

**Master's thesis**

**NTNU**  
Norwegian University of Science and Technology  
Faculty of Engineering  
Department of Civil and Environmental Engineering

Martin Berrum and Håvard Skaar

# Identification of quick clay using cone penetration tests and machine learning

Master's thesis in Civil and Environmental Engineering

Supervisor: Priscilla Paniagua

Co-supervisor: Gudmund Eiksund

June 2021



Norwegian University of  
Science and Technology



Martin Berrum and Håvard Skaar

# Identification of quick clay using cone penetration tests and machine learning

Trondheim, June 2021

## MASTER THESIS: TBA4900

Main supervisor: Priscilla Paniagua

Co-supervisor: Gudmund Eiksund

Department of Civil and Environmental Engineering

Norwegian University of Science and Technology (NTNU)



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

## Preface

This Master thesis in geotechnics is part of the MSc in Civil and Environmental Engineering at the Norwegian University of Science and Technology (NTNU) in Trondheim. It is a mandatory part of the course TBA4900 carried out in the spring of 2021 by the Department of Civil and Environmental Engineering.

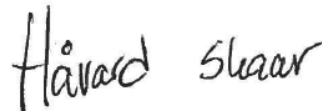
This study aims to investigate the applicability of machine learning techniques to identify quick clay layers from CPTu.

The Master thesis is a continuation of the Project thesis "Applications of correlations and machine learning on CPTu" by Martin Berrum and Håvard Skaar from course TBA4510 during the autumn semester 2020. Ana Priscilla Paniagua Lopez at NGI supplied the idea for both the Project thesis and Master thesis, with inputs from the students.

Trondheim, 2021-06-11



Martin Berrum



Håvard Skaar

## **Acknowledgment**

We would like to specially thank our supervisor Priscilla Paniagua. Firstly for supplying us with the idea for this Master thesis, and secondly for all the feedback and guidance throughout this work.

Thanks to Ivan Depina at SINTEF and Zhongqiang Liu at NGI for their time and inputs to this thesis.

Thanks to Sigurdur Mår Valsson for sharing information regarding machine learning and CPTu.

Thanks to R&D Program Norwegian Geo-Test Sites – NGTS supported by The Research Council of Norway Infrastructure program for supplying CPTu data.

## Abstract

Quick clay occurrence is a vital part of the geotechnical engineering field in Norway. Its presence affects the way forward for completion of ground surveys, design engineering, control and quality routines regarding the project work.

Identification of quick clay relies on a combination of field testing, sampling and laboratory testing, together with the interpretation from the geotechnical engineer. Sampling and subsequent laboratory testing is the only unmistakable classification method of quick clay, however due to its expense it is often limited to certain projects and to relatively small depths. Field methods today, such as the cone penetration test, can in many cases give good indication of quick clay, although sampling and laboratory tests are required to verify the occurrences. The in-situ field tests provides a quick way to obtain continuous information about the soil profile.

Techniques of classifying soils with CPTu data are traditionally done through interpretations from geotechnical engineers and classification charts. However, these charts have shown to have difficulties in detection of Norwegian quick clays. Newly proposed methods using machine learning on CPTu have shown promising results in detecting quick clay.

This thesis will work further with testing out machine learning algorithms to classify quick clay by CPTu. Seven algorithms and three datasets will be analyzed for training and testing purposes. NGTS Tiller-Flotten dataset consists of 32 CPTus and functions as a benchmark to test the different algorithms on the same dataset. Both performance and training speed will be measured to compare which algorithm achieve the best results. Two new datasets are implemented in order to analyze how the machine learning algorithms performs when trained and tested on different datasets. The data are divided into two classes, respectively quick clay and other material. Visualization of the models are done in two and three dimensions to understand how the algorithms separate the classes.

The results show that neural networks generally works well, and that adding convolutional layers to the network can make for more generalizable models. Algorithms using a decision tree architecture struggle with classifying quick clay when the tested CPTus are not part of the training dataset, while the support vector machines tend to not have this problem.

In the case where the training and testing dataset are the same, all algorithms show accuracy scores of at least 97 %. In other cases the performances of the models have higher variances, where random forest and extreme gradient boost suffers the most.

## Sammendrag

Forekomst av kvikkleire er en viktig del av geoteknisk ingeniørarbeid i Norge. Dens tilstedeværelse påvirker måten grunnundersøkelser, dimensjonering, kontroll og kvalitetssikring blir utført på i et prosjekt.

Identifisering av kvikkleire avhenger av en kombinasjon av feltundersøkelser, prøvetaking og laboratorieundersøkelser, sammen med tolkning av geotekniske ingeniører. Prøvetaking og påfølgende laboratorieundersøkelse er den eneste metoden som gir sikker påvisning av kvikkleire, men på grunn av dens kostnader er det ofte begrenset til bestemte prosjekter og til relativt få dybder. Feltundersøkelsesmetoder som trykksondering kan i mange tilfeller gi en god indikasjon på kvikkleire, men laboratorieundersøkelser er nødvendig for å verifisere forekomsten. In-situ feltundersøkelser gir en rask måte å anskaffe kontinuerlig informasjon om løsmasseprofilen.

Metoder for å klassifisere løsmasser med data fra trykksondering er tradisjonelt sett utført ved tolkning av geotekniske ingeniører og klassifiseringsdiagrammer. Disse klassifiseringsdiagrammene har derimot vanskeligheter til å detektere norske kvikkleirer. Nylig foreslåtte metoder som bruker maskinlæring og trykksondering har vist lovende resultater innen detektering av kvikkleire.

Denne avhandlingen vil arbeide videre med å teste ut maskinlæringsalgoritmer til å klassifisere kvikkleire ved trykksondering. Sju algoritmer og tre datasett blir analysert og brukt for trening og testing. NGTS Tiller-Flotten datasettet består av 32 trykksonderinger og virker som standard for måling av ytelse for å teste de ulike algoritmene på samme datasettet. Både prestasjon og tidsbruk vil bli målt for å sammenligne hvilke algoritmer som oppnår best resultat. To nye datasett blir implementert for å analysere hvordan maskinlæringsalgoritmene presterer når de blir trent og testet på ulike datasett. Dataen er inndelt i to klasser, følgende kvikkleire og annet materiale. Visualisering av modellene er utført i to og tre dimensjoner for å forstå hvordan de ulike algoritmene separerer disse to klassene.

Resultatene viser at nevralt nettverk generelt virker bra. Når man legger til et konvolusjonsfilter til nettverket kan modellene bli enda mer generaliserbare. Algoritmer med beslutningstre som arkitekturtype har problemer med å klassifisere kvikkleire når trykksonderingene som blir testet



ikke er del av datasettet algoritmen ble trent på. Det virker ikke som "support vector machines" har dette problemet.

Når algoritmene blir trent og testet på samme datasett viser resultatene en nøyaktighet på minimum 97 %. I andre tilfeller varierer resultatene til modellene mer, der spesielt "random forest" og "extreme gradient boost" får størst unøyaktigheter.

# Contents

Preface . . . . .	i
Acknowledgment . . . . .	ii
Abstract . . . . .	iii
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Objectives . . . . .	3
1.3 Limitations . . . . .	4
1.4 Approach . . . . .	4
1.5 Structure of the Report . . . . .	5
<b>2 Machine learning</b>	<b>7</b>
2.1 Machine learning concept . . . . .	7
2.2 Neural network algorithms . . . . .	10
2.2.1 DNN . . . . .	14
2.2.2 CNN . . . . .	15
2.2.3 ELM . . . . .	16

2.3	Decision tree algorithms . . . . .	17
2.3.1	RF . . . . .	18
2.3.2	XGB . . . . .	19
2.4	Nearest neighbor algorithms . . . . .	20
2.4.1	KNN . . . . .	21
2.4.2	SVM . . . . .	21
<b>3</b>	<b>Presentation of datasets</b>	<b>24</b>
3.1	Dataset I: NGTS Tiller-Flotten . . . . .	24
3.2	Dataset II: Multiple sites located in Norway. . . . .	29
3.3	Dataset III: Saksvik . . . . .	33
3.4	Data preparation . . . . .	36
3.4.1	Parameter choice and normalization . . . . .	36
3.5	The impact of faulty CPTu reading and errors in dataset . . . . .	39
3.6	Implementation in Python . . . . .	42
<b>4</b>	<b>Results</b>	<b>43</b>
4.1	ML models trained and tested on dataset I . . . . .	44
4.2	ML models trained on dataset II and tested on dataset I . . . . .	65
4.3	Validation of trained models on a third dataset . . . . .	71
<b>5</b>	<b>Summary and Recommendations for Further Work</b>	<b>78</b>
5.1	Summary and Conclusions . . . . .	78

5.2 Discussion . . . . . 80

5.3 Recommendations for Further Work . . . . . 81

**Bibliography** **82**

**A CPTu raw data graphs** **88**

**B Saksvik total soundings** **134**

**C Index testing profiles** **140**

**D Saksvik piezometer** **148**

**E 2D plots from NGTS dataset not included in the text** **150**

**F 3D plots from dataset II not included in the text** **158**

# Chapter 1

## Introduction

### 1.1 Background

A critical task in geotechnical engineering is to determine the ground conditions for a project. Several factors have an impact on the degree of difficulty of this task, spanning from economical reasons to whether there is access for the boring rig at the given site. Determining the ground conditions incorrectly may lead to catastrophic events, especially if highly sensitive material such as quick clay is present, which often is the case below the marine limit in Norway.

Classification of soils are commonly done by interpreting different geotechnical field tests, soil sampling and laboratory testing, all of which need the judgment of a geotechnical engineer. In-situ field tests does in many cases give good indications on brittle material behavior, although soil sampling and subsequent laboratory testing is necessary to verify this. Determination of quick clay is mainly restricted to the laboratory, where the falling cone test on a remolded sample is the standard procedure. Quick clay is defined as clay with remolded shear strength  $c_{u,r} < 0.5kPa$  (Sandven et al., 2019). Significant work has been made in creating ways to characterize soil by CPTu, typically by the use of classification charts such as Robertson charts (Robertson, 2016). Studies however show that the reliability of classification charts are often underwhelming when predicting highly sensitive soils as the characteristic properties of the soil may vary substantially from one site to another.

Recent research have looked at the use of machine learning techniques to more accurately characterize and identify quick clay soil layering based on CPTu soundings (([Valsson, 2019](#)), ([Godoy et al., 2020](#)), ([Berrum and Skaar, 2020](#)), ([Erharter et al., 2021](#))). These studies have mainly focused on a limited amount of machine learning methods. In this work, a further research into several machine learning methods such as deep neural networks, decision tree models and nearest neighbor algorithms will be performed.

### **Problem Formulation**

This thesis will be a research on the applicability of machine learning techniques to detect quick clay from CPTu. The method for detecting quick clay is heavily reliant on laboratory testing as other methods only gives indications of its presence. The main purpose of this thesis is to determine which algorithm is best suited to predict quick clay soil layering. Provided that a machine learning algorithm can accurately classify quick clay layering from CPTu, it can give an early indication to the geotechnical engineer and lead to better design decisions, resulting in safer projects and smaller project costs.

### **Literature Survey**

The use of machine learning for detecting quick clay from CPTu is a mostly new topic which isn't widely researched. However, some articles have been published in this area in recent years.

The most prominent researcher on the topic is S. M. Valsson, who at the moment works on a PhD regarding soil classification by the use of machine learning. In 2018 he released an article where CPTus from 50 projects around Norway were gathered in a database to be used for training ML models, ([Valsson et al., 2018](#)). In the study Valsson focuses on selecting the best pair of features (parameters) which best predict three classes. In addition to quick clay, he also include a class for brittle soil material defined by high sensitivity and low remoulded shear strength. The results showed that the machine learning models performed better than existing charts, especially regarding the amount of points incorrectly classified as quick clay.

A followup article where a diverse dataset of 240 CPTus were analyzed was released a year later,

(Valsson, 2019). Here he found that using three parameters when training outperformed models with two parameters. Normalized values for  $B_q$ ,  $R_f$  and  $q_e$  were found to be the best combination for the K-nearest neighbor model. Techniques for visualizing the data in two and three dimensions were shown to give the reader a intuitive understanding of how the model separate the classes.

In 2019, Christian Godoy Leiva wrote a master thesis on the topic, (Godoy, 2019). A year later he released an article summarizing the findings, (Godoy et al., 2020). Two testing sites with several CPTus were trained with three machine learning algorithms. The performance of the models were plotted as a function of how many CPTus were used during training, showing how fast each algorithm manage to learn. The performance of the models were compared against classification charts to show which did better. While the models performed very well on one of the datasets, lower accuracy scores were found on the second dataset. One of the datasets used in his article will also be applied to this thesis.

Lastly it is worth mentioning that a lead up to this master thesis is a specialization project Berrum and Skaar (2020), where KNN was used for classification of quick clay. A modified edition of the dataset used in this project will be used as a part of this thesis.

## 1.2 Objectives

The main objectives of this project are

1. Describe the theoretical framework behind different machine learning algorithms and its application in Python.
2. Compare the performance of the different algorithms prediction of soil layering and distinguish between quick and other material layering.
3. Study the influence that different datasets has on the performance of each ML algorithms.
4. Analyze the effect of changing the amount of input parameters on each ML algorithms ability to accurately predict quick-clay layering.

### 1.3 Limitations

This work will only focus on the separation of two classes, respectively quick clay and other material (other material), as the main goal is to evaluate which machine learning algorithm most accurately can predict the occurrence of quick clay. For other types of soil, classification by methods such as classification charts have previously been thoroughly researched and will not be included.

CPTus from NGTS Tiller-Flotten site are limited to 20 meters depth as data below show a lot more variance. Therefore, less noise and random fluctuations of the data are included in the training of the machine learning algorithms. This can possibly result in a higher accuracy score than if all depths were used.

$Q_t$ ,  $F_r$  and  $B_q$  were chosen as the only input parameters to facilitate the comparison of the machine learning algorithms. These are included since they are well-known derived CPTu parameters in geotechnical engineering. As [Valsson \(2019\)](#) already has done work on optimizing the parameter selection, this will not be a focus in this thesis.

While the dataset analyzed in this thesis involve CPTus from all over Norway, the overall diversity of the points are relatively low as the datasets consist of total 45 CPTus with 32 of them coming from one site and seven of the remaining are only used for testing purposes. The relatively low amount of CPTus in the datasets makes it difficult to say that the models trained are truly generalizable.

### 1.4 Approach

When approaching the problem, an insight into different machine learning techniques is required. Therefore the project starts with a literature study of the theoretical background of different machine learning models. One such technique, namely K nearest neighbor, is already used in a project leading up to this thesis. Publicly released lectures from Massachusetts Institute of Technology ([Winston, 2015](#)) are a vital source for understanding the theoretical framework of different machine learning models. Google Scholar is used to find relevant articles.



Informative videos have been very helpful in order to give a visual understanding of machine learning algorithms.

To implement the different machine learning algorithms, the programming language Python is used (Van Rossum and Drake Jr, 1995). A total of seven machine learning techniques is to be studied and compared, which in turn mean that several parameters needs to be tuned and experimented with. A code is developed to illustrate where the border between the classes are set after training to visualize how the models separates the data. Implementation of the machine learning algorithms is done using available libraries and include the following: scikit-learn (Pedregosa et al., 2011), HP-ELM (Akusok et al., 2015), XGBoost (Chen and Guestrin, 2016a) and keras (Chollet et al., 2015). Scikit-learn is a wide machine learning library and the following three will be used: K nearest neighbor (KNN), support vector machine (SVM) and random forest (RF) classifiers. The Keras library is built upon the Tensorflow architecture developed by Google (Abadi et al., 2015), and give a simple interface for the deep learning algorithms (DNN and CNN). HP-ELM and XGBoost contains the algorithms their names suggest.

Three datasets are used to evaluate the different models. One dataset is solely used for training, one is used only for testing and one is used for both training and testing. All points in the CPTus used in the datasets are given labels whether they are quick or not based on laboratory tests at their respective sites, however some points have been labeled unknown due to a lack of data or testing. The prediction of the algorithms are compared to the labels from the lab which indicates how accurate they are.

## 1.5 Structure of the Report

The rest of the report is structured as follows. Chapter 2 is an introductory literary study to basic machine learning concepts. Models such as neural networks, decision tree and nearest neighbor will briefly be presented. A representation of the sites used as basis for the datasets is presented in chapter 3. Soil layering profiles and basic soil parameters of each site are shown. Chapter 3 also describes how the CPTu data is prepared, which parameters are chosen and how the data is processed in Python.

Results are discussed and presented in chapter 4 and are structured into several sections. Section 4.1 describes the results of training and testing the algorithms on the NGTS Tiller-Flotten dataset. Section 4.2 shows results when the models are trained on six sites around Norway and tested on the NGTS dataset. The models trained on six CPTus will be further validated on a third dataset consisting of seven CPTus in section 4.3.

The report is concluded with chapter 5 which gives a summary and conclusion of the work done, with some discussion about the validity of the results. Recommendations for further work on the topic are included. At the end, the Appendix contains plots, figures and diagrams that were not chosen to be added to the main text, but in some way give a good insight into further results for the discussion as well as background data.

## Chapter 2

# Machine learning

As the level of computational capability of computers has increased drastically over the years, new ways of solving problems have been possible. More and more machine learning techniques have been developed to interpret and evaluate data ([Ayodele, 2010](#)). These techniques are part of a cutting edge technology in several areas of knowledge such as business, medicine and engineering. Each machine learning technique has been implemented in a huge range of problems, and new research is carried out continuously. A diverse range of machine learning algorithms may be implemented to the same problem with varying results. Therefore it is of interest to compare how different machine learning algorithms solve a given problem.

In this chapter the concept of machine learning will be described before going into detail of the different algorithms applied in this thesis.

### 2.1 Machine learning concept

Machine learning algorithms are based on a computational process where input data is given in order to obtain particular output data without directly specifying how to achieve this. The algorithms are able to adapt the way they learn through experience so they can perform their tasks better and better. In his textbook [Alpaydin \(2020\)](#) defines machine learning as "Programming computers to optimize a performance criterion using example data or past experience".

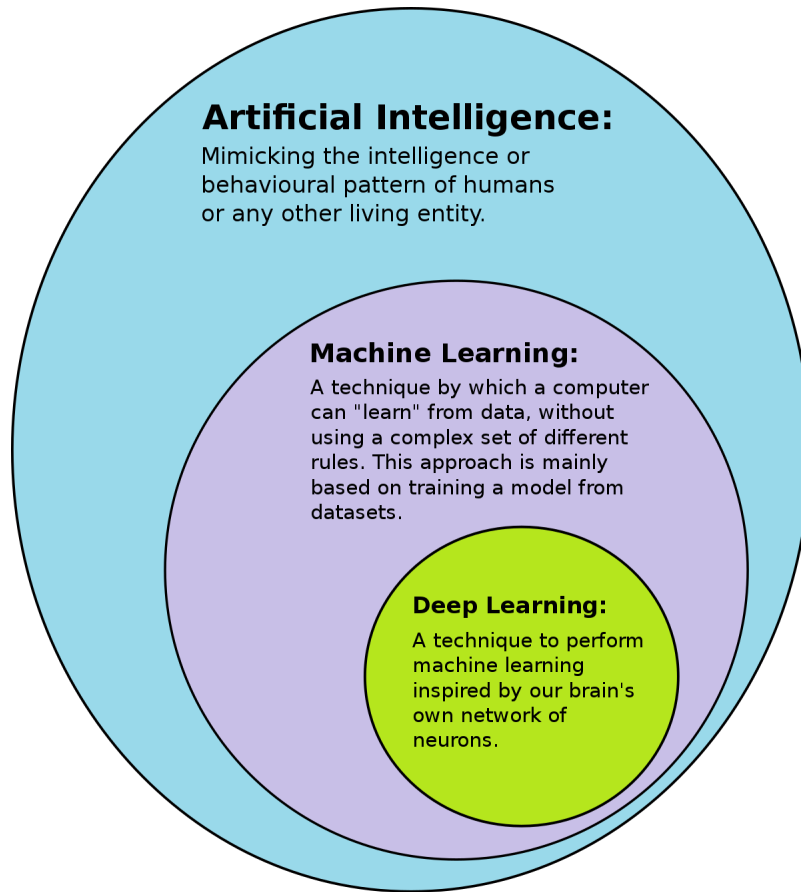


Figure 2.1: Relationship between AI, ML and Deep Learning

To adapt or train the algorithms, input data is given together with desired output data. Thereafter the machine learning algorithms learn how to produce the desired outcome from the training inputs, and furthermore generalize to produce desired outcomes from data which the algorithms have not yet been trained on. This is referred to the "learning" part of machine learning. The learning process will continue as long as the algorithm has new input data available.

A terminology used in machine learning about how well a model learns and generalizes data is overfitting and underfitting. These are the two biggest causes as to why a machine learning algorithm may experience poor performance. An overfitting model learns the noise and random fluctuations in a training data as concepts by the model which do not apply to new data. Overfitting negatively impacts the model's complexity and the ability to generalize. A model experiencing underfitting struggles to model the training data which in turn negatively affects the performance on new data (Dietterich, 1995).

As the goal of this work is to apply machine learning techniques to classify soils as quick or other material by using CPTu, the methods will use classification tools. In general, classification is a process in which data points are predicted to form a class. Classification predictive modeling is the task of approximating a mapping function ( $f$ ) from input variables ( $X$ ) into output variables ( $y$ ) (Yuan et al., 2012).

There are several ways to measure how well the machine learning models are performing on a given dataset. The most common used metric is the accuracy score, which simply gives the ratio between the amount of correct predictions and the total amount of predictions:

$$\text{Accuracy Score} = \frac{\# \text{ of correct predictions}}{\# \text{ of total predictions}} \quad (2.1)$$

In some cases however the accuracy score might not be the best metric to describe the performance. If one of the classes is significantly more represented than the other(s), the false positive rate or the true positive rate might give a more reasonable representation of the performance:

$$\text{False positive rate} = \frac{\# \text{ false positives}}{\# \text{ false positives} + \# \text{ true negatives}} \quad (2.2)$$

$$\text{True positive rate} = \frac{\# \text{ true positives}}{\# \text{ true positives} + \# \text{ false negatives}} \quad (2.3)$$

Machine learning algorithms can be classified as supervised and unsupervised based on what the purpose of the algorithm is. Most machine learning models is based on supervised learning. This type of learning consists of input and output variables and a machine learning algorithm which tries to learn the mapping function between these variables. The purpose is to map these functions well enough to predict output variables for new input data (Love, 2002). Learning is stopped as the algorithm achieves the desired performance. Among supervised learning algorithms are regression and classification problems. Classification problem is based on output variables put into categories, such as quick clay or other material. Regression problem predicts continuous responses in data (Mohri et al., 2018).

