the other hand it takes only two months to observe a strong decrease in the water in the ground. What can be concluded from those observations is that an increase in the water level in the lower layer requires significant rainfall and it takes time for water to infiltrate the subsurface deep enough to reach it. On the other hand the response to the lack of rain is quicker.

4-2-9 Summary

There are many similarities between the results achieved by both clusters and classifiers. Applying the sample weight decreases accuracies of both of them. The balancing of classes seem not to benefit either but it performs slightly better with classifiers. Among experiments run with clusters it is clear that none of the tests with balancing of the classes applied achieved a good result. The minimum sensitivity method was more beneficial for clusters than for classifiers.
Generally the results are promising and the achieved accuracies are high.

4-3 Summary and comparison of experiments with synthetic and real profiles

To a big extent results are similar to ones obtained with synthetic data. As there are fewer test executed (the focus was on trying the most promising configurations) the

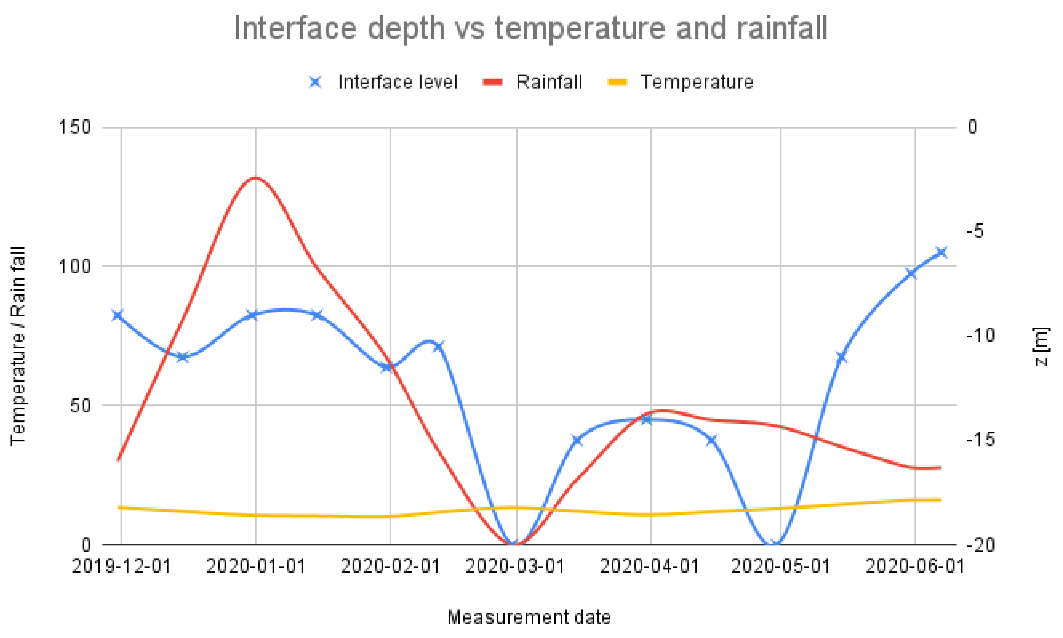


Figure 4-2.4: The depth of the interface below the surface compared to the rainfall and temperature.

observations may not be as conclusive as they were for the synthetic experiments. Still, some of the same tendencies can be confirmed for both synthetic and real data. The similarities are strongest in relation to balancing, and minimum sensitivity. The first one has a fairly limited impact on the accuracy of both clusters and classifiers, whereas the latter one is clearly beneficial to the clusters but not for the classifiers. Neither clusters nor classifiers benefited from weighting of the samples with sensitivity. This is the case for experiments with synthetic data sets as well and this observation is valid for both clusters and classifiers. Regarding the hyperparameter tuning there is a bit less of similarity between experiments run with synthetic and real data. With synthetic data hyperparameter tuning clearly had a positive impact on the results for both clusters and classifiers but this is not the case for real profiles. The clusters managed to achieve high accuracies while predicting on real profiles without hyperparameter tuning. The classifiers still benefited from the tuning.
Generally, it can be said that the there are many similarities between the influence of different preprocessing methods on the results achieved while prediction on synthetic and real profiles. There are some differences as well but they are not strong enough to conclude that prediction on different types of data requires significantly different preprocessing. However the influence of some methods has been tested with limited set of experiments and more detailed investigation might be required to draw strong and general conclusions.