Unsupervised learning on the other hand only consists of input data. Here the purpose is to learn the underlying structure of the data without the solution. The algorithms are left alone to discover structures in the data. Among unsupervised learning algorithms are clustering and association problems. Clustering discovers how the data groups up based on certain attributes. Association discover basic rules that describe the data (Mohri et al., 2018).

Ideally, machine learning will partly imitate the way human beings processes input to conclude a task (Mohri et al., 2018). One task could be recognizing different patterns, for example using CPTu input data to distinguish different soil layering in the ground. Machine learning has the potential to recognize more complex and complicated patterns than most humans do and may therefore be a big asset in engineering practice. Following is a basic representation of the different machine learning algorithms used in this thesis.

## 2.2 Neural network algorithms

Neural networks are one of the most used deep learning algorithms and is inspired by the structure of how biological neurons in a human brain signal each other. The structure consists of node layers: an input layer, one or more hidden layers and an output layer. Every node in one layers is connected to all the nodes in the next. (Winston, 2015).

Weights are initialized randomly and assigned to each connection between nodes in the network. This is to highlight the importance of the different variables contribution to the output layer. The weights are multiplied with each node and thereafter summed up. This sum plus a bias is then put through an activation function which truncates the value of the node to a range between 0 and 1. The bias is a number which controls the inactivity of a node where it is only considered meaningfully active when the weighted sum is above a certain threshold (Winston, 2015).

In general the mathematical function that shows the connection between two layers (layer 0 and

1) in a neural network can be written in vector form as:

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^0 \\ a_1^0 \\ \vdots \\ a_n^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_0^1 \\ a_1^1 \\ \vdots \\ a_n^1 \end{bmatrix}$$

In a more compact vectorized form:

$$\mathbf{a}^1 = \mathbf{w}\mathbf{a}^0 + \mathbf{b} \quad (2.4)$$

where weights ( $\mathbf{w}$ ) are multiplied and summed with neurons from one layer ( $\mathbf{a}^0$ ) and a bias ( $\mathbf{b}$ ) to achieve the activation values in neurons in the next layer ( $\mathbf{a}^1$ ). Figure 2.2 shows the structure behind a basic neural network with one hidden layer.

In order to have the value of each neuron between 0 and 1 to imitate the biological analogy of neurons being either active (1) or inactive (0), the sigmoid function is often used for hidden layers and softmax function is used for the output layer (Winston, 2015).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.5)$$

The sigmoid function acts as an activation function. It transforms very positive values into 1 and very negative values into 0. Values close to 0 end up somewhere between 0 and 1.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2.6)$$

The softmax function transforms a vector ( $\vec{z}$ ) of real values into a vector of K real values that sum up to 1. All input values are transformed into values between 0 and 1.

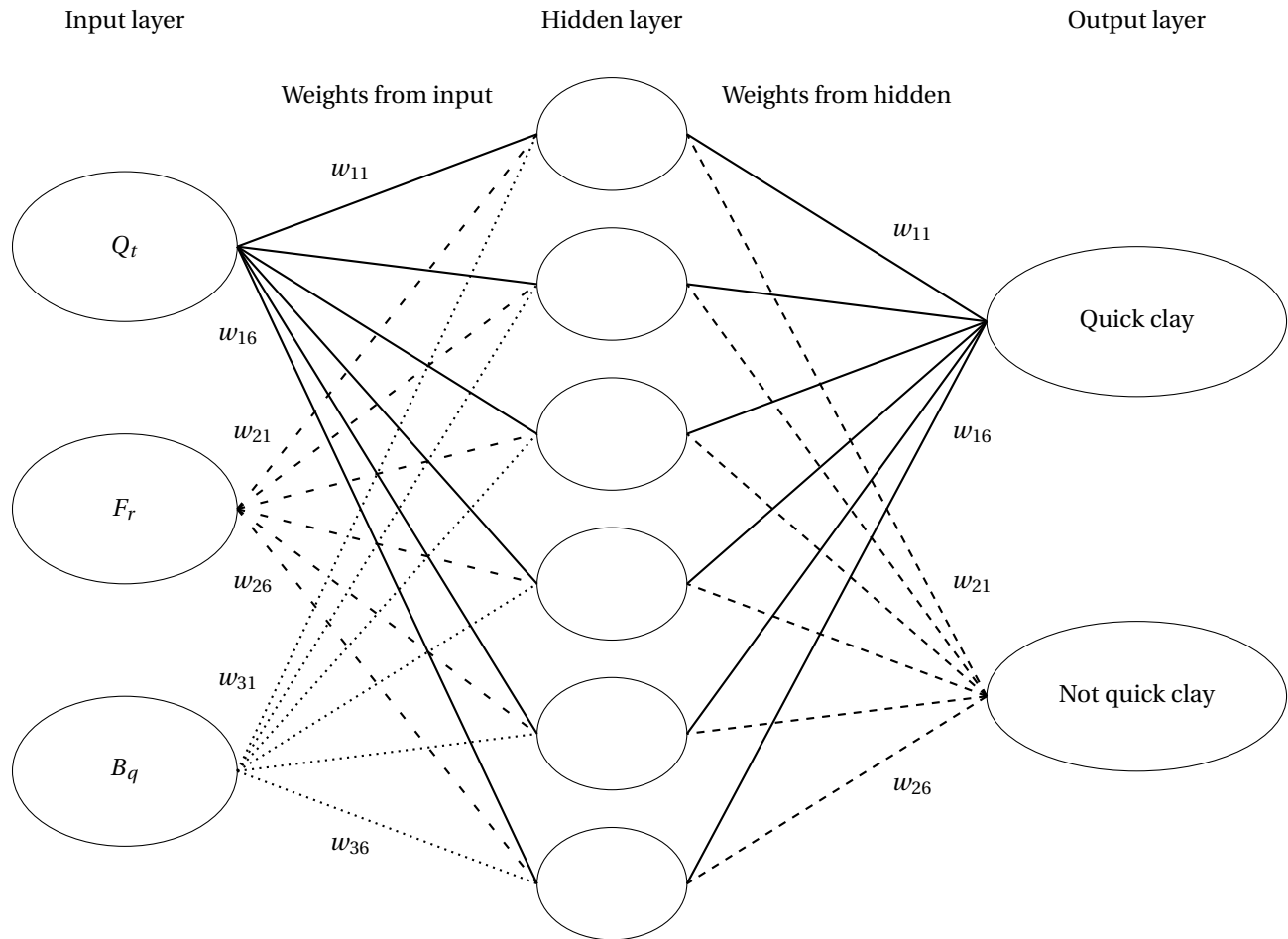


Figure 2.2: A principal sketch of how a neural network classifying quick clay could be structured. In this scenario the input layer consists of three nodes:  $Q_t$ ,  $F_r$  and  $B_q$  with associated weights to one hidden layer. Weights from the hidden layer determine the output layer, if the clay is quick or not.

Inserting the sigmoid function into equation 2.4 gives the activation of the neuron in the last layer ( $a^l$ ):

$$a^l = \sigma(w^l a^{l-1} + b^l) \quad (2.7)$$

which is correlated to the weights of the last layer ( $w^l$ ), the activation of the previous layer ( $a^{l-1}$ ) and the bias of the last layer ( $b^l$ ).

If the determined output is within a preset threshold it activates the node and passes data to the next layer of the network. A feedforward neural network is defined as data being passed from one node to another in this specified manner.



A way to evaluate the accuracy of a neural network algorithm is by using a cost function. The most commonly used cost function is the mean squared error (MSE).

$$C = (a^l - y)^2 \quad (2.8)$$

where  $C$  is the cost function,  $y$  is the true value and  $a^l$  is the predicted value. This cost function takes all the weights and biases in the neural network as input and gives one number (the cost) as output. Based on this number the algorithm gets an indication of how accurate those weights and biases are by comparing the true and predicted value of the input and output.

By minimizing this cost function the algorithm can ensure higher accuracy of fit for any given observation. The way a neural network algorithm learns is by adjusting the weights and bias to minimize the cost function until it reaches a point of convergence.

A way of reaching this convergence or minimized error is by gradient descent. This method utilizes the gradient in order to find the steepest decrease to the local minimum of the function, as known from calculus. It is beneficial to perform this action step wise with step sizes proportional to the slope to eliminate overshooting. In summary: compute the gradient descent, take a small step in that direction, and repeat until convergence at minimum ([Sanderson, 2017](#)).

The core algorithm for how a neural network learn is named backpropagation. This algorithm computes the gradient descent with calculating the derivative of the cost function with respect to weights and biases by the chain rule from the last layer to the previous layers. In this way the algorithm can calculate the error associated with each neuron from output to input ([Ding et al., 2011](#)). The basic principles of backpropagation using the chain rule in a neural network is shown below.

Equation 2.7 can be rewritten as:

$$a^l = \sigma(z^l) \quad (2.9)$$

where

$$z^l = w^l a^{l-1} + b^l$$

The partial derivative of the cost function with respect to the weights can be written as:

$$\frac{\partial C}{\partial w^l} = \frac{\partial z^l}{\partial w^l} \frac{\partial a^l}{\partial z^l} \frac{\partial C}{\partial a^l} \quad (2.10)$$

The partial derivative of the cost function with respect to the bias can be written as:

$$\frac{\partial C}{\partial b^l} = \frac{\partial z^l}{\partial b^l} \frac{\partial a^l}{\partial z^l} \frac{\partial C}{\partial a^l} \quad (2.11)$$

These partial derivatives of the cost function gives the gradient descent that nudges the weights and biases in the right direction to converge at the lowest cost.

$$\vec{W} = -\nabla C(\vec{W}) \quad (2.12)$$

where  $\vec{W}$  is a column of weights and biases and  $\nabla$  is the gradient (derivative). As the model gets more and more training examples (input data), the parameters adjust to gradually converge at the lowest error, resulting in weights and biases close to desired values (IBM, 2020).

### 2.2.1 DNN

Deep neural network (DNN) is a type of neural network which consists of two or more hidden layers between the input and output layers, and uses backpropagation algorithms to learn.

This architecture is able to model complex non-linear relationships. The additional hidden layers enable values from previous layers, giving the model a chance to address more complex data than a shallow network. According to [Rolnick and Tegmark \(2018\)](#) a deeper network has more power than a shallower one: "*The total number of neurons  $m$  required to approximate natural classes of multivariate polynomials of  $n$  variables grows only linearly with  $n$  for deep neural networks, but grows exponentially when merely a single hidden layer is allowed. When the number of hidden layers is increased from 1 to  $k$ , the neuron requirement grows exponentially not with  $n$  but with  $n^{1/k}$ , suggesting that the minimum number of layers required for practical expressibility grows only logarithmically with  $n$ .*"

### 2.2.2 CNN

Convolutional neural networks (CNN) is built up with with an approximately similar architecture as DNN. They differ from each other as CNN includes convolutional layers which employs the the mathematical operation convolution. In reality, convolution layers determines the output of neurons that are connected to local regions of the input neurons. Each convolutional neuron processes data only from its receptive field. Determination of the output is based on calculation of the dot product between the weights and the connected region of the input. ReLU (rectified linear unit) is applied to the output on similar grounds as the sigmoid function in order to find the activation from the previous layer. A pooling layer can be added to downsample the given input to reduce the number of parameters in that activation. The reader is referred to ([O'Shea and Nash, 2015](#)) for a more in depth introduction to CNN.

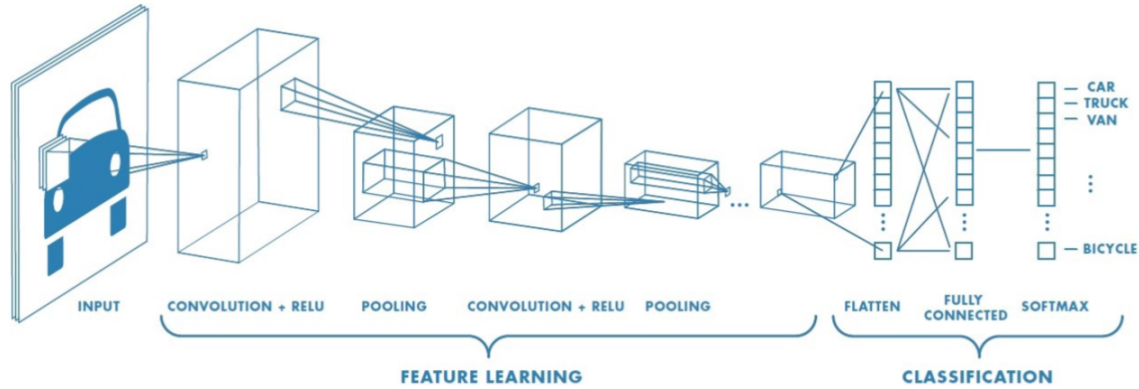


Figure 2.3: Principle of a CNN model for classification of images. The model consists of an input layer, convolution and pooling layer, hidden layers and output layer. Figure extracted from (Shyamel and Pingel, 2017).

### 2.2.3 ELM

The extreme learning machine (ELM) randomly sets the weight and biases for the input layer which are not changed. By randomly choosing weights to the input layer, an improvement of the generalization properties of the solution may be obtained as they produce weakly correlated features for the hidden layer. As weights and biases are randomly set, the output weights and input weights are independent of each other and gives a direct solution without iteration, unlike backpropagation. Since no iteration is performed, the calculation time is greatly reduced compared to other neural networks (Lai et al., 2020).

The architecture of an ELM model can be looked at as a single hidden layer feedforward neural network. Figure 2.2 gives the principal foundation of how this might look like. A description can be formulated as:

For  $N$  distinct training samples  $(x_i, t_i)$ ,  $i \in [[1, N]]$  and  $L$  hidden neurons, the formula for the estimated outputs with ELM is considered to be:

$$y_i = \sum_{j=1}^L \beta_j \sigma(w_j x_i + b_j) = t_i + \epsilon_i, \quad i \in [[1, N]] \quad (2.13)$$

where  $y_i$  are the estimated outputs,  $t_i$  the true outputs,  $x_i$  inputs,  $\sigma$  the activation function sigmoid,  $w_j$  the input weights,  $b_j$  the biases,  $\beta_j$  the output weights and  $\epsilon_i$  the noise.

Neurons in the hidden layer transform data from the input layer in two steps. By using the weights and biases from the input layer, data is first projected onto the hidden layer. Then the data is transformed by an activation function. A non-linear activation function is preferred as it increases the learning capability of the ELM algorithm (Akusok et al., 2015). The transformed data is thereafter used to find weights for the output layer. The algorithm only includes nodes that reduce the cost function.

ELM is a regression model which can be adapted to a classification model (Akusok et al., 2015). A target is created for each class if they are categorical and independent of each other. The targets are set to 1 if it is correct, and 0 if it is incorrect. Prediction of which class is correct is set accordingly to what target has the largest ELM output. The hidden layer output weights are the global optimal solution solved by the least square method to avoid falling into the dilemma of local optimum (Lai et al., 2020).

Selecting the correct model structure can prevent overfitting and accumulation of noise by limiting the ELM learning ability. A model with overfitting gives a worse generalized performance. An optimal generalized performance can be obtained by tuning model parameters or adding a regularization to the model.

### 2.3 Decision tree algorithms

Decision tree are part of supervised learning algorithms and are commonly used in classification problems and regression problems. Through learning simple decision rules from training data, the goal is to create a training model that can predict class or value of the desired target. These models aim to divide a search space into a number of subsets in a top-down recursive way (Zhong, 2016). Classes are achieved through sorting from the root node to the leaf nodes where the classification is provided. Figure 2.4 gives a basic understanding of the decision tree structure.

The nodes are divided into a subset of nodes by determining which way of separation is best through the Gini index. Gini index can be understood as a cost function that evaluates splits in the data set, and can be calculated by subtracting the sum of the squared probabilities of each

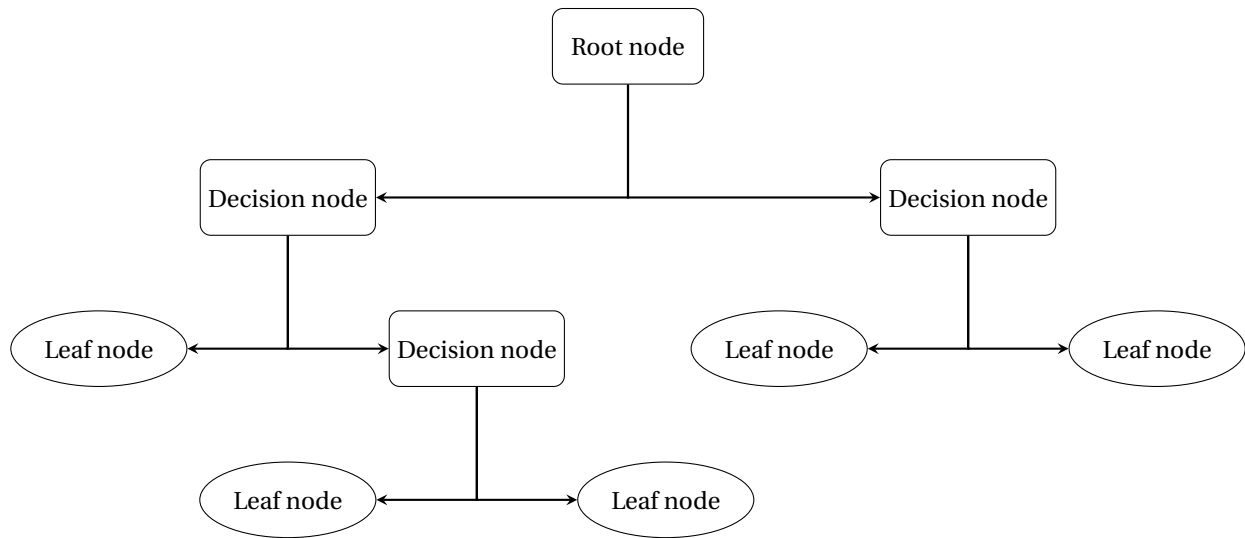


Figure 2.4: A principal sketch of how a decision tree algorithm can be structured. The model starts at the root node which represents the entire sample base and is divided into two or more internal nodes. These represent an attribute, the branches represent a decision rule and each leaf node represents outcomes.

class from one (Raileanu and Stoffel, 2004):

$$Gini = 1 - \sum_{i=1}^C (p_i)^2 - \sum_{i=1}^C (q_i)^2 \quad (2.14)$$

where  $p_i$  is the probability of success and  $q_i$  is the probability of failure.

### 2.3.1 RF

Random Forests are made out of decision trees. Combining the simplicity of these trees with flexibility gives a huge improvement in accuracy. A large number of these decision trees operating as an ensemble forms the Random Forest. The basic idea for class prediction is to let each tree in the forest predict a class, where the class with the majority of votes is chosen (Pal, 2005).

The general architecture behind a Random Forest algorithm can be described as follows. A bootstrapped data set is created from the original full dataset. The bootstrapped dataset is the same size as the original and consists of randomly selected samples from the original dataset. The same sample can be picked more than once. Each bootstrapped dataset is grown into a tree like the decision trees are, but with one important modification: instead of choosing the best split from each node among all the variables, a random subset of the variables are chosen and the

best split is made among those sampled variables. As the bootstrapped datasets are randomly generated and only a subset of the variables are considered at each step, the trees will vary from each other. This process is known as bagging (Liaw et al., 2002). The feature randomness this gives results in higher variation and diversification among the trees generated.

Commonly, 1/3 of the data from the original dataset does not end up in the bootstrapped dataset (Breiman, 2001). This is called the "out-of-bag" dataset. By running these "out-of-bag" datasets through each bootstrap tree, an estimation of the error rate may be obtained and an accuracy of the Random Forest algorithm can be estimated. For more in depth information regarding Random Forests, see (Breiman, 2001).

### 2.3.2 XGB

Similar to Random Forests, extreme gradient boost (XGB) is based on an ensemble of decision trees. Decision tree based algorithms are considered to perform really good when it comes to small and medium structured or tabular data.

Making an initial prediction is the first step in order to fit a XGB model to the dataset. The initial prediction is by default set to 0.5, meaning that for soil classification the probability of a sample in the data set being classified as quick clay is 50 %, but can in fact be set to any desirable value. Samples in the dataset that are labeled as quick clay and other material, will have observed values of 1 and 0 respectively. Residuals measures how good the initial prediction is, and are defined as the differences between the observed and predicted values (Chen and Guestrin, 2016a).

Furthermore, an XGB tree is fit to these residuals. A XGB classification tree starts out as a single leaf containing the calculated similarity score of the residuals. For classification purposes the similarity score can be calculated by

$$\frac{(\sum Residual_i)^2}{\sum [PreviousProbability_i \times (1 - PreviousProbability_i)] + \lambda} \quad (2.15)$$

where  $\lambda$  is a regularization parameter which reduces the prediction's sensitivity to an individual observation ([Chen and Guestrin, 2016a](#)).

Similarity scores are calculated for each node and added in order to determine the gain. The gain value controls the manner of which the tree is built. XGB trees are always grown to max depth first. The algorithm limits the tree afterwards by pruning. Nodes are pruned if the splitting of a node leads to negative gain due to the regularization.

Output values of each tree is calculated in the same manner as similarity scores in equation 2.15, although the numerator (sum of residuals) are not squared. New predictions are made by implementing gradient boost for classification ([Chen and Guestrin, 2016a](#)). The new predictions are thereafter used to grow new decision trees based on the new residuals. This process repeats until the residuals (difference between observed and predicted values) are very small, or the maximum number of trees are reached.

Tianqi Chen and Carlos Guestrin presented a paper on XGB in 2016 ([Chen and Guestrin, 2016a](#)). A number of explanatory sites have been made since then, for example the YouTube channel StatQuest with Josh Starmer ([Starmer, 2020](#)). For a more detailed review of the model, see these references.