Another important conclusion is that clusters and classifiers can achieve good accuracies while prediction on both synthetic and real profiles.

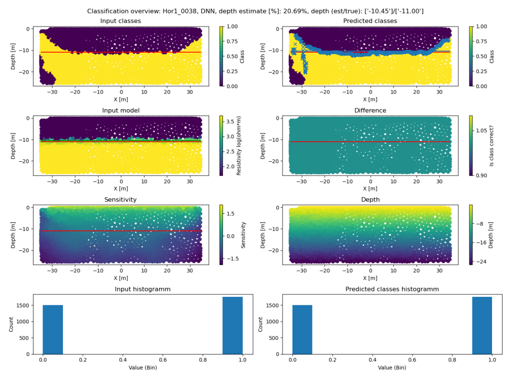
Chapter 5

Summary

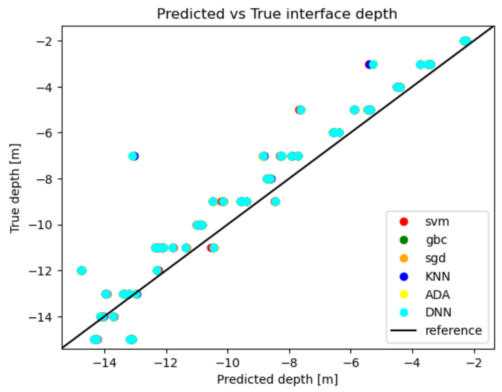
Figure 5-0.1 shows an overview of a selected profiles from the experiments delivering the best and worst results among experiments run with the classifiers and clusters. As it can be seen in Figures 5-0.1a, 5-0.1c, 5-0.1e, and 5-0.1g, there is a significant difference between interfaces detected by the best and worst in each category. This shows what a wide range of results can be achieved with different configurations. The good results show that ML techniques can be applied in a way that makes it possible to automatically detect interfaces in the subsurface at an accuracy comparable to conventional techniques, such as drilling, or manual picking. Comparison of results obtained by the steepest gradient method and borehole drilling presented in paper: [Chambers et al., 2012] shows that for many interfaces similar accuracy (difference between true and predicted depth lesser than 0.5 meter) was achieved. This is a more accurate result, however in this study a more general problem is investigated.

5-1 Limitations

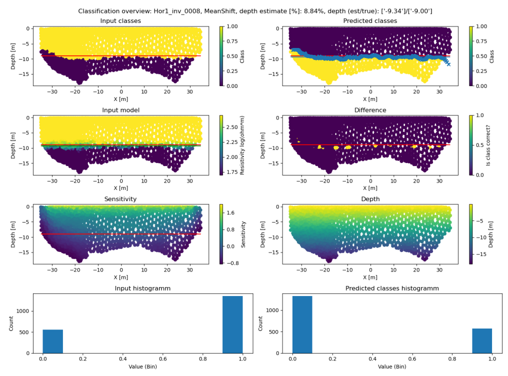
There are two major factors that limit the prediction accuracy found in the experiments. First one is the quality of the inversion. Comparing the accuracy of profiles between the experiments some trends can be observed. For example, profile Hor1_0030 has performed poorly independent of the configuration of the experiments. To some extent it seems to correlate with either a too shallow or too deep position of the interface. Both situations are challenging to the predict. If the interface is too shallow, there are very few points representing the upper layer which may lower a chance of accurate prediction for some classifiers. An unfavorable size and positions of the cells it can lead to large relative errors. When the interface is too deep in the subsurface, the sensitivity can play a significant role, where deeper parts of the subsurface are known to be less well



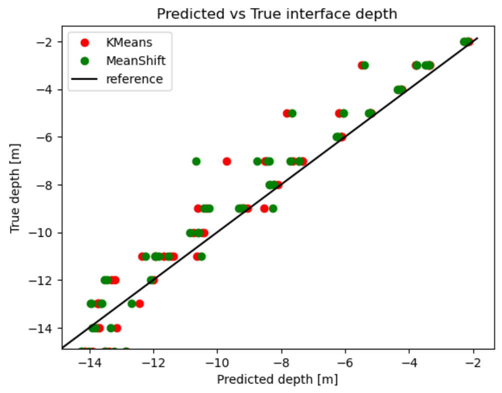
(a) Classifiers best result - A14 - overview of the prediction result.



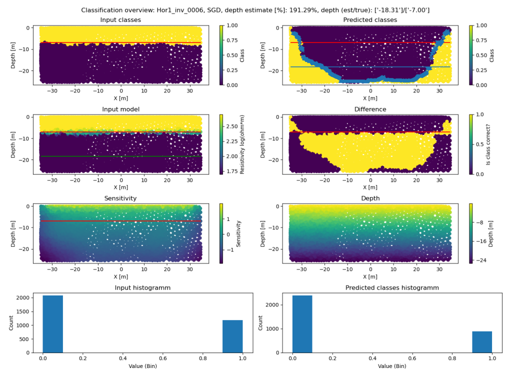
(b) Classifiers best result - A14 - true vs predicted interface depth.



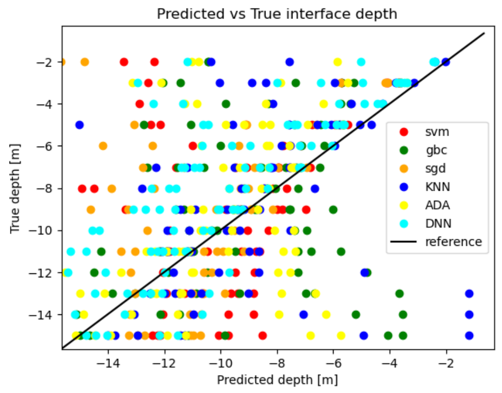
(c) Clusters best result - U17 - overview of the prediction result.



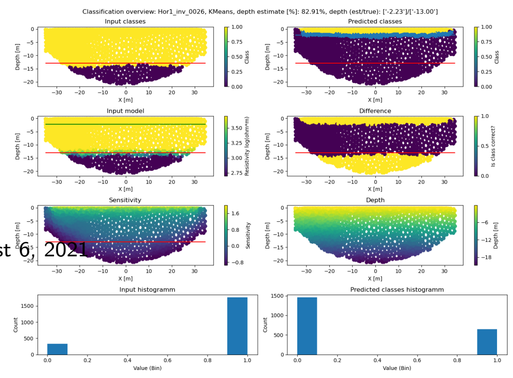
(d) Clusters best result - U17 - true vs predicted interface depth.



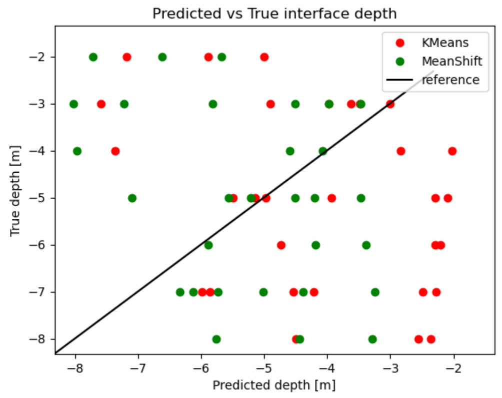
(e) Classifiers worst result - A11 - overview of the prediction result.