## 2.4 Nearest neighbor algorithms

One of the earliest made machine learning classification algorithms is nearest neighbor classification. It can be applied in a broad way while still achieving a highly accurate score. The method aims to label unknown objectives while distinguishing two or more destination classes. Classification in general requires some sort of training data with given labels, making it a supervised learning method. The simplest variant is based upon an objective that inherits the label from the closest sample in the training set ([Seidl et al., 2009](#)).



### 2.4.1 KNN

K-nearest neighbor (KNN) is a variant of this classification algorithm, but unlike nearest neighbor it's extended to make a decision from the  $k$  closest neighboring points for any  $k > 1$ . The rule of decision combines labels from these  $k$  samples by simple majority voting or by weighting closer points more than more distant points in order to decide the predicted label for the new object. Figure 2.5 illustrates how the KNN method classifies a new point.

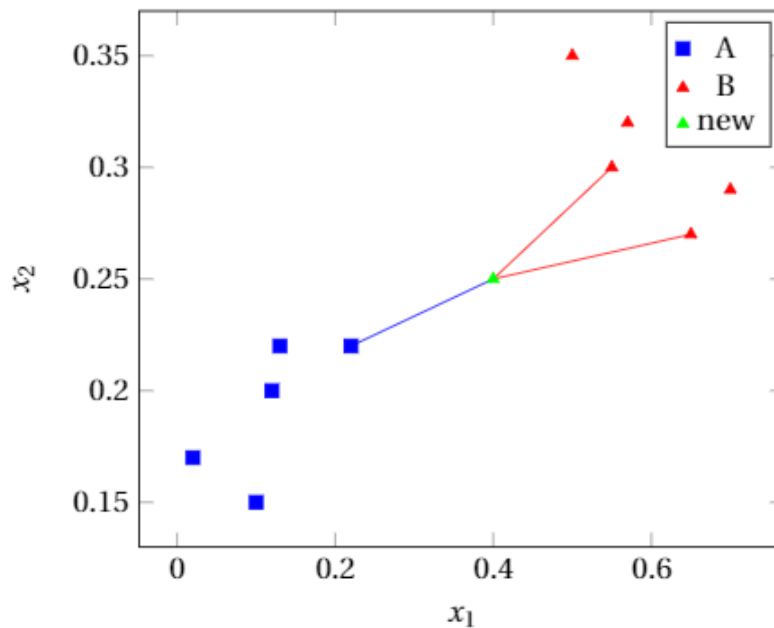


Figure 2.5: Graphic illustration of the KNN method with  $k = 3$ . Illustration extracted from (Berrum and Skaar, 2020).

KNN was the machine learning algorithm used in the project thesis Berrum and Skaar (2020), and a deeper understanding of the model can be found there, or in the book "Encyclopedia of Database Systems" (Seidl et al., 2009).

### 2.4.2 SVM

Support vector machines (SVM) is another algorithm regularly used for classification problems. The main objective of the algorithm is to determine a hyperplane that distinctly classifies the data points.

There are several ways to distinguish two classes of data points. SVM constructs hyperplanes (a

separating threshold) to separate the classes. The hyperplane is determined based on a maximal margin classifier. The margin is defined as the shortest distance between the data point of each class and the hyperplane. Implementation of the maximum margin provides some reinforcement so that future data points can be classified with more confidence. However, maximal margin classifiers are very sensitive to outliers in the data set. To handle faulty data points, SVM algorithms are modified by a "soft margin" that allow some data points to cross the separating hyperplane without affecting the final result (Noble, 2006). The hinge loss function on a training set  $(\mathbf{x}_i, y_i)$  is implemented:

$$\max(0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i - b)) \quad (2.16)$$

where:

$y_i \in [-1, 1]$  and indicates which class  $\mathbf{x}_i$  belongs to

$\mathbf{x}_i$  = p-dimensional vector

$\mathbf{w}$  = normal vector to the hyperplane

$b$  = bias

If  $\mathbf{x}_i$  lies on the correct side of the margin, the function equals 0. Data on the wrong side of the margin gives a value proportional to the distance from the margin (Zhang et al., 2004). Cross validation is implemented to determine what soft margin results in the best classification. Implementing this soft margin is called a support vector classifier. The data points on the edge and within the soft margin are called support vectors. Optimization is achieved by minimizing the expression:

$$\left[ \frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i - b)) \right] + \lambda \|\mathbf{w}\| \quad (2.17)$$

where:

$\lambda$  = parameter which ensures  $\mathbf{x}_i$  lies on the correct side of the margin

In situations where the dataset has a lot of variation and overlap, the separation of the classes based on hyperplanes might be problematic as the number of misclassifications increases. The kernel function provides a solution to this problem by moving the data into a higher dimension ([Zhang et al., 2004](#)) where the support vector classifier easier can separate the data.

## Chapter 3

# Presentation of datasets

Three datasets are used to train, test and compare the machine learning algorithms to the in-situ layering decided by sampling and laboratory investigations. Dataset I consists of 32 CPTus from the NGTS Tiller-Flotten site. Dataset II consists of six CPTu soundings from six different locations in Norway. Dataset III consists of seven CPTus from Saksvik in Trøndelag. The database has three different sounding methods, namely SCPTu (CPTu with recorded shear waves), RCPTu (CPTu with recorded resistivity) and CPTu. Raw data graphs from each individual sounding can be found in the appendix, see [A](#). All three datasets has highly sensitive to quick clay and other material present. In this Master's thesis only two groups of materials are considered in order to put emphasize on quick clay detection rather than predicting all types of soil.

### 3.1 Dataset I: NGTS Tiller-Flotten

NGTS (Norwegian Geo-Test Site) located at Tiller-Flotten is one of five established test sites involved in the R&D Program Norwegian Geo-Test Sites – NGTS supported by The Research Council of Norway Infrastructure. NGTS is lead by NGI together with NTNU, SINTEF/UNIS and Statens Vegvesen with the intention of testing and verifying new methods for ground investigations and field procedures. Tiller-Flotten was chosen as a testing site for the program due to the presence of highly sensitive quick clay located close to Trondheim.

Figure 3.1 shows a quaternary map from the area surrounding the NGTS test site at Tiller-Flotten (NGU, 2021). The map indicates mostly marine deposits together with fluvial deposits and bogs. Marine sediments emerged in the area as a result of the glacio-isostatic uplift where melting ice caps relieved weight on land. After the marine clay was raised above sea level its been exposed to fresh groundwater flow leaching the salt ions, resulting in a sensitive clay. Soundings at the research site shows a sedimentation thickness of more than 50 m (L'Heureux et al., 2019).

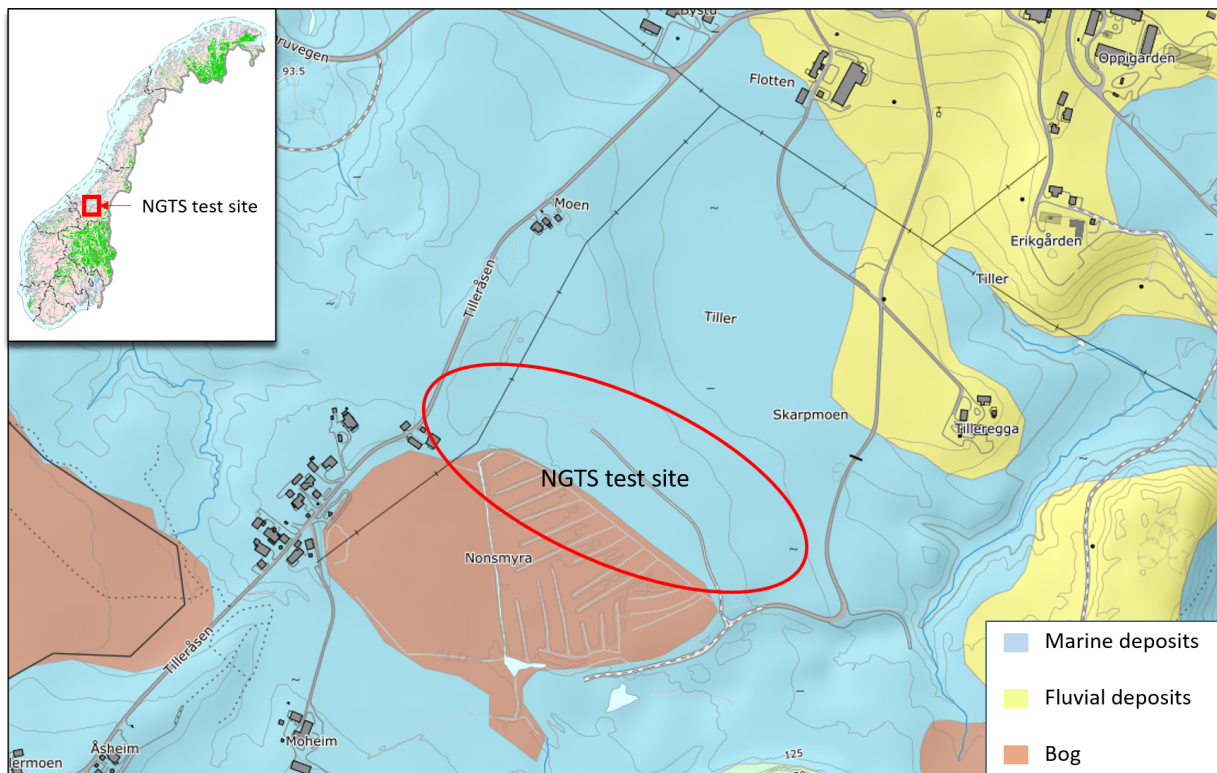


Figure 3.1: Quaternary map from NGU at the location of NGTS test site at Tiller-Flotten (NGU, 2021)

The layering of the site shows that the top 2 m consists of a desiccated and weathered clay. From 2 to 7.5 m depth lies a low to medium sensitive clay. 7.5 to 20 m consists of a very sensitive clay. Groundwater level in the area is located between 1 to 2 m below ground level and is underhydrostatic (approximately 20% of hydrostatic pore pressure) as a result of a downwards gradient due to groundwater flow and differences in elevation. The water content varies between 30-50 % and the bulk unit weight varies between 17-19  $kN/m^3$ . Research shows that the layering at the test site is relatively homogeneous, and therefore mean values of the soil profile, stratigraphy and index properties have been used. The reader is referred to (L'Heureux et al., 2019) for a

more thorough representation of the soil characteristics at the NGTS Tiller-Flotten site.



Figure 3.2: Location of the CPTu-soundings at Tiller-Flotten NGTS test site. Figure extracted from the NGTS quick-clay project, <http://www.geocalcs.com/datamap>.

Figure 3.2 shows the location of the CPTu tests at NGTS Tiller-Flotten site. The dataset consists of 33 CPTu tests. As is evident, the majority of the tests are located in a cluster to the south-east, while three tests are farther north-west. CPTu test TILC18 has been discarded from this data set as the test showed an elevated sleeve friction compared to the rest. Figure 3.3 shows a summary of the recorded values for the 32 CPTu tests used in the Tiller-Flotten dataset, the in-situ pore pressure  $u_0$  and the soil layering at the site. The entirety of the data set has been limited to 20 m depth as the recorded CPTu values farther down showed more variance.

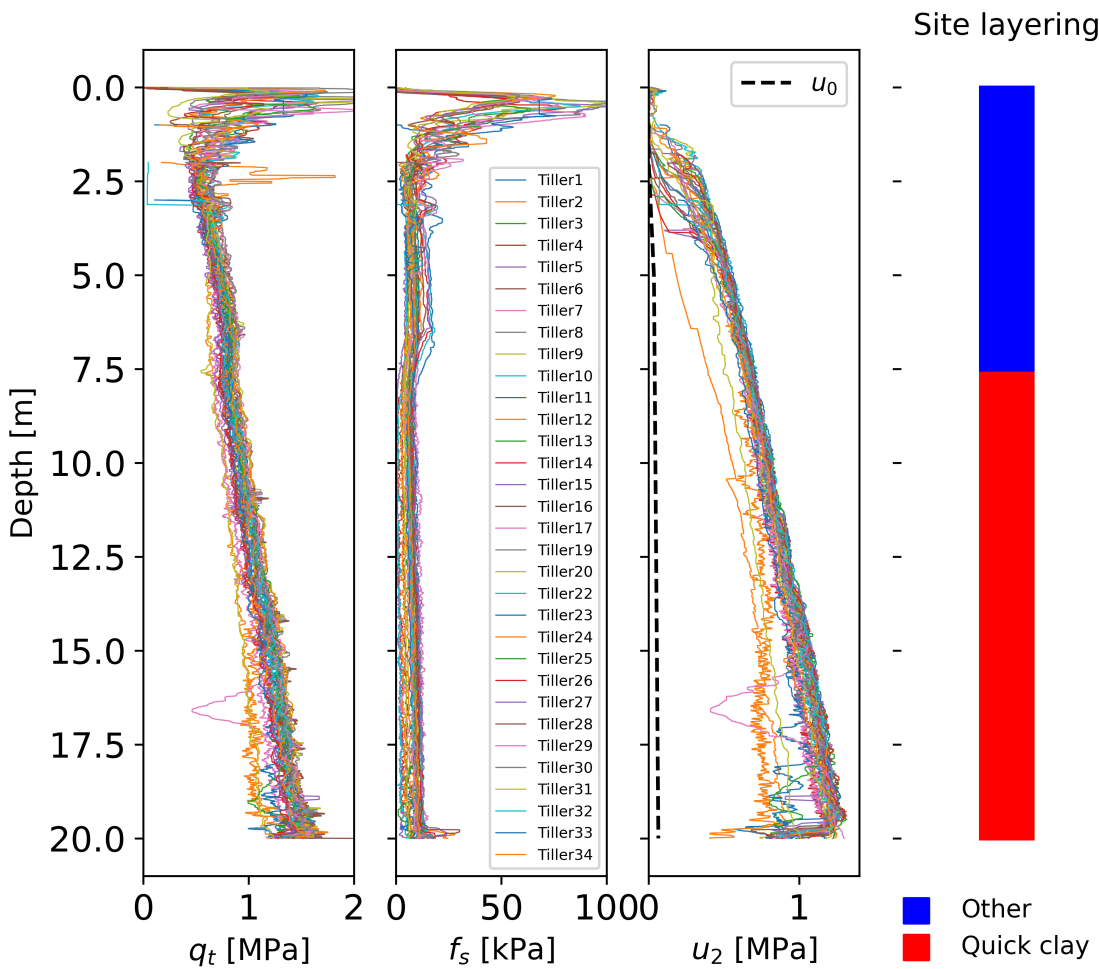


Figure 3.3: Summary of 32 CPTu tests at Tiller-Flotten NGTS test site. The diagrams show corrected tip resistance  $q_t$ , sleeve friction  $f_s$ , pore pressure plotted versus depth and soil layering profile.

Table 3.1 and figure 3.4 shows the number of data points in the Tiller-Flotten data set and which class they are labeled as. Even though there are more quick clay points, they are grouped in a smaller area in figure 3.4 compared to the other material material. Around 60 % of the dataset are labeled as quick clay due to the Tiller-Flotten site having a large quick clay layer starting at approximately 7.5 m depth. The ground conditions are rather homogeneous, which can be illustrated by the low variance in the quick clay data. The majority of the quick clay points are found in an area with boundaries  $Q_t < 10$  and  $F_r < 0.05$ .

Table 3.1: Number of data points labeled as either quick clay or other material from NGTS Tiller-Flotten CPTus. The majority of data points are labeled as quick clay, as the site layering shows a surplus of quick clay compared to other material.

	Quick clay	Other
N points	68926	40113
Portion	63.2 %	36.8 %

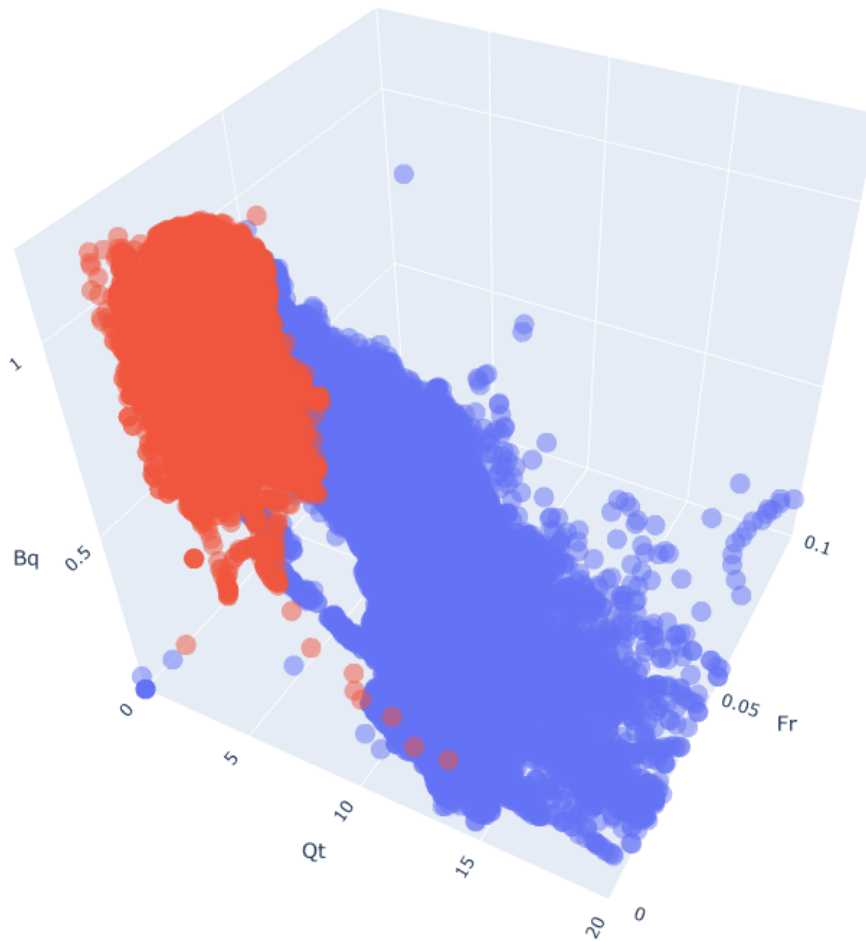


Figure 3.4: 3D plot of all CPTu data points from Tiller-Flotten with parameters  $Q_t$ ,  $F_r$ ,  $B_q$ . Red points are quick clay (points below 7.5 meters) and blue points are other material (points above 7.5 meters).



### 3.2 Dataset II: Multiple sites located in Norway.

Dataset II consists of CPTu soundings from six different sites around Norway. The sites concerned are Onsøy, Koa, Skatval, Nybakk-Slomarka, E6 Kvithammer-Åsen and FRE16 (Ringeriksbanen and E16 - the joint railway & road project). This is the same dataset used in the project thesis TBA4510 (Berrum and Skaar, 2020). Each site consists solely of one CPTu. Figure 3.5 shows the approximate location of each site used in the dataset.

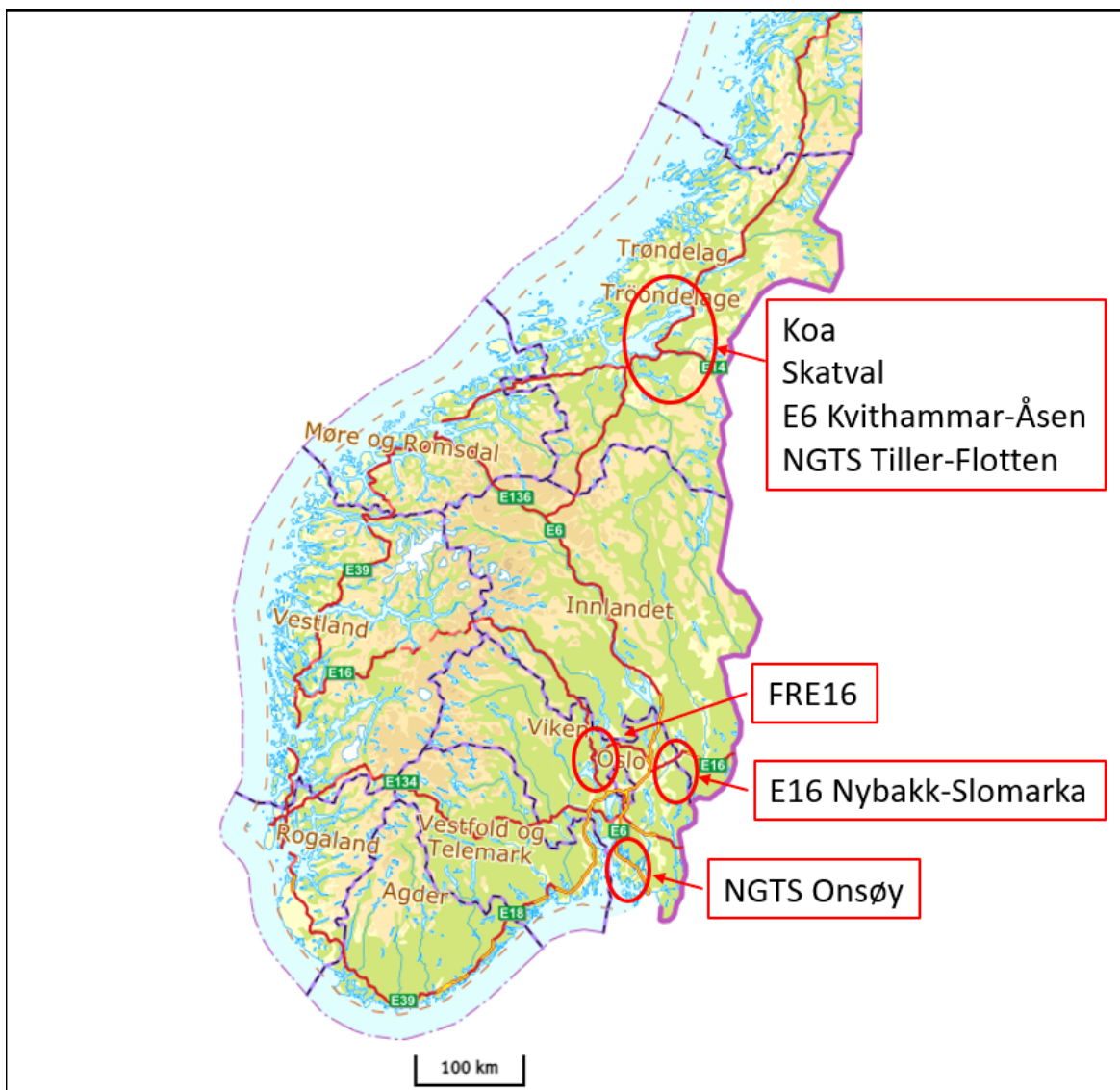


Figure 3.5: Overview of the approximate site locations that forms dataset II. The map is from Norgeskart (Kartverket, 2021).