(f) Classifiers worst result - A11 - true vs predicted interface depth.



(g) Clusters worst result - U44 - overview of the prediction result.



(h) Clusters worst result - U44 - true vs predicted interface depth.

Figure 5-0.1: Best results: an example for (a) classifiers - A11; (b) clusters - U55.

resolved when using surface electrodes arrays. If the lower layer is placed in the part of the profile with low resistivity, it may be challenging to extract sufficient information to resolve it reliably. It can lead to a situation in which some points used for training will be assigned incorrect classes due to errors in the result of the inversions for specific points. Training clusters and classifiers on a dataset counting many of those points may hinder their capabilities of generalizing.
As many of the experiments achieved fairly similar accuracies, it may suggest that the clusters and classifiers have managed to reach their limits for the used datasets. There are two potential conclusions that could be drawn from it. It is possible that insufficient pre-processing was applied to the data that were used in the clustering and classification. Scikit-learn contains many pre-processing tools that were not used in this project. Perhaps additional techniques available in the toolbox and/or other external preprocessing steps could be used to improve the results. The other direction that can be taken in order to improve the results is related to the profiles. In this experiment all profiles were inverted with the same parameters. This "one-size-fits-all" approach was taken in order to limit the amount of variables tested in the experiments, but it is possible that with better parameter choices the quality of the inversion would improve. The training and prediction accuracies would most probably benefit from it. Those two ways to improve the results could be applied and validated simultaneously.
The second limiting factor seems to be the knowledge of the contrast in the subsurface. Experiments run with real measurements have shown clearly that training classifiers with the same contrast as present in the data will lead to a strong improvement of the achieved accuracies in the class and interface prediction. Neither clusters nor classifiers were capable of prediction accurately when they were given contrast of $5 - 50\Omega * m$ when they were only trained on contrasts of $50 - 500\Omega * m$ and $500 - 5000\Omega * m$. They have consistently assigned all points to the same class, hence the identification of the interface was not possible. After the classifiers and clusters have been trained on the contrast matching the one in the profiles they were predicting on the accuracy increased immensely. This made detection of the interfaces possible and in many cases accurate. Normalizing and applying the decimal logarithm to the data did not mitigate this issue sufficiently. Potentially this could be solved with bigger data sets, containing profiles with more diverse contrasts in resistivity. But it will require additional investigation. An alternative would be to prepare a synthetic dataset for training based on the available knowledge about the subsurface structure in the surveyed area. It can be based on other geological survey conducted in the region or on results from borehole logging.

5-2 Applicability

Even though clustering and classification have some significant limitations, this study showed that there is a lot of potential in applying ML techniques to ERT monitoring problems. It is possible to make reliable predictions with both classifiers and clusters. However the experiments conclude that clusters and classifiers have to be trained for each monitored area separately and that some prior knowledge about the geology and the subsurface situation is required.

5-3 Outlook

The results and conclusions from this study show that ML should be applicable to ERT data to automatically extract valuable information over space and time. However there is still a lot of potential for improvements. Apart from this there remain many things that need more detailed investigation.
First and the most obvious direction for further investigation and research is prediction on profiles with more complex geology. Classifiers are partially capable of making use of spatial information about the samples (in form of the depth Y and sensitivity SEN). Will the importance of those features increase with flat but not horizontal layers? A hint on this can be found in experiments with real measurements that include slopes as interfaces. The experiments with real measurements were run with profiles that showed a slope oriented towards the left end of the profile. Otherwise the surface and interface were flat (following the surface slope). This is still a very simple topography but it shows that clusters and classifiers can work correctly and deliver good results with profiles that violate the assumptions with which the training profiles were created. What requires further investigation is how strongly the topography and subsurface structure can differ between training and prediction profiles. Another interesting question is whether clusters and classifiers can include the Y and SEN features in their prediction and if it can improve the accuracy of prediction performed on profiles with complex structure and topography if trained on profiles with matching complexity.

Another interesting area that could be investigated is the performance of ML techniques with profiles with more interfaces. There are many questions that could be asked and answered in regard to them. How should the classifiers be trained? If there are different contrasts within one profile should they be presented to classifiers in form of examples with single interface or can they be combined? Throughout this project some experiments were run with two interfaces in the subsurface but due to the limited time they have been abandoned in favour of more detailed investigation of simpler problems with only one interface. Introducing more interfaces in the subsurface not only increases the complexity of the geology, but also increases the number of parameters and variables that need to be investigated to fully understand the capabilities of ML to predict on those kind of profiles.

Another area of potential improvements is related to the potential of generalization of cluster and classifiers. In the experiments performed during this project the biggest limitation was related to the contrast of resistivities. Both clusters and classifiers seem to struggle making predictions on profiles with resistivity contrast that they were not trained to recognize. The capability of generalizing the clusters and classifiers to predict on wider range of contrasts has to be investigated. Considering how some of them work (for example KNN) exposure to more diverse contrast in the training phase should mitigate this problem. On the other hand, presenting classifiers with too much diversity may create confusion and decrease their performance. The exact impact of the size and diversity of the training dataset has to be investigated further. Alternatively the training can be executed specifically for each monitored area focusing on the resistivity contrasts characteristic to that area. The needed prior knowledge can be extracted from borehole or other methods that will likely be applied in order to validate the results.

Generally, the clusters and classifiers have proven to be capable of detecting the interfaces between layers of different resistivities. The accuracy varies and there is a lot of space for improvement but the results are promising. There is more research and testing required.

Bibliography

Abolmasov, B., Ristic, A., & Govedarica, M. (2013). Applying GPR and 2D ERT for Shallow Landslides Characterization: A Case Study. *Landslide Science and Practice: Early Warning, Instrumentation and Monitoring*, *2*, 495–502. https://doi.org/10.1007/978-3-642-31445-2-65

Archie, G. (1942). The electrical resistivity log as an aid in determining some reservoir characteristics. *Transactions of the American Institute of Mining, Metallurgical, and Petroleum Engineers*, *146*, 54–62.

Bai, W., Kong, L., & Guo, A. (2013). Effects of physical properties on electrical conductivity of compacted lateritic soil. *Journal of Rock Mechanics and Geotechnical Engineering*, *5*(5), 406–411. https://doi.org/https://doi.org/10.1016/j.jrmge.2013.07.003

Bednarczyk, Z. (2013). *An On-Line Landslide Monitoring System in Flysch Carpathians* (Vol. All Days) [_eprint: https://onepetro.org/ISRMEUROCK/proceedings-pdf/EUROCK13/All-EUROCK13/ISRM-EUROCK-2013-098/1556243/isrm-eurock-2013-098.pdf].

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer Science+Business Media, LLC.

Borgatti, L., & Soldati, M. (2010). Landslides and climatic change [ISBN: 9780511807527]. *Geomorphological Hazards and Disaster Prevention*, 87–96. https://doi.org/10.1017/CBO9780511807527.008

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 144–152.

Butler, D. (2005). *Near-sruface geophysics*. Society of Exploration Geophysics.

Chambers, J., Ogilvy, R., Kuras, O., Cripps, J., & Meldrum, P. (2002). 3D electrical imaging of known targets at a controlled environmental test site. *Environmental Geology*, *41*. https://doi.org/10.1007/s00254-001-0452-4

Chambers, J., Wilkinson, P., Wardrop, D., Hameed, A., Hill, I., Jeffrey, C., Loke, M., Meldrum, P., Kuras, O., Cave, M., & Gunn, D. (2012). Bedrock detection beneath river terrace deposits using three-dimensional electrical resistivity tomography. *Geomorphology*, *s 177–178*, 17–25. https://doi.org/10.1016/j.geomorph.2012.03.034

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(8), 790–799. https://doi.org/10.1109/34.400568

Comaniciu, D., & Peter, M. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 281–288.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 273–297.