Table 3.2: Summary of basic site properties at the six different locations in dataset II used in training the ML models.

Parameter	Onsøy	Koa	Fre16	E6	Skatval	Nybakk-Slomarka
Unit weight $\gamma$ ( $kN/m^3$ )	16-18	19.4	18-20	19-20	19.4	18.5
Water content (%)	65	30	27-38	25-38	32	35
Sensitivity	5-10	13-63	240-510	4-11	5-50	5-150
Plasticity index IP	25-45	8-25	6-18	11-24	11-17	8-17
Overconsolidation ratio OCR	1.2-1.7	3-4	2-4.2	1.2-1.8	2-4	2-6
Clay content (%)	50-65	50-53	(-)	(-)	35-43	40-47

Unlike the Tiller-Flotten dataset which have quite homogeneous soil layering and properties, dataset II consist of heterogeneous soil layering profiles and properties as its made up of six sites located at different parts of Norway. Table 3.2 summarize the variance of the soil parameters at the different sites which make up dataset II.

Onsøy is part of NGTS program similar to Tiller-Flotten, however it is classified as a soft clay site with no recorded quick clay present. The E6 site also lack presence of quick clay, and consists of soft to medium firm clay. The remaining sites in dataset II however have presence of quick clay. Soil layering are determined by sampling and laboratory investigations from each site and is visualized in figure 3.6.

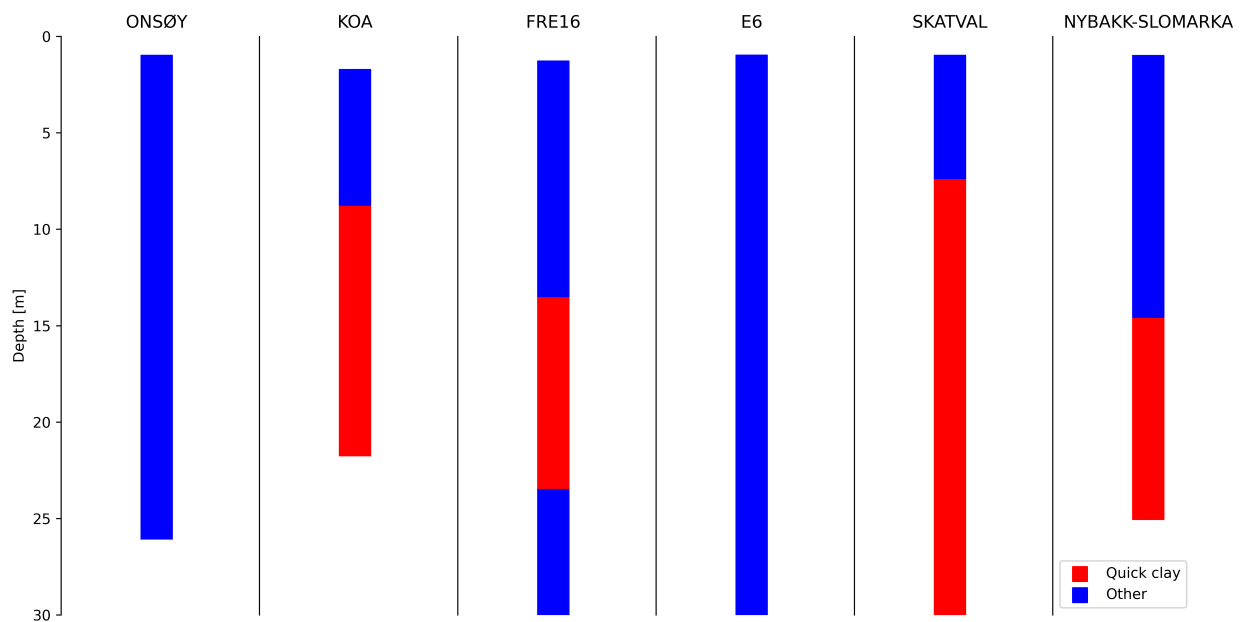


Figure 3.6: Soil layering with depth for each site in dataset II.

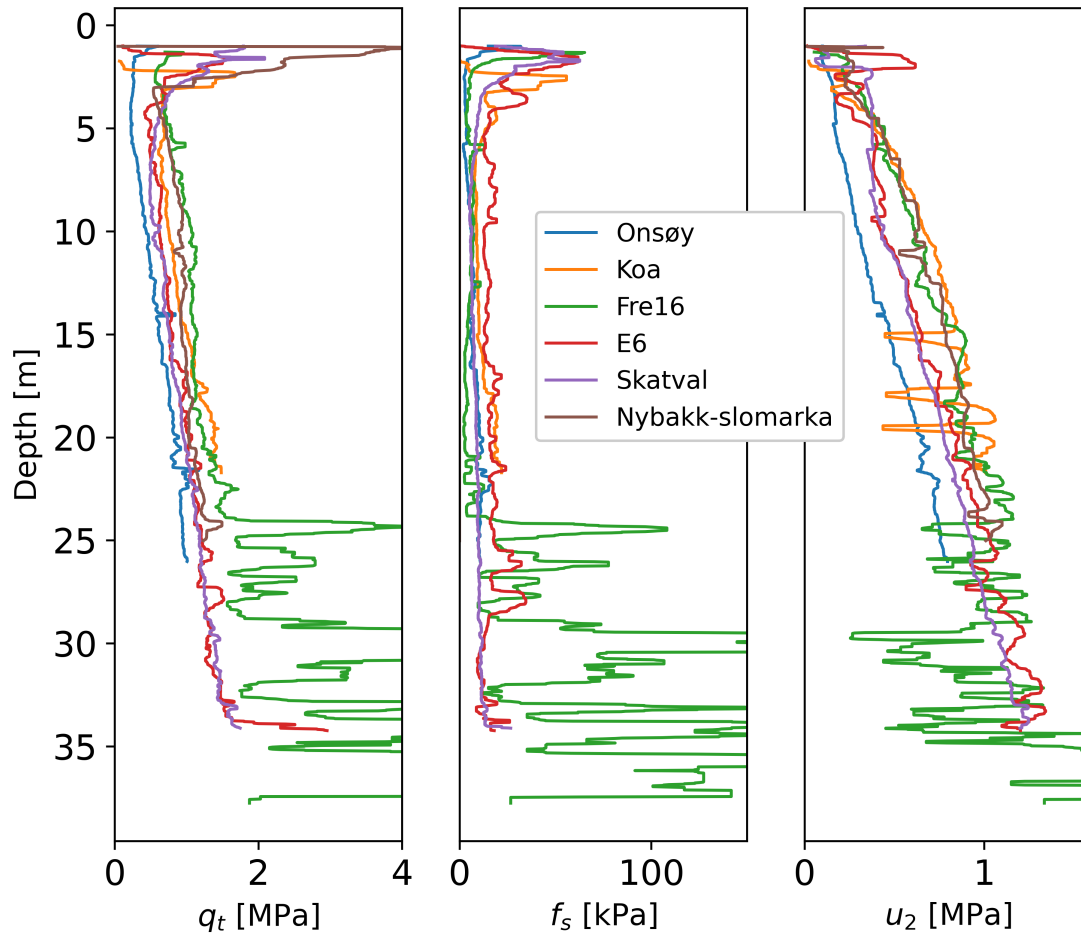


Figure 3.7: Summary of six CPTu tests from dataset II. The diagrams show recorded tip resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$  plotted versus depth.

A presentation of the recorded values  $q_t$ ,  $f_s$  and  $u_2$  for the six CPTu tests in dataset II is shown in Figure 3.7. It can clearly be seen that dataset II have CPTu raw data with more variance than the dataset from NGTS Tiller-Flotten.

Dataset II are split into two different approaches in order to get a more complete understanding of how the different machine learning models work. Approach 1 uses all data points from dataset II as basis for training the machine learning algorithms. Table 3.3 shows the total number of points and the portion ratio between the two classes. Approach 2 uses a reduced number of data points from dataset II where only one point per meter of CPTu data is included. Figure 3.8 visualizes the total number of data points in dataset II and which class they are labeled as. Unlike the Tiller-Flotten site, a more heterogeneous dataset will usually give more generalizable models.

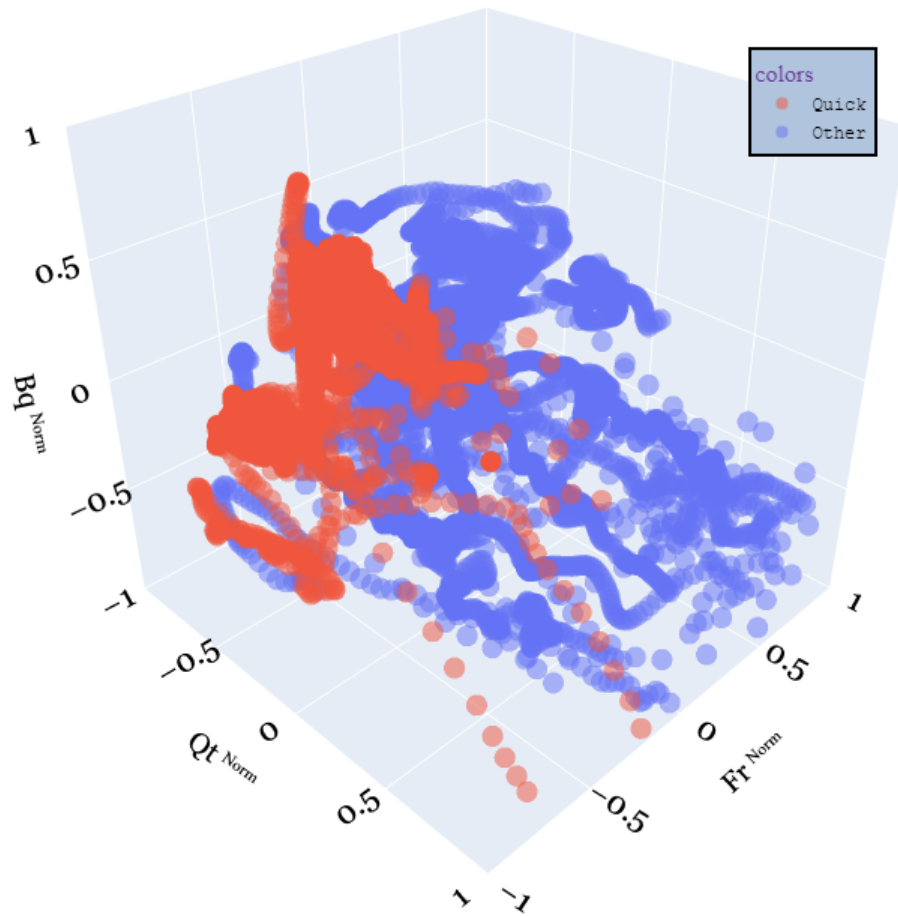


Figure 3.8: 3D representation of all CPTu data points from dataset II. Parameters  $Q_t$ ,  $F_r$  and  $B_q$  are normalized and plotted in a range from -1 to 1, see chap 3.4.1 for how the normalization are done.

Table 3.3: Approach 1 consists of all data points labeled as either quick clay or other material from CPTus in dataset II. Approach 2 consists of a reduced number of data points labeled either as quick clay or other material.

Full dataset	Quick clay	Other
N points	3810	6471
Portion	37.1 %	62.9 %
Reduced dataset	Quick clay	Other
N points	50	64
Portion	43.8 %	56.1 %

For further information regarding the different sites in dataset II, the reader is referred to the following articles: Skatval and Koa ([Paniagua et al., 2019](#)), Nybakk-Slomarka ([L'Heureux et al., 2018](#)) and Onsøy ([Gundersen et al., 2019](#)). Boring profiles for E6 Kvithammar-Åsen and Fre16 are included in the appendix, see figures C.1 and C.2 respectively.

### 3.3 Dataset III: Saksvik

Dataset III consists of seven CPTu soundings from Saksvik in Malvik kommune, Norway. The geotechnical ground investigations have been carried out by NGI and Rambøll and are connected to a new treatment plant that will partly be founded on a quick clay zone. Soil layering is varying, although a common profile is a stiff upper layer consisting of sand, silt and weathered clay. Below is a soft to medium firm clay. Only borehole 2 have confirmed presence of quick clay from sampling and laboratory testing, although there might also be quick clay in other areas. There is also brittle material present. Groundwater level is located 1 to 2 m below ground level and shows slightly over hydrostatic values with depth. Pore pressure measurements can be found in the appendix D.1. Water content in the clay is approximately 30 %, with plasticity index at 10 %. Sensitivity varies from 2 to 98.

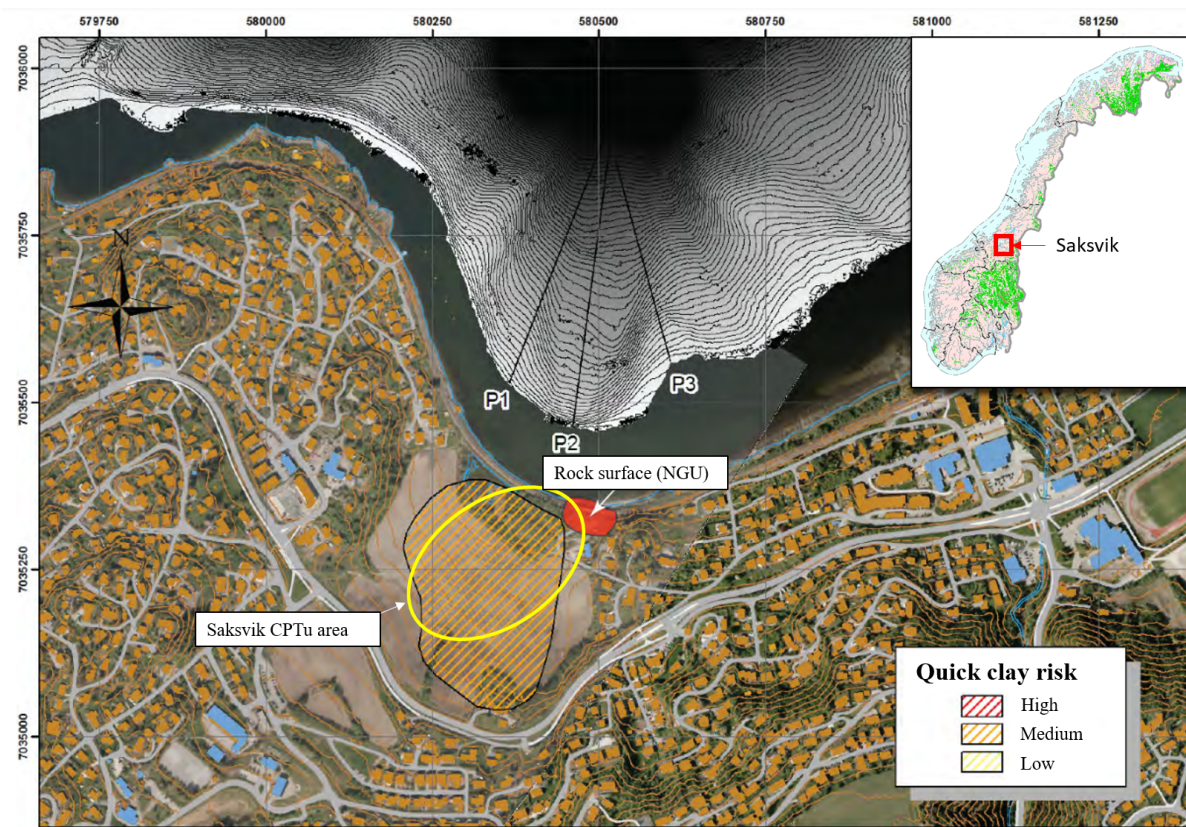


Figure 3.9: Map of Saksvik site shows the approximate area of CPTu soundings, quick clay zone and a nearby rock surface. The map is modified from attachment M page 2 in (L'Heureux, 2013).

Quaternary maps from NGU indicates mostly marine sea deposits and thick ocean deposits, which is an indication that quick clay can occur. Figure 3.9 shows a map from the surrounding

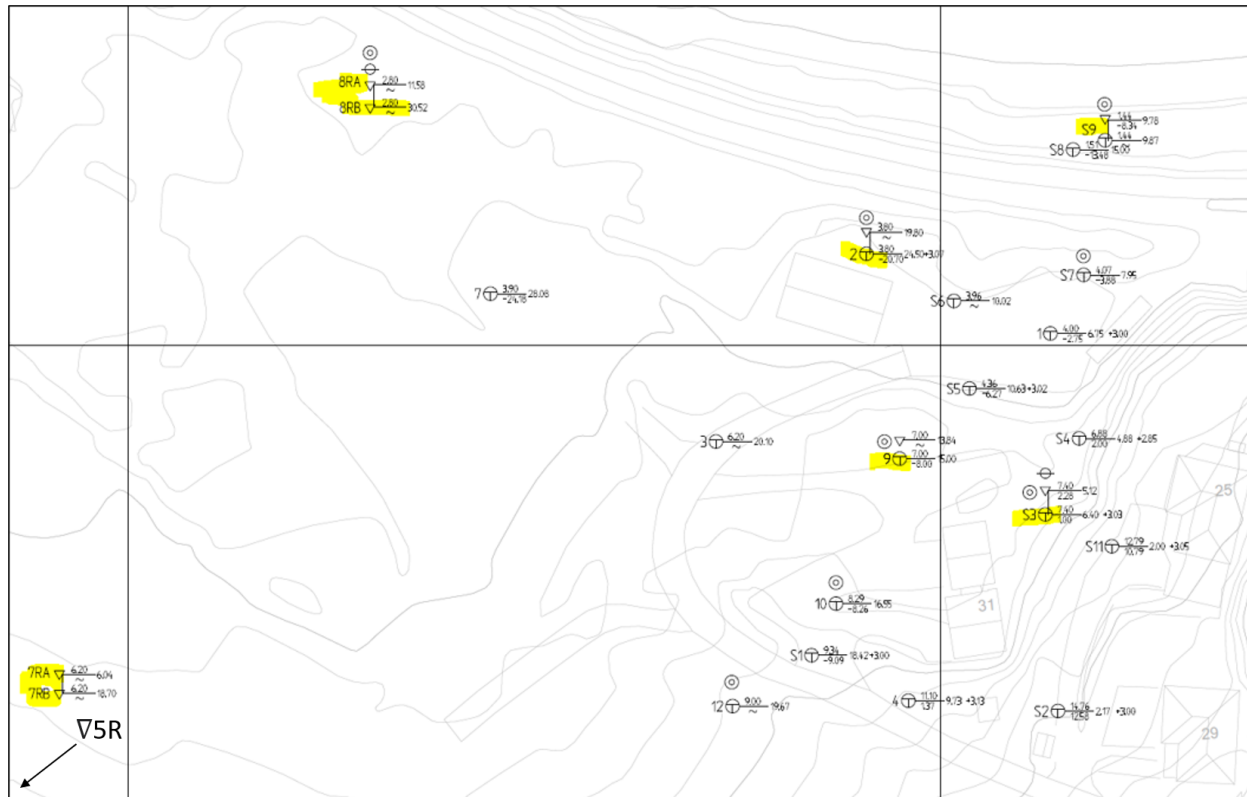


Figure 3.10: Map over geotechnical investigations at the Saksvik site.

area at Saksvik. There is a quick clay zone in Saksvik with medium degree of danger. The locations of the CPTus at Saksvik is given in Figure 3.10, and a summary of the seven CPTu tests used in dataset III are given in Figure 3.12. As seen from the map, many of the CPTu in the area is located in the quick clay zone.

Sampling have not been continuous, as some boreholes only have bag samples or no samples at all. A complete soil layering profile are therefore hard to produce, and some assumptions in the interpretation have been made, see Figure 3.11. CPTu 8R, 9 and S9 all consists of soil from depths where there are no samples. Quick or brittle clay may be present, however it haven't been determined by laboratory testing and it is therefore not possible to know for certain. Available total soundings from Saksvik are given in appendix B and index testing data are given in appendix C. In section 4.3 the depths with unknown lab data will be analyzed as if they were either completely quick clay or completely non quick clay.

Figure 3.12 gives a summary of the raw CPTu data from Saksvik and show high variance in both sounding depth and recorded values. Compared to Tiller-Flotten it is a more heterogeneous site

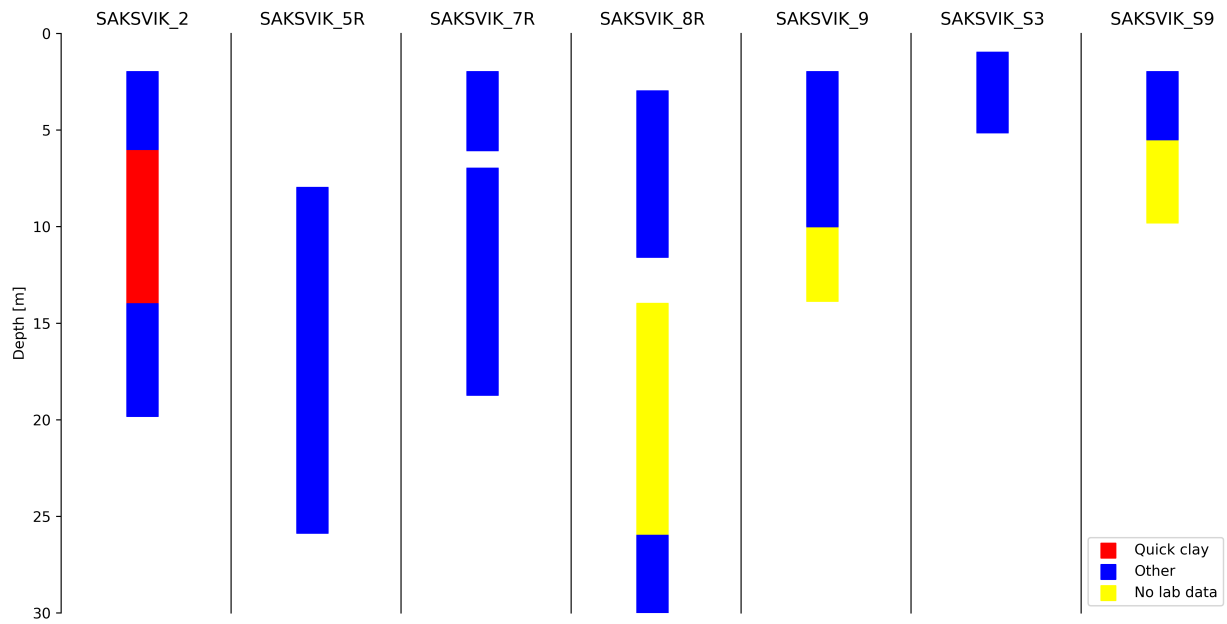


Figure 3.11: Soil layering of the CPTus at Saksvik.

and should make it more difficult for the models to predict correctly.