Datsios, Z., Mikropoulos, P., & Karakousis, I. (2017). Laboratory characterization and modeling of DC electrical resistivity of sandy soil with variable water resistivity and content. *IEEE Transactions on Dielectrics and Electrical Insulation*, *24*, 3063–3072. https://doi.org/10.1109/TDEI.2017.006583

Delforge, D., Watlet, A., Kaufmann, O., Camp, M. V., & Vanclooster, M. (2021). Time-series clustering approaches for subsurface zonation and hydrofacies detection using a real time-lapse electrical resistivity dataset. *Journal of Applied Geophysics*, *184*, 104203. https://doi.org/https://doi.org/10.1016/j.jappgeo.2020.104203

Dick, P. K. (1956). *The Minority report*.

Drucker, H. (1997). Improving Regressors Using Boosting Techniques. *Proceedings of the 14th International Conference on Machine Learning*.

Dyhoum, T., Lesnic, D., & Aykroyd, R. (2014). Solving the complete-electrode direct model of ERT using the boundary element method and the method of fundamental solutions. *Electronic Journal of Boundary Elements*, *12*, 26–71. https://doi.org/10.14713/ejbe.v12i3.1850

Ellis, D., & Singer, J. (2007). Well Logging for Earth Scientists.

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/https://doi.org/10.1006/jcss.1997.1504

Froude, M. J., & Petley, D. N. (2018). Global fatal landslide occurrence from 2004 to 2016. *Natural Hazards and Earth System Sciences*, *18*(8), 2161–2181. https://doi.org/10.5194/nhess-18-2161-2018

Gariano, S. L., & Guzzetti, F. (2016). Landslides in a changing climate. *Earth-Science Reviews*, *162*. https://doi.org/10.1016/j.earscirev.2016.08.011

Glover, P. (2016). Archie's Law – A reappraisal. *Solid Earth Discussions*, 1–18. https://doi.org/10.5194/se-2016-47

Goldman, e. a. (2010). Combined ERT-TDEM measurements for delineating saline water intrusions from an estuarine river into the adjacent aquifer. *SWIM21 - 21st Salt Water Intrusion Meeting*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Günther, T., Rücker, C., & Spitzer, K. (2006a). Three-dimensional modelling and inversion of DC resistivity data incorporating topography - I. Modeling. *Geophysical Journal International*, *166*, 506–517.

Günther, T., Rücker, C., & Spitzer, K. (2006b). Three-dimensional modelling and inversion of DC resistivity data incorporating topography - II. Inversion. *Geophysical Journal International*, *166*, 506–517.

Hartigan, J. A. (1975). *Clustering algorithms [by] John A. Hartigan* [Type: Book]. Wiley New York.

Highland, L., & Bobrowsky, P. (2008). *The Landslide Handbook – A Guide to Understanding Landslides* [Publication Title: US Geological Survey Circular].

Huang, J., Zheng, J., Gao, S., Liu, W., & Lin, J. (2020). Grid text classification method based on DNN neural network. *MATEC Web of Conferences*, *309*, 03016. https://doi.org/10.1051/matecconf/202030903016

Hubbard, Y. R. S. S. (2005). *Hydrogeophysics*. Springer.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment [Publisher: IEEE COMPUTER SOC]. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Igel, H. (2016). *Computational Seismology: A Practical Introduction*. https://doi.org/10.1093/acprof:oso/9780198717409.001.0001

Jiang, C., Igel, J., Dlugosch, R., Müller-Petke, M., Günther, T., Helms, J., Lang, J., & Winsemann, J. (2020). Magnetic resonance tomography constrained by ground-penetrating radar for improved hydrogeophysical characterization. *Geophysics*, *85*(6), JM13–JM26. https://doi.org/10.1190/geo2020-0052.1

Jivani, A., Shah, K., Koul, S., & Naik, V. (2016). The Adept K-Nearest Neighbour Algorithm - An optimization to the Conventional K-Nearest Neighbour Algorithm. *Transactions on Machine Learning and Artificial Intelligence*, *4*. https://doi.org/10.14738/tmlai.41.1876

Joblib-Development-Team. (2020). Joblib: Running Python functions as pipeline jobs. https://joblib.readthedocs.io/

Jongmans, D., & Garambois, S. (2007). Geophysical investigation of landslides: A review. *Bulletin de la Societe Geologique de France*, *178*. https://doi.org/10.2113/gssgfbull.178.2.101

Lech, M., Skutnik, Z., Bajda, M., & Markowska-Lech, K. (2020). Applications of Electrical Resistivity Surveys in Solving Selected Geotechnical and Environmental Problems. *Applied Sciences*, *10*(7). https://doi.org/10.3390/app10072263

Maurer, S., Hansruedi; Friedel. (2006). Outer-space sensitivities in geoelectrical tomography. *Geophysics*, *71*(3), 93–96. https://doi.org/10.1190/1.2194891G93

Milenkovic, S., Jelisavac, B., Vujanic, V., & Jotic, M. (2013). Monitoring of the "Razanj" Landslide in Serbia. *2*, 25–31. https://doi.org/10.1007/978-3-642-31445-2-3

Müller, A. C., & Guido, S. (2017). *Machine Learning with Python: A Guide For Data Scientists*. O'Reilly Media Inc.

Occhiena, C., Pirulli, M., & Scavia, C. (2013). Application of a Multiplet-Location Coupled Technique to Microseismic Data for Identification of Rock Slope Active Surfaces [ISBN: 978-3-642-31444-5]. *Landslide Science and Practice: Early Warning, Instrumentation and Monitoring*, *2*, 101–107. https://doi.org/10.1007/978-3-642-31445-2-13

Pandey, L. M., Shukla, S., & Habibi, D. (2015). Electrical resistivity of sandy soil. *Géotechnique Letters*, *5*, 178–185. https://doi.org/10.1680/jgele.15.00066

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perrone, A., Lapenna, V., & Piscitelli, S. (2014). Electrical resistivity tomography technique for landslide investigation: A review. *Earth-Science Reviews*, *135*, 65–82. https://doi.org/https://doi.org/10.1016/j.earscirev.2014.04.002

Philip Kearey, I. H., Michael Brooks. (2002). *An Introduction to Geopysical Exploration* (3rd). Blackwell Science.

Powers, D. (2008). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. *Mach. Learn. Technol.*, *2*.

Ramirez, A., Daily, W., & Binley, A. (2000). Electrical resistance tomography. *The Leading Edge*, *23*. https://doi.org/10.2172/15010154

Reback, J. (2020). Pandas-dev/pandas: Pandas [Publisher: Zenodo Version Number: latest]. *Pandas*. https://doi.org/10.5281/zenodo.3509134

Reci, H., Muceku, Y., & Jata, I. (2013). The Use of ERT for Investigation of Berzhita Landslide, Tirana Area, Albania. *Landslides*, *2*, 117–123. https://doi.org/10.1007/978-3-642-31445-2-15

Reynolds, J. M. (2011). *An Introduction to Applied and Environmental Geophysics* (2nd). John Wiley & Sons, Ltd.