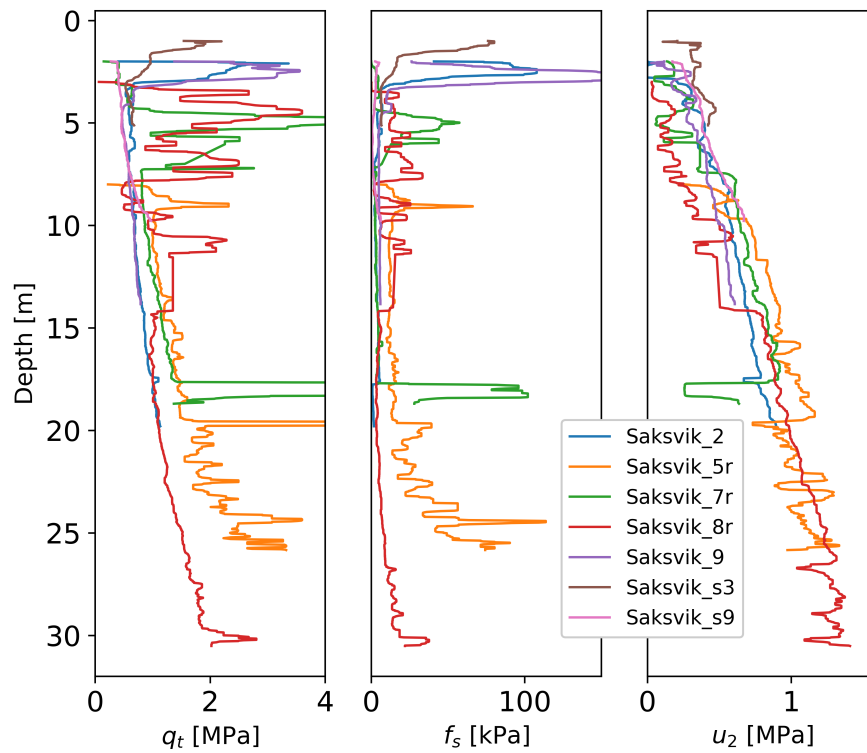


Figure 3.12: Summary of the seven CPTu tests from dataset III. The diagrams show corrected tip resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$  plotted versus depth.

### 3.4 Data preparation

The CPTu performs a continuous measurement of pore pressure, tip resistance and side friction through the soil and the data is gathered into raw data files. It is normally not possible to detect brittle or quick clay from the measured data alone, therefore the data is further processed to dimensionless, derived values which account for the overburden pressure at each depth (Sandven et al., 2015). Using these values combined with additional normalization and filtering should make it easier for the models to classify the soil.

#### 3.4.1 Parameter choice and normalization

The raw data was initially stored in Microsoft Excel files, so a script was made to read it into Python before converting it to a comma separated value format (csv). The Excel files contained CPTu data about depth, tip resistance  $q_c$ , side friction  $f_s$ , pore pressure  $u_0$  and  $u_2$  and the value of the area ratio  $\alpha$ . The tip resistance is corrected by the effects of pore pressure on the conical tip:

$$q_t = q_c + (1 - \alpha) \cdot u_2 \quad (3.1)$$

Four parameters were initially considered as candidates to be used as input for the machine learning models.  $Q_t$ ,  $F_r$  and  $B_q$  are commonly used in existing soil classification methods, and were naturally good candidates. Godoy (2019) adopted  $U_2$  which is another pore pressure parameter, so this was also considered. After initial testing however, the three first parameters were preferred as it is easier to visualize three parameters than four, and the additional parameter didn't prove to increase accuracy. Valsson (2019) performed a parameter selection study which shows that three parameters performed best, and using more parameters did not add value to the interpretation, but instead confuses the models. Since  $B_q$  seems to be the more popular of the pore pressure parameters it was preferred over  $U_2$ .

$$Q_t = \frac{q_t - p'_0}{p_0} \quad (3.2)$$



$$F_r = \frac{f_s}{q_t - p'_0} \quad (3.3)$$

$$B_q = \frac{u_2 - u_0}{q_t - p'_0} \quad (3.4)$$

$$U_2 = \frac{u_2 - u_0}{p_0} \quad (3.5)$$

where:

$q_t$  = corrected cone resistance

$p_0$  = total overburden stress

$p'_0$  = effective overburden stress

$f_s$  = side friction

$u_0$  = in-situ pore pressure

$u_2$  = measured pore pressure

When training the models it is beneficial to truncate the parameter space to a suitable range such as 0 to 1 or -1 to 1. The parameters will in this case contribute more equally to the decision of the model which is a desirable feature. The visualization of the training data also becomes simpler and negates the need for logarithmic scaling. The normalization of the parameters was chosen as follows:

$$Q_{t,normalized} = \frac{Q_t - Q_t^*}{Q_t^*} = \frac{Q_t - 7.224}{7.224} \quad (3.6)$$

$$F_{r,normalized} = \frac{F_r - F_r^*}{F_r^*} = \frac{F_r - 0.0194}{0.0194} \quad (3.7)$$

$$B_{q,normalized} = \frac{B_q - B_q^*}{B_q^*} = \frac{B_q - 0.82}{0.82} \quad (3.8)$$

$Q_t^*$ ,  $F_r^*$  and  $B_q^*$  can be looked at as some typical values found in soft clays and are calculated from the average values in dataset II. While the normalization doesn't force every point in the dataset into a -1 to 1 range, it is assumed that the vast majority of quick clay points should exist in this range. Table 3.4 shows how the normalized values transforms back to the actual  $Q_t$ ,  $F_r$  and  $B_q$  parameters.

Table 3.4: Conversion table from normalized values back to the actual parameters. Note that when the normalized value is 0,  $Q_t = Q_t^*$  and similar for the others. A normalized value of 1 gives  $Q_t = 2 * Q_t^*$  etc.

Normalized values	-1	-0.5	0	0.5	1
$Q_t$	0	3.6	7.2	10.8	14.4
$F_r$	0 %	0.97 %	1.94 %	2.91 %	3.88 %
$B_q$	0	0.41	0.82	1.43	1.64

For the training data, it was used a smoothing filter to reduce unwanted noise from the raw data. Both a running median filter and a running average filter was considered. As seen in figures 3.13a and 3.13b, the moving median filter is more able to filter out the sudden spikes which made it preferable over the moving average filter.

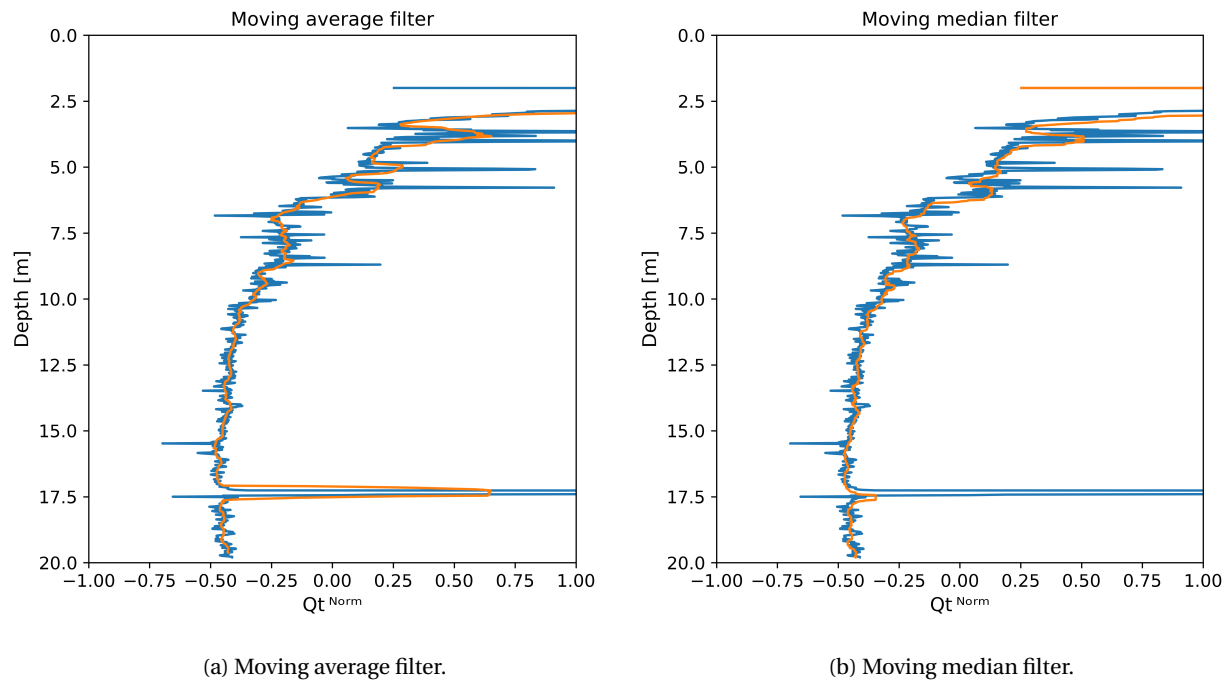


Figure 3.13: Comparison on how a moving average and a moving median in orange filters out unwanted noise from the original  $Q_t^{norm}$  plot from Saksvik CPTu 2 in blue.

### 3.5 The impact of faulty CPTu reading and errors in dataset

NGF (Norwegian Geotechnical Society) have developed a guide for performing CPTu soundings. Equipment and procedures for CPTu are selected based on the desired application class. There are four application classes, which depend on ground conditions and required accuracy. The classes range from very soft soil profiles to mixed and layered soil profiles. Any source of error are taken into account and must be less than the permitted minimum accuracy for the respective application class (Bæverfjord et al., 2010).

According to (Sandven et al., 2015), a number of factors such as equipment selection, planning and execution affect the measurement accuracy of CPTu:

- Probe measuring range and resolution
- Accuracy of calibration
- Temperature effect on probe and electric meters
- Zero point deviation for electric meters
- Saturation of pore pressure gauge
- Deviation in inclination
- Wear on equipment

Inaccuracies from CPTu data may lead to errors in machine learning models. Errors in raw data can lead to data points in the dataset being misclassified and thus confuse the models, resulting in a less accurate model prediction. An example is when the pore pressure gauge is not properly saturated and suction occurs, which affect the pore pressure readings.

High variance in CPTu data can give uncertain results. Figure 3.14 shows an example CPTu that have noisy data from an approximate depth of 24 m. Such noisy data may lead to difficulties for the models to create a decision barrier and therefore affects if the models correctly classify according to the desirable labels.

The machine learning models are also affected by the laboratory and in-situ data at each site. Correct values of soil density and pore pressure are important for the accuracy of the models, as

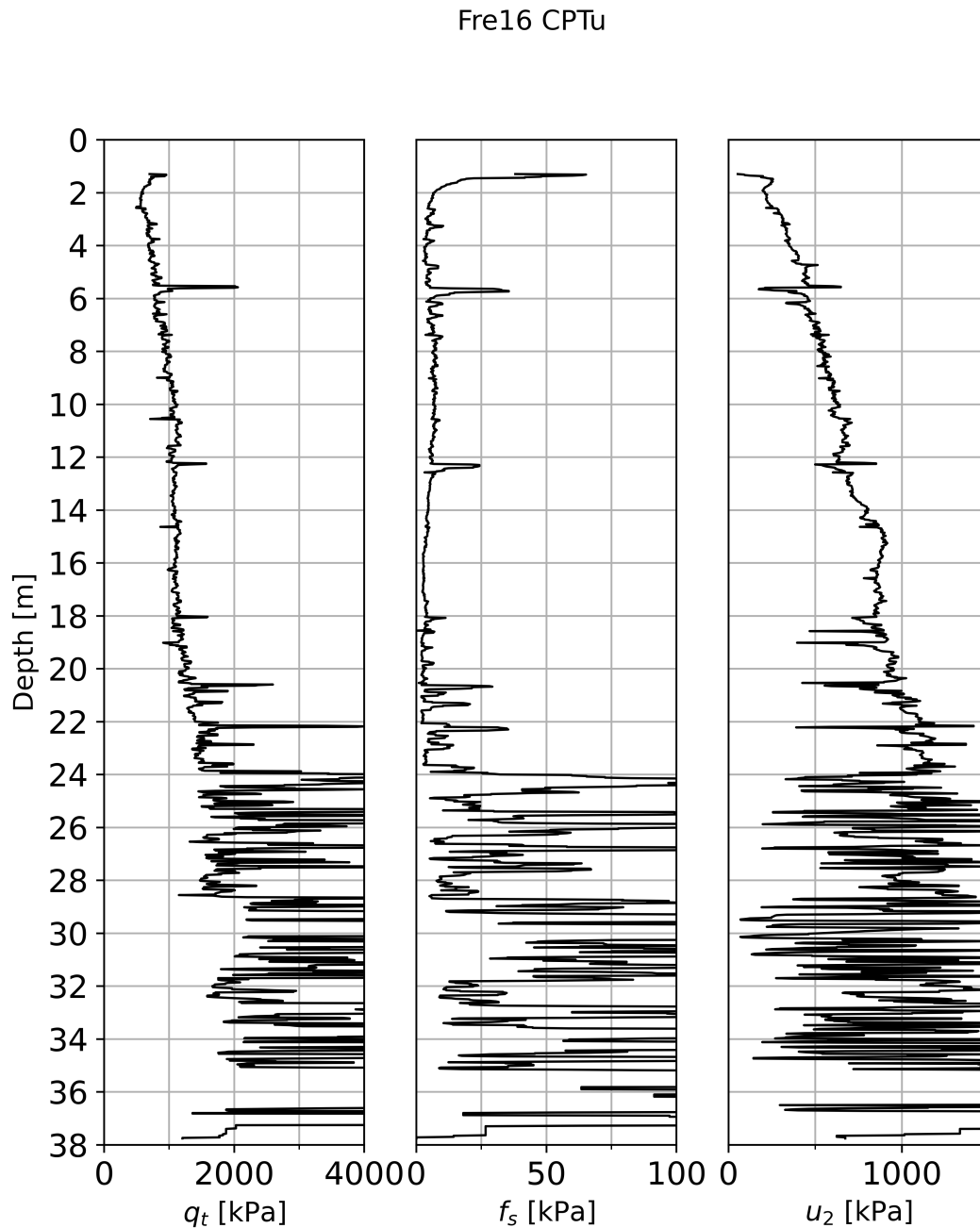


Figure 3.14: Example of a noisy CPTu with high variance in recorded values. CPTu from Fre16 (Ringeriksbanen and E16 - the joint railway & road project).

equations 3.2, 3.3 and 3.4 heavily rely on overburden stress and pore pressure. A slight deviation in for example pore pressure interpretation may lead to a consequential error in the normalized soil parameters. For example, piezometer recordings at NGTS Tiller-Flotten site show pore pressure 20% (approximately 2 kPa per meter) of hydrostatic conditions as a result of down-

wards gradient flow. Assumptions of hydrostatic pore pressure would in this case result in huge deviations.

Deficient laboratory data requires evaluation and engineering interpretation. Interpolation is widely used in engineering practice where soil information is missing. Interpolation in itself is a source of uncertainty, and deviation from real values might be large.

It is also worth noting that the ML algorithms in this thesis are supervised, meaning that the label for each prediction is known beforehand. The predicted values from ML are compared to the defined labels, or true values, which in this case is quick clay or other material. These "true values" also have a possibility for errors, as they are engineering interpretations from field tests and laboratory investigations and may deviate from the real values.

### 3.6 Implementation in Python

After the datasets were processed and ready to be used, a procedure to train and test the ML models was developed in python. As a time saving measure, only algorithms that already had existing implementation were considered. Some models were suggested by the supervisor, while some were included due to them being familiar beforehand.

Due to the amount of algorithms tested, significant work was put into understanding and tuning parameters. Experimentation with different values was needed to improve the performance of the models, however it was limited to the most important variables.

A procedure for training and testing the models on dataset I was developed inspired by the approach of a previous paper using this dataset ([Godoy et al., 2020](#)). Subsets of 5 CPTus picked randomly from the entire set were selected as training data to test an individual CPTu. This was repeated for all 32 CPTus with code ensuring that the testing CPTu was not a part of the testing set.

To visualize how the different algorithms separate the two classes, the plot range was meshed with points that were inputted to the trained machine learning models. In the 2D plots all points which the models classified as quick clay were scattered and the border between the classes were highlighted. For the 3D plots only the boundary were shown as plotting all the points would block out other parts of the figure.

## Chapter 4

# Results

The result chapter in this thesis is divided into three sections. The first consists of training and testing the algorithms on the NGTS Tiller-Flotten dataset (dataset I). Here the choice of model parameters are discussed in detail before analyzing plots of how quickly the algorithms learn to classify quick clay. The way each model separates the two classes are visualized through plots in two and three dimensions, and their shapes are evaluated from a psychical standpoint. While each algorithm have two 2D models, only one will be discussed in this section while the remaining can be found in the appendix. The two dimensional plots in the text will be showing the normalized parameters while the plots in the appendix [E](#) will show the parameters transformed back to their original values.

The second section involve training the models on dataset II (described in section [3.2](#)), and testing them on dataset I. The most interesting results are plotted and discussed in detail, while the performance of the different models are summarized in table format.

Lastly, the third section will be a validation of the trained models on the CPTus from Saksvik (dataset III). The performance will be shown through tables and profiles, and will be discussed in detail.

## 4.1 ML models trained and tested on dataset I

When training and testing on the same dataset it is necessary to figure out a suitable number for how many CPTus the models should be trained on. To solve this an analysis was performed to evaluate how the accuracy depended on how many CPTus the algorithm were trained on. Models trained with different amount of CPTus were tested on the CPTus not included in the training sets and their performances were listed. The results were visualized in box plot, see figure 4.1. The bottom and top of the boxes represent the 25 % and 75 % percentiles respectively, while the whiskers, which are the lines extending beyond the boxes, represent the 5 % and 95 % percentiles. The orange line inside the boxes are the median value of the performance. The percentiles for the whiskers were set manually as it is common in engineering practice to design for data that is 95% certain, for example the characteristic strength of building materials (CEN, 1992). The distance between the 75 % percentile and the 25 % percentiles is called the interquartile range (IQR) and says how big the spread of the data is (Wickham and Stryjewski, 2012).

A problem that occurs when training models with a high amount of CPTus is that there is few remaining CPTus to test them on. This would in practice mean that models trained on 30 CPTus only can be tested on the two remaining. To obtain more samples the procedure was repeated five times for each algorithm, increasing the sample size to ten for the models trained with 30 CPTus. This in turn resulted in the need to train 150 models for every algorithm which required significant computational time. A summary of the run time of the algorithms is given in table 4.1. Note that it is only in this analysis that such high amounts of models are to be trained, meaning that such a high time consumption in practical use will not occur. It does however show how big the difference between the "fast" and "slow" algorithms can be.

Table 4.1: Total run time of each algorithm.

Algorithm	Total run time [seconds]
ELM	16.9
XGB	47.5
RF	39.2
SVM	210.8
KNN	176.8
DNN	1072.8
CNN	3992.9



## Deep neural network

The deep neural network (DNN) model was implemented using the "sequential" class from the Keras library (Chollet et al., 2015) with two hidden layers. As there is no theoretical answer to how many layers and neurons to choose (Winston, 2015), some experimentation found that a total of 50 neurons would be more than enough for the task at hand. The sigmoid function, equation 2.5, was chosen as activation for the hidden layers while softmax, equation 2.6, was used for the output layer. For the DNN algorithm, a compromise between accuracy and computation time has to be made when choosing the learning rate of the network. The value for this parameter was set to 0.1, meaning that the change in the weight vector is scaled down during the back propagation. To prevent excess time use, early stoppage was configured to halt the training when the accuracy didn't improve within 50 iterations.

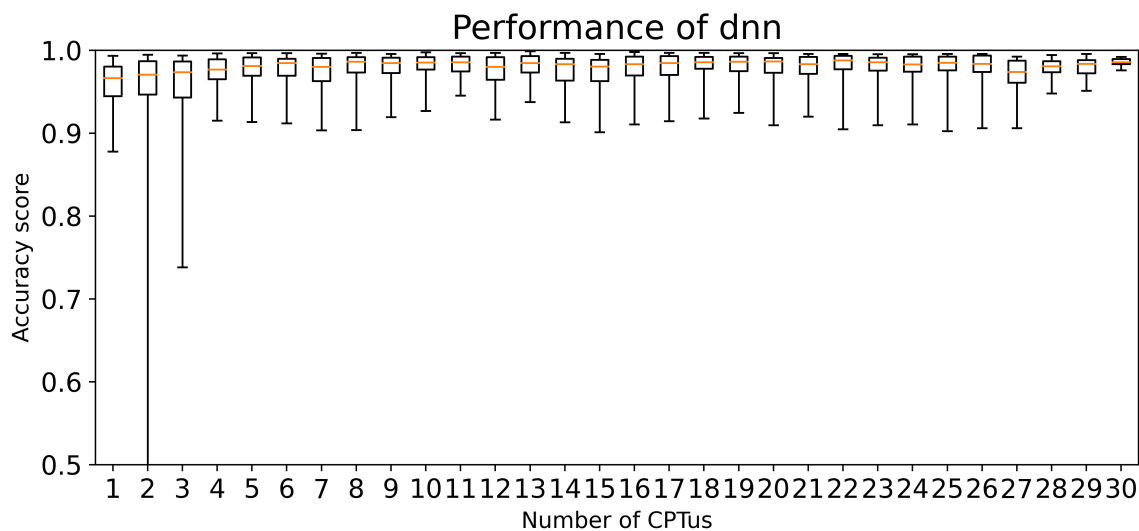


Figure 4.1: Performance of DNN depending on the number of CPTus from NGTS Tiller-Flotten it is trained on. The model uses normalized  $Q_t$ ,  $F_r$  and  $B_q$  for classification.

In figure 4.1 the performance of DNN can be seen converging after it is trained on four or more CPTus. The 5 % percentiles seem to hover around the 90 % accuracy mark while both the median and the 25 % percentiles end up above 95 % accuracy.

Figure 4.2 shows how DNN classifies the quick clay using the two parameters  $Q_t$  and  $F_r$  normalized. The training data consists of 5 CPTus, however all remaining points of the dataset is also plotted to illustrate how it performs as a whole. The shape of the decision barrier suggests that

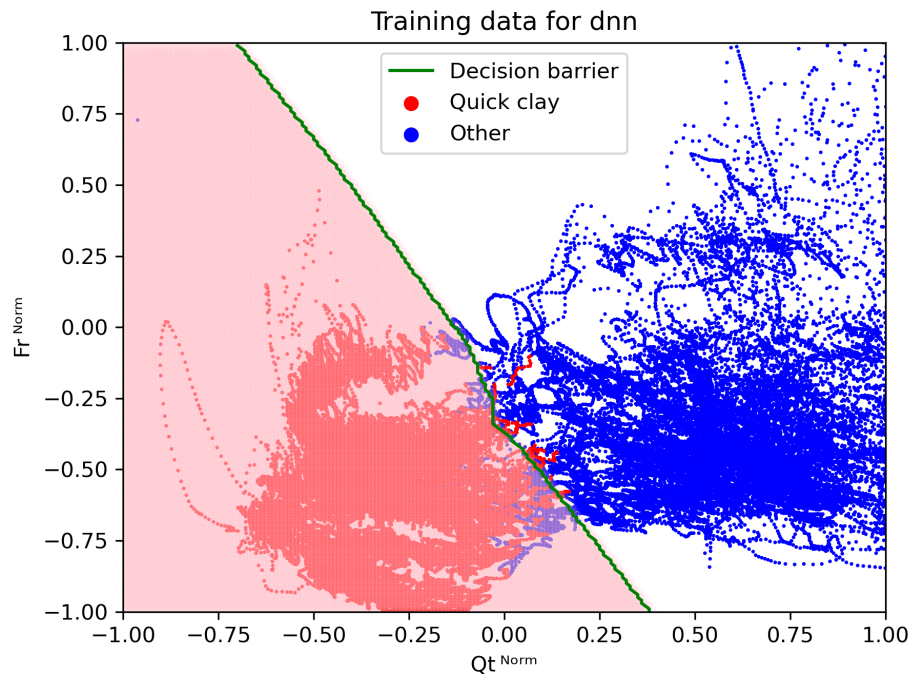


Figure 4.2: Normalized  $Q_t$ - $F_r$  plot of a DNN model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

the model is less likely to classify points as quick clay when  $F_r$  increases.

The plot in figure 4.3 illustrates how DNN encapsulates the quick clay zone in a three dimensional  $Q_t$ - $F_r$ - $B_q$  normalized space. The green shape marks where the model starts classifying points as quick clay. While it separates the dataset quite well, the direction of the left side of the barrier seem to indicate that it is less likely to classify points as quick clay when  $B_q$  increases and  $Q_t$  decreases. This would in turn not make for a good generalized model.

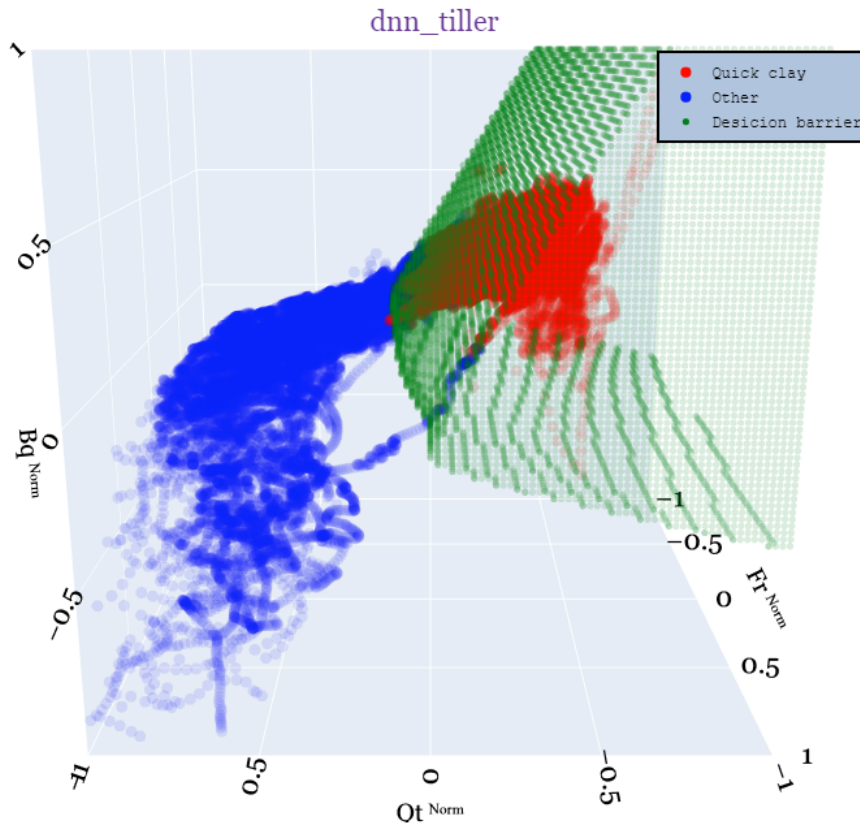


Figure 4.3: Normalized  $Q_t$ - $F_r$ - $B_q$  plot of a DNN model trained on 5 CPTus from NGTS Tiller-Flotten site. Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

### Convolutional neural network

When training the convolutional neural network (CNN) it was decided to use a single hidden layer structure. This was done in an attempt to answer whether it is more beneficial to increase the layer count as was done with DNN, or to alternatively include convolution filters to the network. 10 filters of length 1 were applied to the convolutional layer with the "Conv1D" function (Chollet et al., 2015), while 32 neurons were included in the hidden layer. The remaining parameters such as the activation functions, learning rate and early stoppage were set to the same as the DNN for a more direct comparison.

The number of CPTus vs accuracy score graph for the CNN model in figure 4.4 shows consistently good performance from the very beginning and that there's little to gain adding more than 3 CPTus. The IQR are almost constant throughout and the median hover around 97 % accuracy.

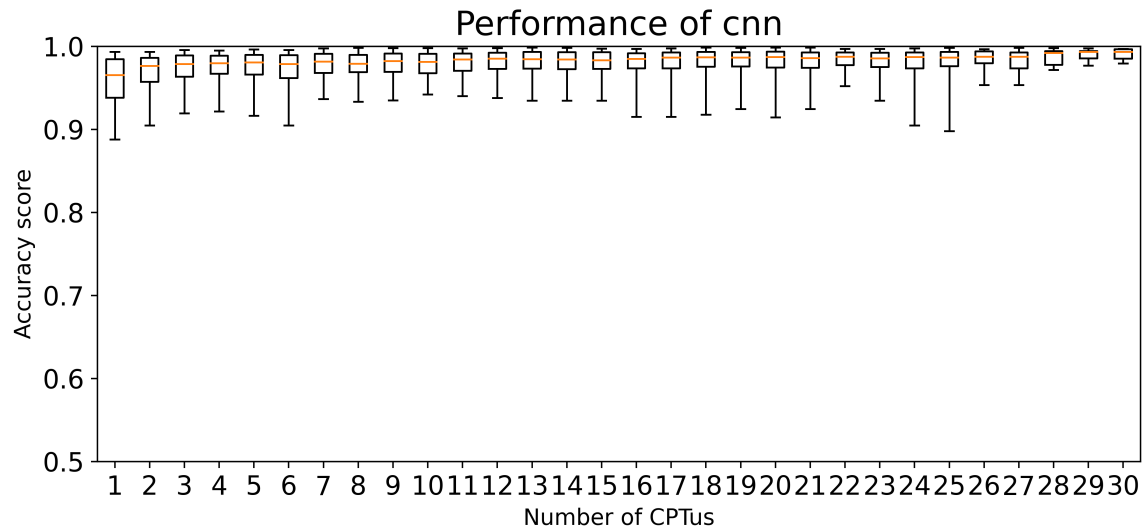


Figure 4.4: Performance of CNN depending on the number of CPTus from NGTS Tiller-Flotten it is trained on. The model uses all  $Q_t$ ,  $F_r$  and  $B_q$  normalized for classification.

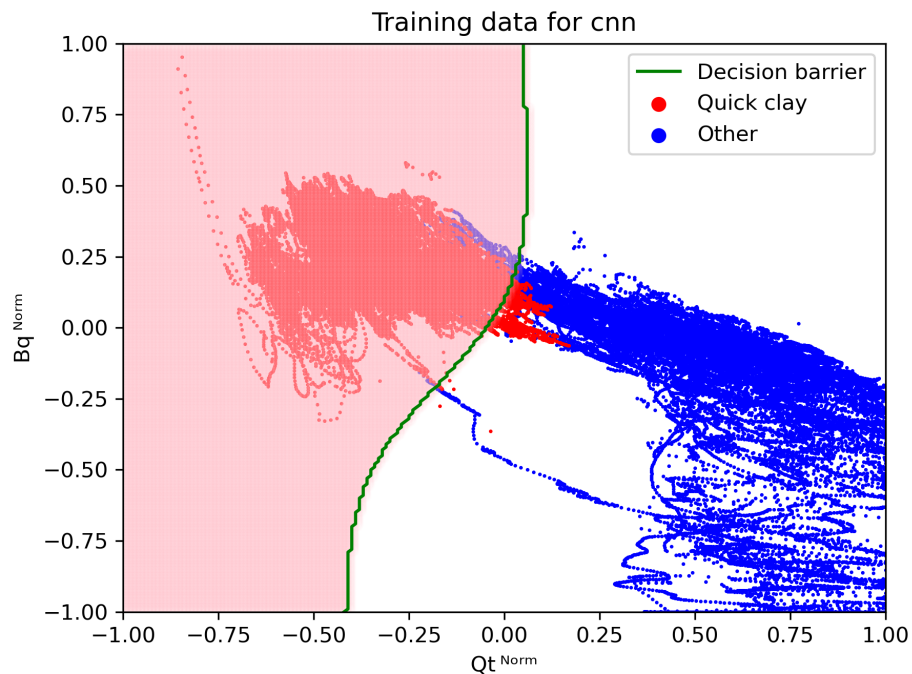


Figure 4.5: Normalized  $Q_t$ - $B_q$  plot of a CNN model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

The three dimensional plot of the CNN model can be seen in figure 4.6 and shows how the algorithms separate the two classes. The shape allows classification of quick clay for low values of  $B_q$ , however this requires low values for both  $Q_t$  and  $F_r$ . For higher values of  $B_q$  the model

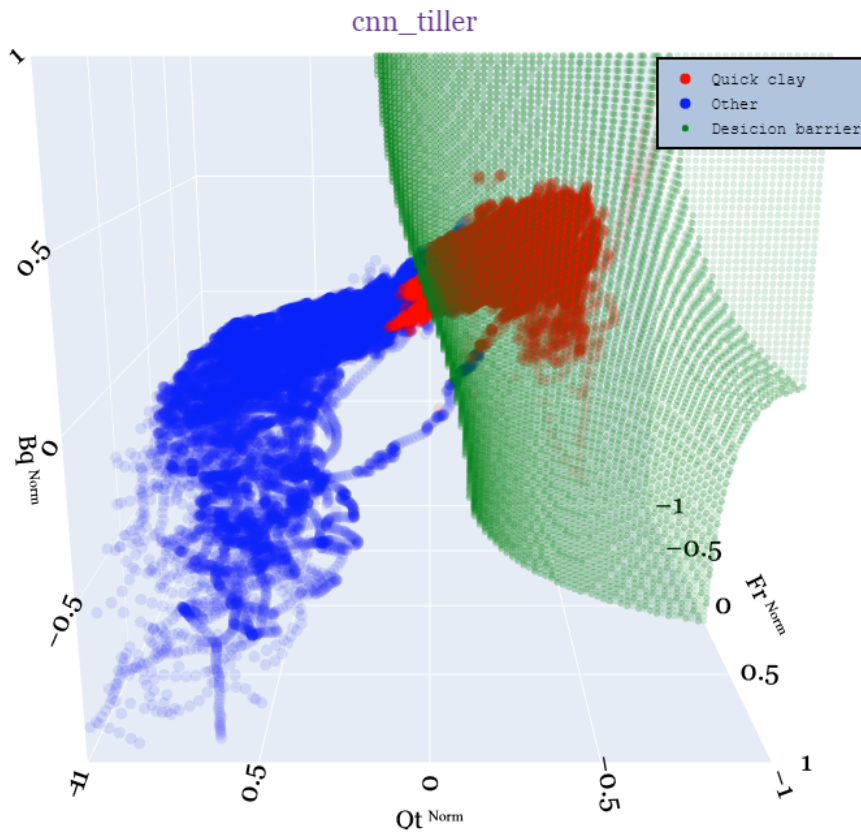


Figure 4.6: Normalized  $Q_t$ - $F_r$ - $B_q$  plot of a CNN model trained on 5 CPTus from NGTS Tiller-Flotten site. Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

is more likely to classify points as quick clay, as can be seen by the shape extending outwards at the top. From the plot some points are clearly misclassified, however this is an implication of the algorithm attempting to prevent overfitting. Notice that the horizontal section of the decision barrier in the top right occurs due to the plot range being -1 to 1. The overall impression CNN gives is that it has the generalizability of DNN, but that the filters gives it an additional perspective making it less likely to overfit the data.

## Extreme learning machine

Due to the amount of randomness involved when training models with the extreme learning machine (ELM), tuning of the model parameters was mostly dependent on a try and failure approach. The activation function used in the hidden layer was set to the radial basis function with the euclidean distance as kernel. A paper documenting the algorithm (Akusok et al., 2015), suggests using this function for tasks where the dependency between input parameters (features) and the output labels are complex. It was discovered that limiting the amount of neurons to 12 reduced the amount of noise in the solution. Since this dataset consists of points which are clustered in certain locations of 3D space, the shape of the decision barrier in many situations ended up being very senseless while still being able to separate the two classes.

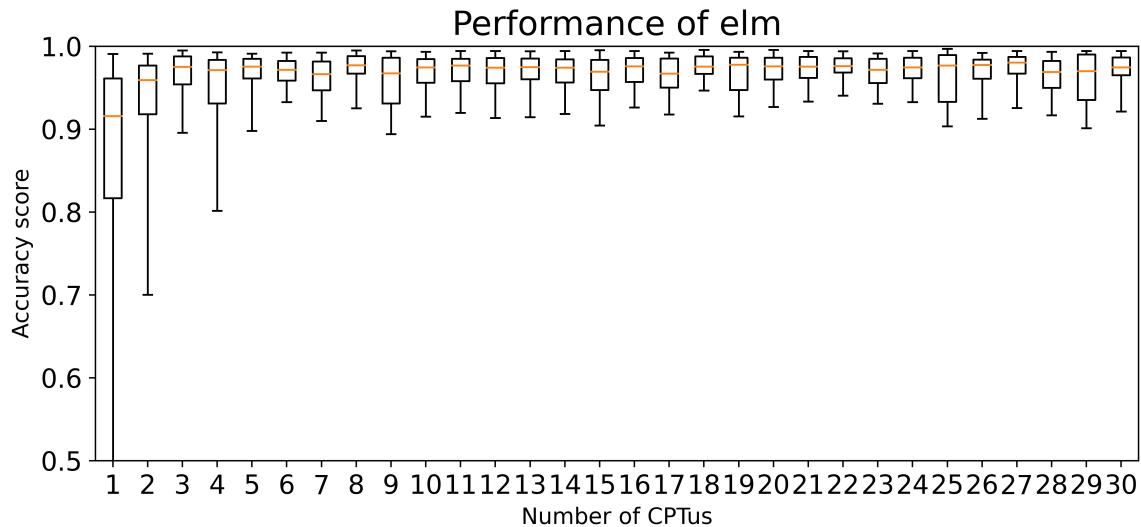


Figure 4.7: Performance of ELM depending on the number of CPTus from NGTS Tiller-Flotten it is trained on. The model uses normalized  $Q_t$ ,  $F_t$  and  $B_q$  for classification.

The performance when training and testing ELM on NGTS Tiller-Flotten is shown in figure 4.7. The accuracy starts converging somewhat after the models are trained with at least 5 CPTus, however there still some visible variance remaining. While the 5th percentile is above 90 % for the majority of the training, the 95th percentile fell below 99 % for the most of them as well.

Since the algorithm produced different solution for every run, several plots had to be considered as potential candidates to be presented. One such run with normalized  $Q_t$  and  $B_q$  as parameters resulted in the plot in figure 4.8. One notable feature of this model is that it creates a hard

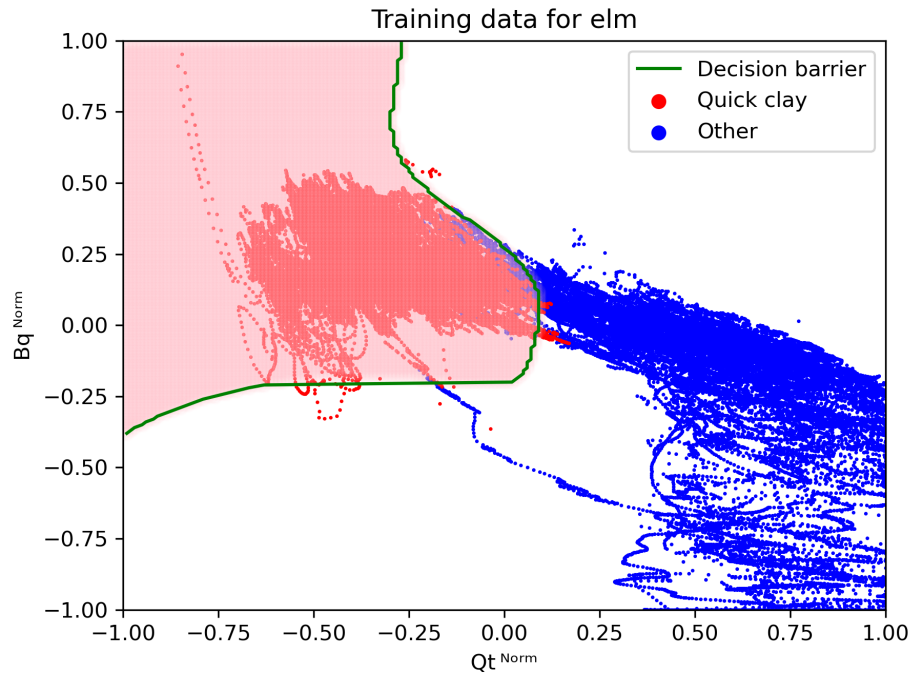


Figure 4.8: Normalized  $Q_t$ - $B_q$  plot of a ELM model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

cap (horizontal line) on quick clay around  $B_q^{norm} = -0.25$ . From the dataset it makes sense to put a line here, as one CPTu goes from other material to quick at this value. The overall plot suffers however from overfitting as the barrier above  $B_q^{norm} = 0.00$  turn inwards, meaning that a reduction in  $Q_t$  and an increase in  $B_q$  would lead to this area being less likely to be classified as quick clay.

In three dimension the algorithm creates an even bigger space for possible outcomes. Due to the low variance in this dataset the model isn't "penalized" for completely encapsulating either of the two classes which could lead to bad generalization. Since ELM is originally made to solve big and diverse datasets (Akusok et al., 2015), it is not expected that it would perform the best on the NGTS Tiller-Flotten dataset. Figure 4.9 shows how one ELM model looks in 3D. While it doesn't necessary overfit the data in this case, the shape is very complex making it hard to validate from a physical standpoint.

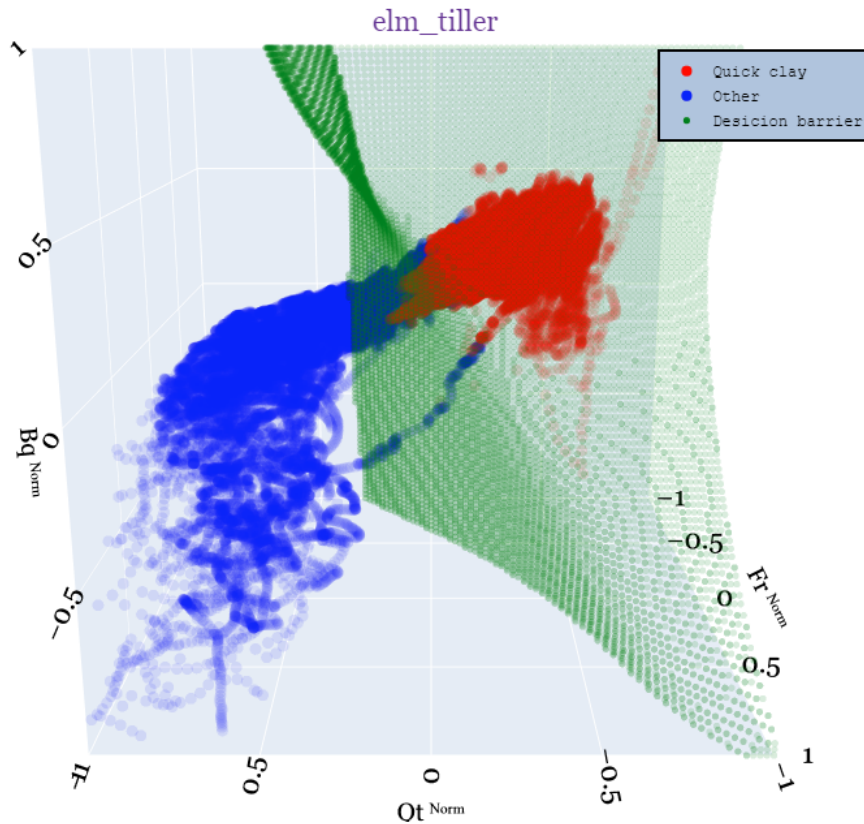


Figure 4.9: Normalized  $Q_t$ - $F_r$ - $B_q$  plot of a ELM model trained on 5 CPTus from NGTS Tiller-Flotten site. Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

## Random Forest

While many of the previously discussed algorithms required significant tuning of model parameters, the random forest (RF) classifier along with k-nearest neighbor and support vector machine from the sci-kit learn library already had their parameters prefilled with appropriate values. Since they performed well out of the box, time was spared researching how to optimize them.