Rücker, C., Günther, T., & Wagner, F. M. (2017). pyGIMLi: An open-source library for modelling and inversion in geophysics. *Computers and Geosciences*, *109*, 106–123. https://doi.org/10.1016/j.cageo.2017.07.011

Salih, A., & Dhannoon, B. N. (2021). Weighted k-Nearest Neighbour for Image Spam Classification. *Iraqi Journal of Science*, *62*, 1036–1045. https://doi.org/10.24996/ijs.2021.62.3.32

Sarwar, A. (2017). K-Nearest Neighbours based diagnosis of hyperglycemia. *International Journal of Trend in Scientific Research and Development*, *Volume-2*, 611–614. https://doi.org/10.31142/ijtsrd7046

Thuro, K., Singer, J., & Festl, J. (2013). A Geosensor Network Based Monitoring and Early Warning System for Landslides [ISBN: 978-3-642-31444-5]. *Landslide Science and Practice: Early Warning, Instrumentation and Monitoring*, *2*, 79–86. https://doi.org/10.1007/978-3-642-31445-2-10

Tikhonov, A. N., & Arsenin, V. Y. (1977). *Solutions of Ill-posed problems*. W.H. Winston.

Tsuruoka, Y., Tsujii, J., & Ananiadou, S. (2009). Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th Inter-*

national Joint Conference on Natural Language Processing of the AFNLP, 477–485. https://aclanthology.org/P09-1054

Uhlemann, S., Chambers, J., Meldrum, P., McClure, P., & Dafflon, B. (2021). Geophysical monitoring of landslides – a step closer towards predictive understanding? *Understanding and Reducing Landslide Disaster Risk* (pp. 85–91). https://doi.org/10.1007/978-3-030-60311-3_8

Uhlemann, S., Chambers, J., Wilkinson, P., Maurer, H., Merritt, A., Meldrum, P., Kuras, O., Gunn, D., Smith, A., & Dijkstra, T. (2017). Four-dimensional imaging of moisture dynamics during landslide reactivation [_eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2016JF003983]. *Journal of Geophysical Research: Earth Surface*, *122*(1), 398–418. https://doi.org/https://doi.org/10.1002/2016JF003983

Van Rossum, G., & Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Vapnik, V., & Kotz, S. (2006). *Estimation of Dependences Based on Empirical Data* (Vol. 41) [Publication Title: The Statistician]. https://doi.org/10.2307/2988246

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., . . . SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

W. O. Winsauer, H. M. S. (1952). Resistivity of Brine-Saturated Sands in Relation to Pore Geometry [Publisher: American Association of Petroleum Geologists AAPG/Datapages]. *AAPG Bulletin*, *36*. https://doi.org/10.1306/3d9343f4-16b1-11d7-8645000102c1865d

Wahlstrom M., Guha-Sapir D. (2015). The Human Cost of Weather-Related Disasters 1995-2015, 30.

Ward, W., Wilkinson, P., Chambers, J., Oxby, L., & Bai, L. (2014). Distribution-based fuzzy clustering of electrical resistivity tomography images for interface detection [_eprint: https://academic.oup.com/gji/article-pdf/197/1/310/2028734/ggu006.pdf]. *Geophysical Journal International*, *197*(1), 310–321. https://doi.org/10.1093/gji/ggu006

Whiteley, J. S., Chambers, J. E., Uhlemann, S., Wilkinson, P. B., & Kendall, J. M. (2019). Geophysical Monitoring of Moisture-Induced Landslides: A Review [_eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018RG000603]. *Reviews of Geophysics*, *57*(1), 106–145. https://doi.org/https://doi.org/10.1029/2018RG000603

Zhu, J., Rosset, S., Zou, H., & Hastie, T. (2006). Multi-class AdaBoost. *Statistics and its interface*, *2*. https://doi.org/10.4310/SII.2009.v2.n3.a8

Zhu, T., Lin, Y., & Liu, Y. (2017). Synthetic minority oversampling technique for multiclass imbalance problems. *Pattern Recognition*, *72*, 327–340. https://doi.org/10.1016/j.patcog.2017.07.024

Zienkiewicz, O., Taylor, R., & Zhu, J. (2005). *The Finite Element Method: Its Basis and Fundamentals* (Vol. 1). Elsevier.