The performance of the random forest seem to cap out after being trained on 3 CPTus as seen in figure 4.10. The variance is low throughout, however some inaccuracy is observed at the end.

When plotting how the algorithm divides the classes in figure 4.11, it becomes clear how the use of decision trees impact the shape of the decision barrier. While previous plots showed gradual transition between quick and other material, the use of decision trees makes for the



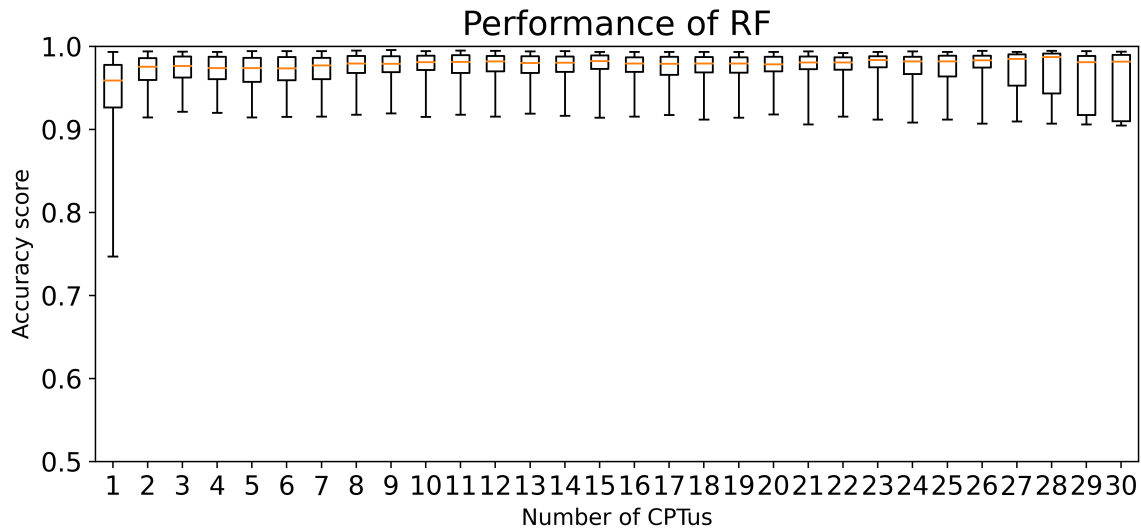


Figure 4.10: Performance of RF depending on the number of CPTus from NGTS Tiller-Flotten it is trained on. The model uses normalized  $Q_t$ ,  $F_r$  and  $B_q$  for classification.

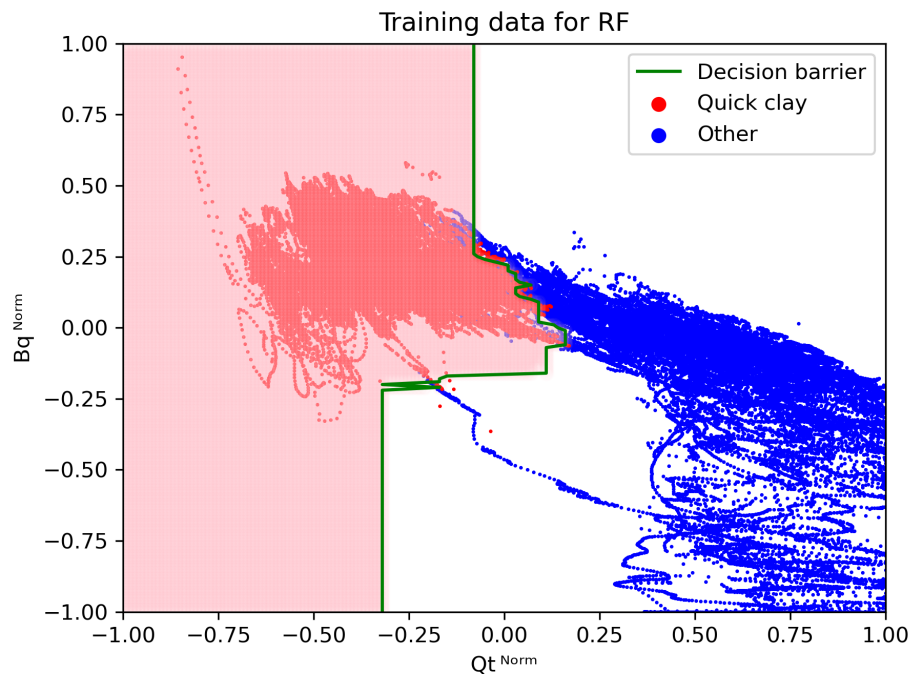


Figure 4.11: Normalized  $Q_t$ - $F_r$  plot of a RF model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

barrier consisting of straight lines. While this can be a limiting factor in accuracy, the psychical implication of the decision barrier is easy to interpret. The plot essentially tells us that it is more likely to classify quick clay as  $B_q$  increases, however the middle part of the plot

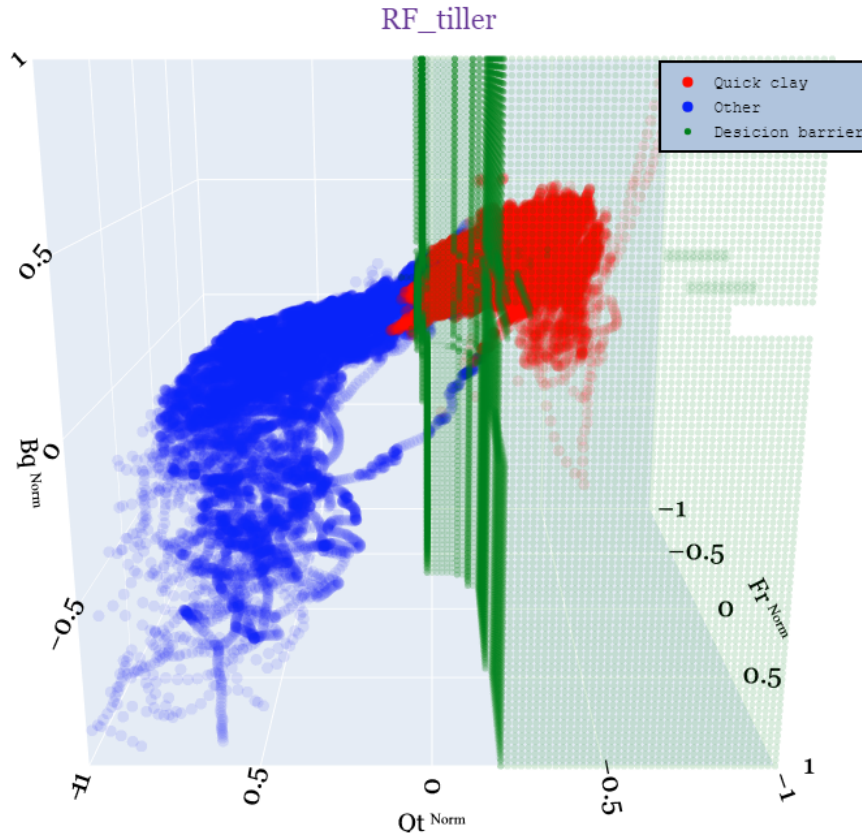


Figure 4.12: Normalized  $Q_t$ - $F_r$ - $B_q$  plot of a RF model trained on 5 CPTus from NGTS Tiller-Flotten. Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

overfitting the points to obtain higher accuracies for the dataset.

The three dimensional version of the algorithm in figure 4.12 contain the same characteristics as was seen in the 2D case with straight plane-like shapes. It can be observed that these planes are almost exclusively vertical meaning that model doesn't really make use of the  $B_q$  parameter when separating the classes. For higher values of  $F_r$  it is less likely to classify quick clay which can be observed by the barrier being shifted to the right here. It should however be noticed that the dependency on  $F_r$  is very minor compared to  $Q_t$ . The  $Q_t^{norm}$  range in which the barrier exists is around -0.2 to 0.0, indicating that the model overwhelmingly relies on this parameter.

## Extreme gradient boost

The Extreme gradient boost (XGB) algorithm turned out to be quite complex one to work with. Since it is built up of several optimization techniques and complex searching methods it consists of a lot of parameters that has to be tuned. To not get too overwhelmed a handful were considered while the remaining were left as default values. Since it is of interest to create models that can be generalized beyond the dataset it is trained on, model parameters that impact how conservative it classifies points were tuned. The values that were chosen are summarized in table 4.2. Detailed description of how they impact the training can be found in the toolbox documentation ([Chen and Guestrin, 2016b](#)).

Table 4.2: Model parameters used to train XGB.

Parameter	Values
Max search depth	6
Learning rate	0.3
Sub sample	0.8
Lambda	2.5
Minimum child weight	2
Number of boosting iterations	100

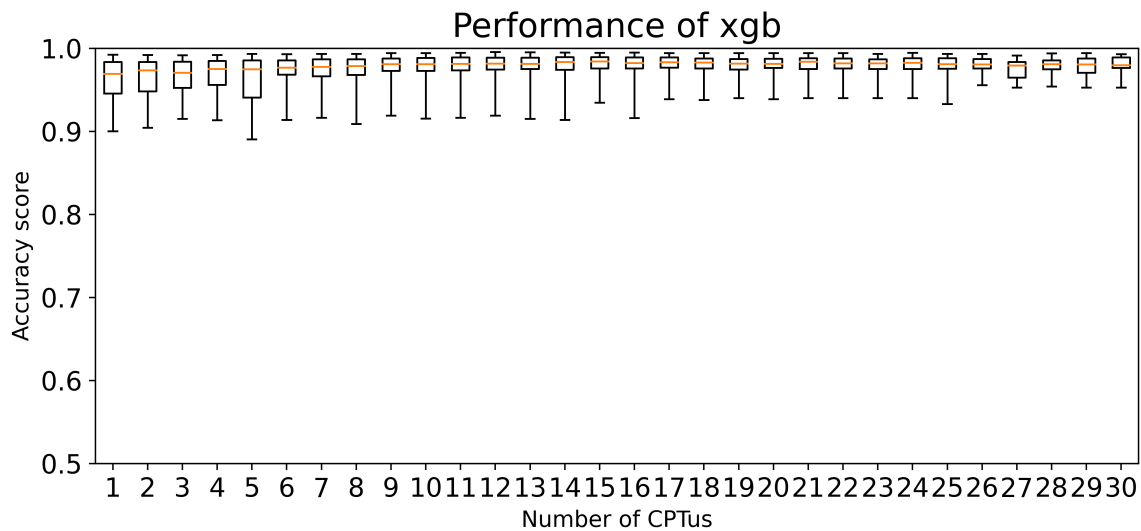


Figure 4.13: Performance of XGB depending on the number of CPTus from NGTS Tiller-Flotten it is trained on. The model uses normalized  $Q_t$ ,  $F_r$  and  $B_q$  for classification.

The performance plot in figure 4.13 shows that XGB is instantly able to predict the dataset accurately, with the 25th percentile starting at an accuracy of 95 %. Small increment in performance is observed until 6 models trained with 6 CPTus where it eventually caps out. The

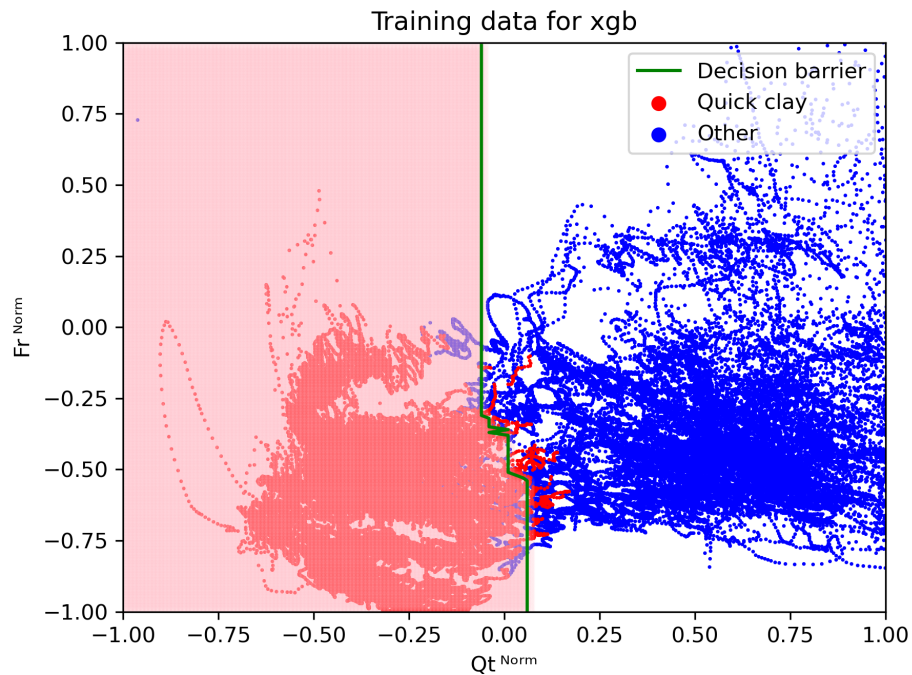


Figure 4.14: Normalized  $Q_t$ - $F_r$  plot of a XGB model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

small IQR indicate that it performs very equal on all the CPTus in the set.

The two dimensional plot in figure 4.14 shows that it barely uses  $F_r$  when classifying. This is clear when observing the range of  $Q_t^{norm}$  in which the decision barrier lays. For  $F_r^{norm} > -0.25$ ,  $Q_t$  needs only to be slightly lower to be classified as quick clay. While this technique is able to separate the training data very well, it is not done in a way that would not make sense for a general model. Changing model parameters to make the model more conservative didn't make a significant difference because of the algorithms underlying structure and the homogeneousness of the dataset.

The three dimensional case in figure 4.15 tells much of the same story with  $Q_t^{norm}$  being the parameter of most significance. The barrier is slightly pushing outwards for lower values of  $F_r$  and marginally for high  $B_q$ . Due to the fact that both XGB and RF are using decision trees they end up giving very similar results. It is clear that the algorithm is good at predicting the dataset it is trained on, however a more diverse dataset is required if the objective is make a model that can be generalized.

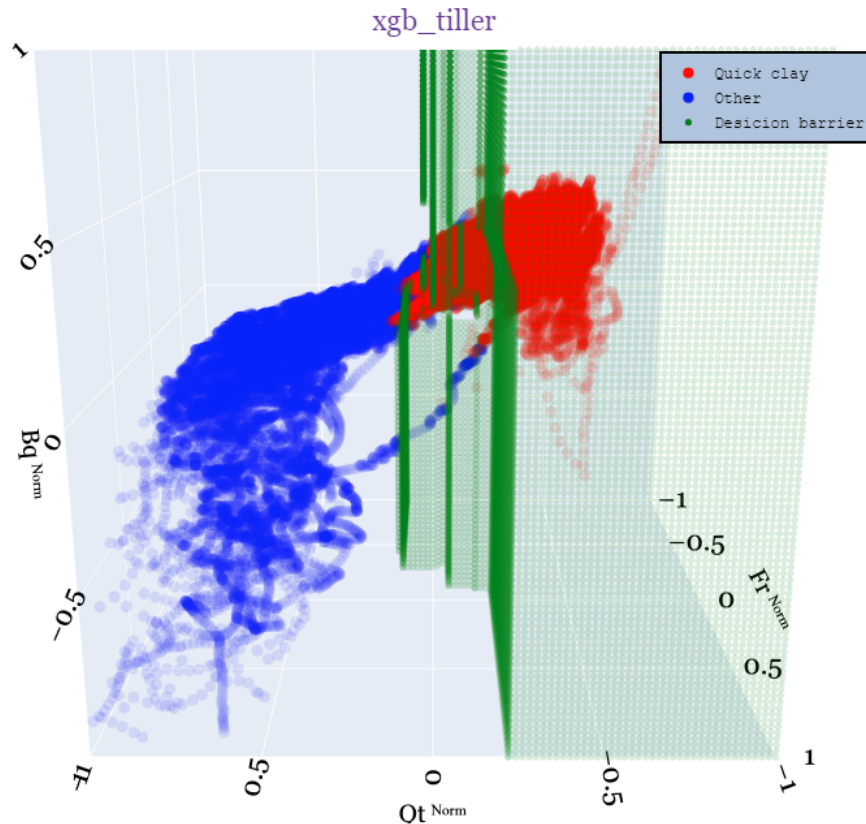


Figure 4.15: Normalized  $Q_t$ - $F_r$ - $B_q$  plot of a XGB model trained on 5 CPTus from NGTS Tiller-Flotten. Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

### K-nearest neighbor

The simple logic behind the k nearest neighbor (KNN) algorithm make for a good initial test case for the dataset. Realistically, only one model parameter is of significant importance, namely the k number of neighbors that should be considered. It might sometimes also be of interest to use a distance weighted sum instead of the highest number of points when deciding the class, however it was found to not make a significant impact on the dataset so this was not applied. Choosing a fixed value for k is often used as a starting point, though iterating through different values could also be an option. The process of iterating could lead to overfitting (Seidl et al., 2009), so it was ultimately decided to set the  $k = \sqrt{N_{samples}}$

The dependency plot in figure 4.16 shows a very constant accuracy during the training. Due to the algorithm only considering what the nearest neighbors classes are, it is expected that it

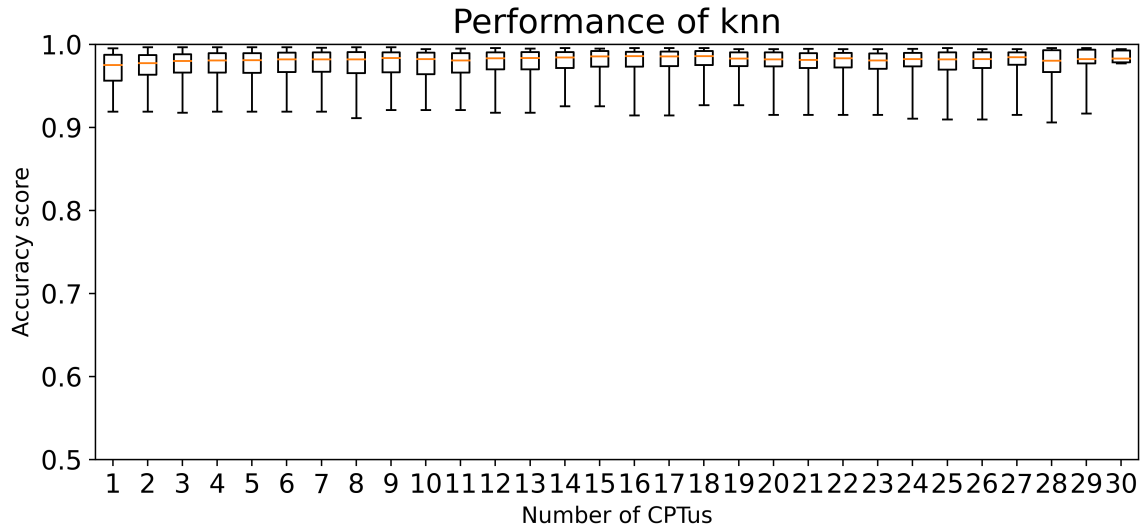


Figure 4.16: Performance of KNN depending on the number of CPTus from NGTS Tiller-Flotten it is trained on. The model uses normalized  $Q_t$ ,  $F_r$  and  $B_q$  for classification.

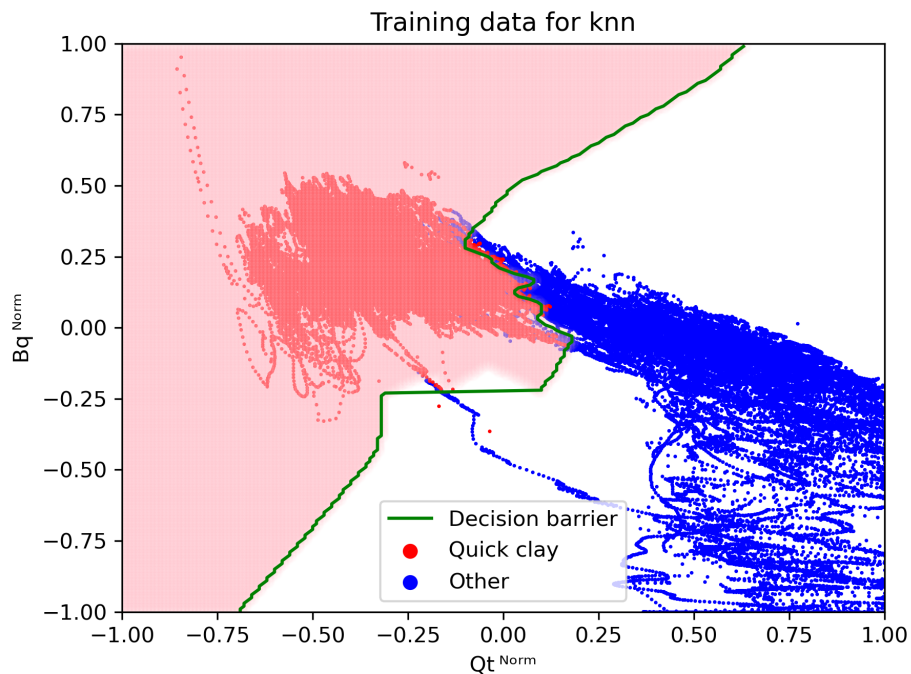


Figure 4.17: Normalized  $Q_t$ - $B_q$  plot of a KNN model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

should be good at predicting a homogeneous dataset. Since the increased amount of CPTus during training doesn't better the performance significantly, more than 3 CPTus to train with would only marginally increase accuracy.

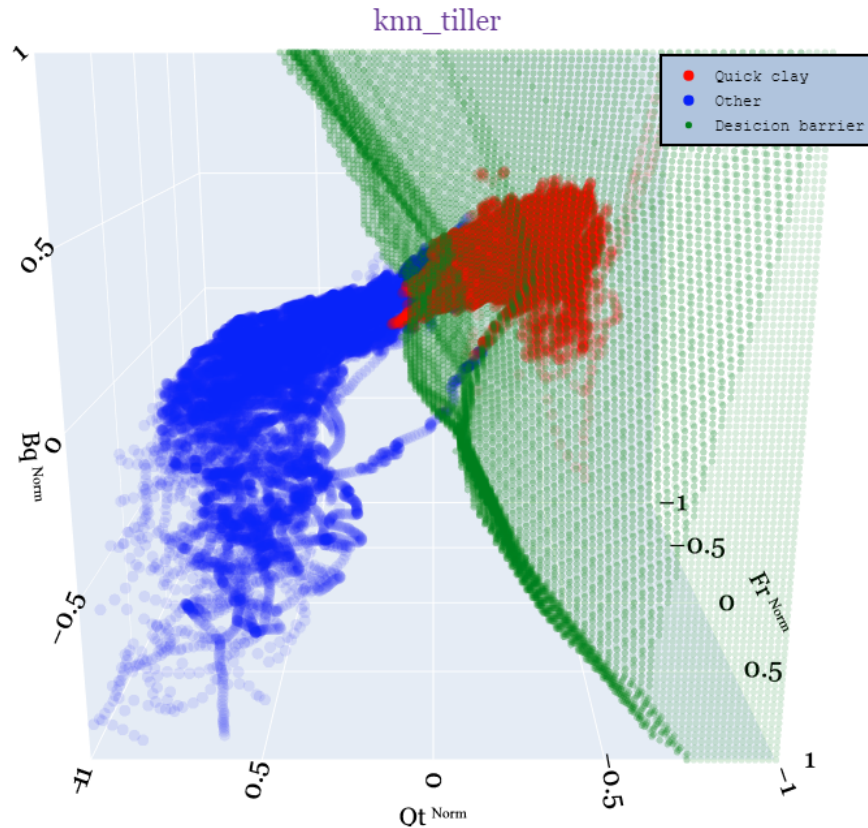


Figure 4.18: Normalized  $Q_t$ - $F_t$ - $B_q$  plot of a KNN model trained on 5 CPTus from NGTS Tiller-Flotten. Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

Looking at the 2D plot of the KNN algorithm in figure 4.17, an almost perfect separation of the classes can be seen, considering the dataset at hand. The shape of the barrier indicate that it is more likely to classify quick clay as  $B_q$  increases. In the middle however the points in the dataset forces the barrier to follow them, resulting in overfitting.

The fact that points are classified dependent only on their nearest neighbors, it results in a barrier that are very complex in figure 4.18. The shape is very irregular and difficult to interpret from a physical standpoint. Avoiding overfitting with KNN might be difficult, however it seems to reflect the importance of  $B_q$  when classifying quick clay. This can be seen by the barrier expanding outwards for higher values of  $B_q$ .

## Support vector machine

For the implementation of the support vector machines (SVM), some tuning of model parameters are generally possible. However, initial testing with the algorithm showed promising results, and hence no further tuning of parameters were performed. This means that the default values as described by the sci-kit learn library (Pedregosa et al., 2011) were used.

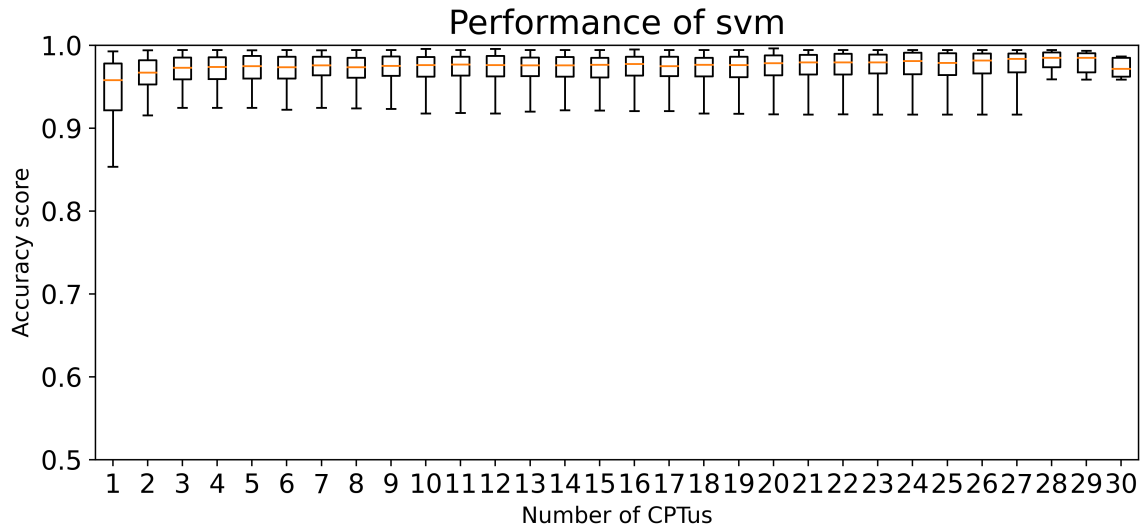


Figure 4.19: Performance of SVM depending on the number of CPTus from NGTS Tiller-Flotten it is trained on. The model uses normalized  $Q_t$ ,  $F_r$  and  $B_q$  for classification.

The performance plot in figure 4.19 shows much of the same picture as with KNN, where it caps out very quickly without much improvement after adding any more CPTus. The mean accuracy is above 96% from the first CPTu trained and tested.

In the 2D plot in figure 4.20, the algorithm draws a straight line to separate the classes. While this result in more misclassifications of points in the dataset, it doesn't show the same overfitting tendency some of the other models do. The shape of the decision barrier suggests that it is less likely to classify quick clay as  $F_r$  increases.

While the 2D plot draws a straight line to separate the classes, the 3D plot, as can be seen in figure 4.21, is represented by a plane surface. The shape of the plane suggest that it is more likely to classify quick clay as  $B_q$  increases, while  $F_r$  and  $Q_t$  decreases, which all makes sense from a physical standpoint. As with the 2D case, the algorithm offers less accuracy when tested on the dataset it is trained in exchange for a more generalizable model. While SVM suggests



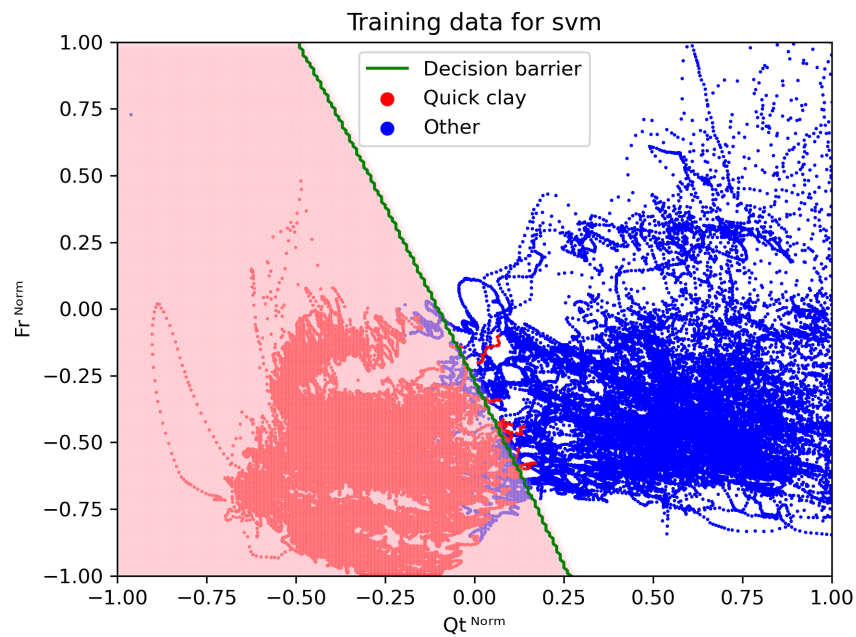


Figure 4.20: Normalized  $Q_t$ - $F_r$  plot of a SVM model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

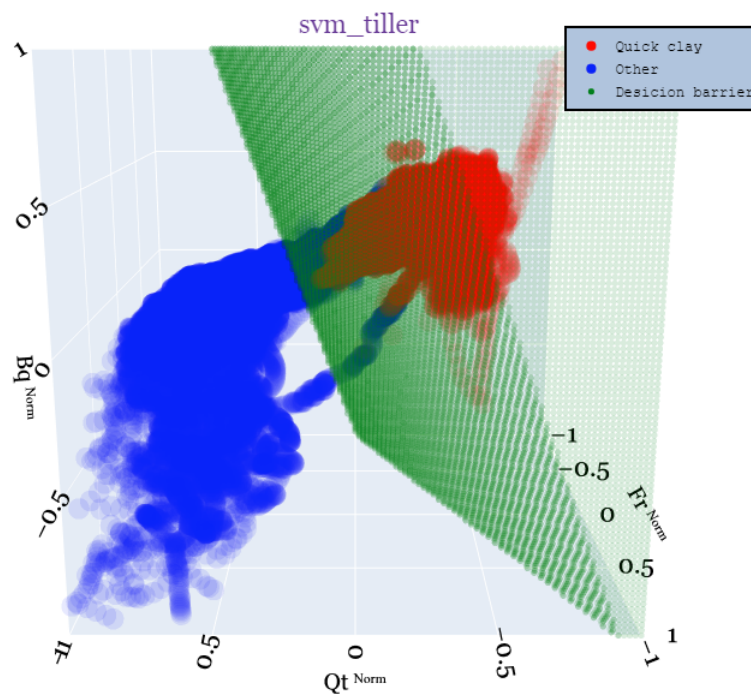


Figure 4.21: Normalized  $Q_t$ - $F_r$ - $B_q$  plot of a SVM model trained on 5 CPTus from NGTS Tiller-Flotten. Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

one plane based on least square error terms, it would be possible to experiment with different planes as it is easily described mathematically. This way it could be fine tuned to for example reduce the amount of false positives it produces. Note that even though the algorithm ended up with a plane separating the classes for this dataset, for other datasets, the radial basis kernel will likely find more optimal margins when transforming the parameters into higher dimensions ([Aizerman, 1964](#)).

### Comparison of models on Tiller-Flotten dataset

The performance of each CPTu based on the chosen models are shown below in table 4.3.

Table 4.3: Table showing how each algorithm trained on 5 CPTus from Tiller perform on each individual CPTu. Models are trained using all three features  $Q_t-F_r-B_q$  normalized.

CPTu number	ELM	XGB	RF	SVM	KNN	DNN	CNN
1	98 %	98 %	98 %	99 %	98 %	99 %	98 %
2	92 %	91 %	91 %	93 %	93 %	87 %	93 %
3	96 %	99 %	99 %	98 %	99 %	100 %	100 %
4	96 %	98 %	97 %	96 %	99 %	98 %	97 %
5	91 %	99 %	99 %	96 %	100 %	99 %	99 %
6	95 %	98 %	97 %	96 %	99 %	99 %	99 %
7	96 %	95 %	99 %	97 %	99 %	96 %	96 %
8	100 %	99 %	99 %	99 %	99 %	100 %	100 %
9	97 %	98 %	98 %	96 %	98 %	99 %	98 %
10	96 %	99 %	99 %	97 %	99 %	98 %	99 %
11	99 %	98 %	98 %	97 %	98 %	98 %	98 %
12	98 %	96 %	96 %	95 %	96 %	98 %	97 %
13	99 %	99 %	99 %	98 %	100 %	98 %	99 %
14	99 %	99 %	99 %	96 %	99 %	99 %	100 %
15	93 %	98 %	97 %	95 %	99 %	97 %	98 %
16	98 %	100 %	100 %	97 %	99 %	100 %	100 %
17	95 %	97 %	97 %	99 %	94 %	99 %	98 %
19	91 %	99 %	99 %	98 %	97 %	100 %	99 %
20	99 %	99 %	100 %	98 %	99 %	100 %	99 %
22	90 %	93 %	93 %	90 %	94 %	93 %	95 %
23	99 %	94 %	96 %	99 %	96 %	94 %	97 %
24	99 %	94 %	94 %	99 %	94 %	94 %	96 %
25	97 %	97 %	98 %	97 %	96 %	97 %	97 %
26	96 %	99 %	99 %	96 %	99 %	99 %	99 %
27	95 %	98 %	96 %	96 %	98 %	99 %	97 %
28	96 %	98 %	98 %	97 %	99 %	97 %	99 %
29	100 %	99 %	96 %	99 %	96 %	99 %	99 %
30	100 %	99 %	99 %	99 %	99 %	100 %	100 %
31	90 %	91 %	91 %	89 %	91 %	90 %	91 %
32	99 %	98 %	98 %	97 %	98 %	98 %	99 %
33	98 %	98 %	97 %	98 %	96 %	97 %	96 %
34	99 %	99 %	98 %	99 %	99 %	99 %	98 %
Median	97.0 %	98.2 %	97.9 %	96.9 %	98.3 %	98.5 %	98.3 %
Mean error	3.6 %	2.7 %	2.7 %	3.3 %	2.7 %	2.5 %	2.2 %

From this analysis it can be observed that CNN has the most consistency with only a mean error of 2.2 %. ELM performed overall worst with the lowest median accuracy and highest mean error.

DNN had the highest median accuracy, however looking at the decimals only 0.2 % separated CNN and DNN indicating that CNN slightly outperformed DNN overall. While some performed better, all algorithms managed to clearly capture the separation of the two classes. The high performance obtained here is mainly due to the homogeneity of the dataset having clear grouping of the two classes as can be seen by the 3D plots shown earlier in this section.

A similar analysis was done where the models only used two parameters in the training. While all three combinations of two parameters were tested, only models trained with  $Q_t-F_r$  and  $Q_t-B_q$  (normalized) will be presented as they performed significantly better than  $F_r-B_q$ . The median and mean error for the two parameter sets are summarized in table 4.4. The overall performance of the models trained with two parameters tend to be somewhat worse, as is similar to the results found by Valsson (2019). ELM and SVM however, showed best accuracy when trained with normalized  $Q_t$  and  $F_r$ .

Table 4.4: Performance of the algorithms trained on 5 CPTus from NGTS Tiller-Flotten and tested on each individual CPTu. The top table shows median and mean error performance of models trained with normalized  $Q_t$  and  $F_r$  as features while the bottom are trained with normalized  $Q_t$  and  $B_q$  as features.

$Q_t-F_r$	ELM	XGB	RF	SVM	KNN	DNN	CNN
Median	97.5 %	96.7 %	97.2 %	97.5 %	97.8 %	97.1 %	98.0 %
Mean error	3.5 %	4.9 %	4.4 %	3.2 %	3.2 %	4.0 %	2.7 %
$Q_t-B_q$	ELM	XGB	RF	SVM	KNN	DNN	CNN
Median	97.1 %	98.2 %	98.3 %	97.0 %	98.4 %	98.8 %	98.2 %
Mean error	3.7 %	2.7 %	2.6 %	3.7 %	2.5 %	2.2 %	2.7 %

While the performance seem to be better when increasing the amount of features, getting a good intuition of how the algorithms separates the classes become more difficult. Visualizing a two parameter model requires only a single 2 dimensional chart. When dealing with models using three parameters, an interactive 3D plot is needed to get a full perspective for how it classifies. A snapshot of the 3D models as shown in this section should at least illustrate that the different algorithms obtain very different decision boundaries, see for example figure 4.18 and 4.21.

## 4.2 ML models trained on dataset II and tested on dataset I

From the previous section it is clear that similarities of the training and testing datasets impacts the performance heavily. It would therefore be of interest to see how a secondary dataset would perform on the CPTus from NGTS Tiller-Flotten. Since dataset I is not part of this dataset it became a good candidate for testing the models. The characteristics of the sites in dataset II are summarized in 3.2.

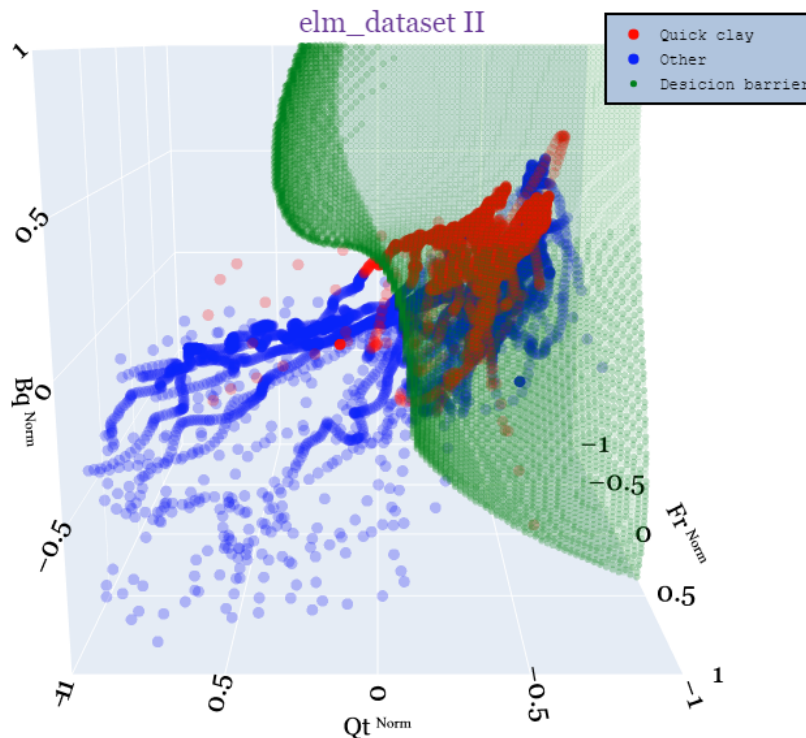


Figure 4.22: 3D plot of ELM trained on dataset II (full). Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

Figure 4.22 illustrates the dataset in a normalized  $Q_t$ ,  $F_r$  and  $B_q$  plot. As can be seen in this figure, this dataset is more complicated making it more difficult to separate the two classes. When training the machine learning algorithms on this dataset, two different approaches were considered. Approach 1 involved training models with the whole dataset where the algorithms have to deal with points that might be misclassified. Approach 2 consist of only including points every whole meter of the CPTu recordings, resulting in dataset with much less points. The second approach makes it very easy to filter out bad behaving points, however the amount of remaining

points become very slim.

When testing the models with the two different approaches, three of the algorithms performed best with the complete dataset, while the rest performed better with the reduced dataset. Table 4.5 summarizes the median and mean error for the two approaches. In both approaches SVM is outperforming the others, showing it's strength in terms of generalization. XGB and RF, which both uses decision trees, fail to separate the quick clay behaviour, showing the worst performance overall when the training and testing data is from different sites. ELM shows quite good results in both approaches and doesn't seem to be negatively affected by the inclusion of bad behaving points. CNN performs good in the second approach and decent in the first, outperforming KNN and DNN which both seem to suffer from overfitting.

Table 4.5: Performance of the algorithms for the two different approaches trained on dataset II with normalized  $Q_t$ ,  $F_r$  and  $B_q$  as parameters. The models are tested on dataset I

Approach 1	ELM	XGB	RF	SVM	KNN	DNN	CNN
Median	94 %	71 %	85 %	95 %	88 %	86 %	91 %
Mean error	6.5 %	26.5 %	17.3 %	6.2 %	15.4 %	13.9 %	9.7 %
Approach 2	ELM	XGB	RF	SVM	KNN	DNN	CNN
Median	93 %	51 %	73 %	97 %	95 %	94 %	94 %
Mean error	8.0 %	35.4 %	26.9 %	4.8 %	9.8 %	8.9 %	6.4 %

Looking at figure 4.22 it shows how the ELM model is able to create a smooth transition between the classes despite the data being quite messy. Even though the model is clearly misclassifying quite a lot of points, it can be seen as an advantage as it doesn't suffer from overfitting. KNN and DNN on the other hand will try to fit the training data as best they can, creating very complicated decision borders, and as a result become less generalizable. The best performing model is shown in figure 4.23, and shows how SVM separates the reduced dataset like a bowl shape. The physical implication of it's shape is quite intuitive as it shows a tendency to classify a point as quick as  $B_q$  increases and  $Q_t$  and  $F_r$  decreases. Figure 4.24 shows how the result of training the SVM model on all the points in the dataset. The shape in this case is quite different and complicated when compared to the model trained on the reduced dataset. It seems to classify points as quick clay even for higher values of  $Q_t$  which could lead to a high false positive rate in certain cases.

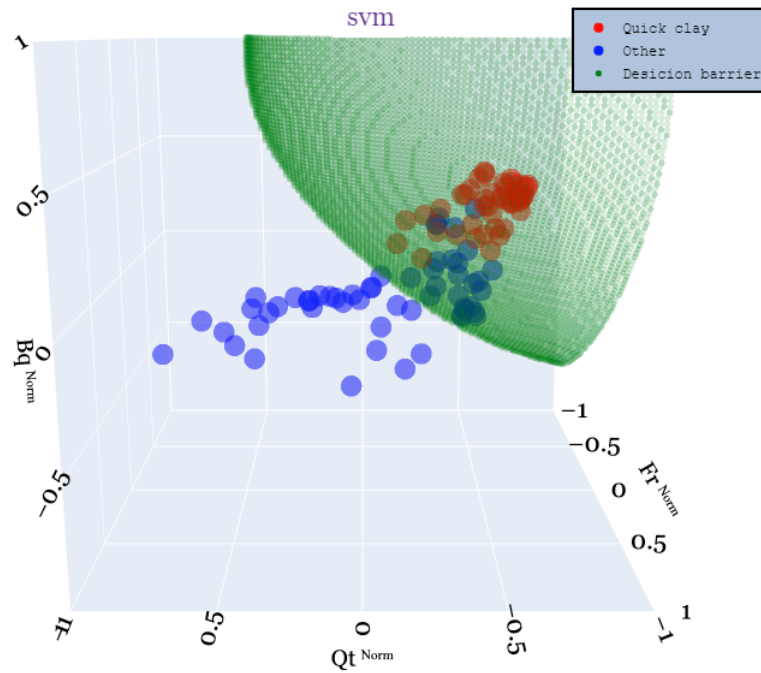


Figure 4.23: 3D plot of SVM trained on dataset II (reduced). Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

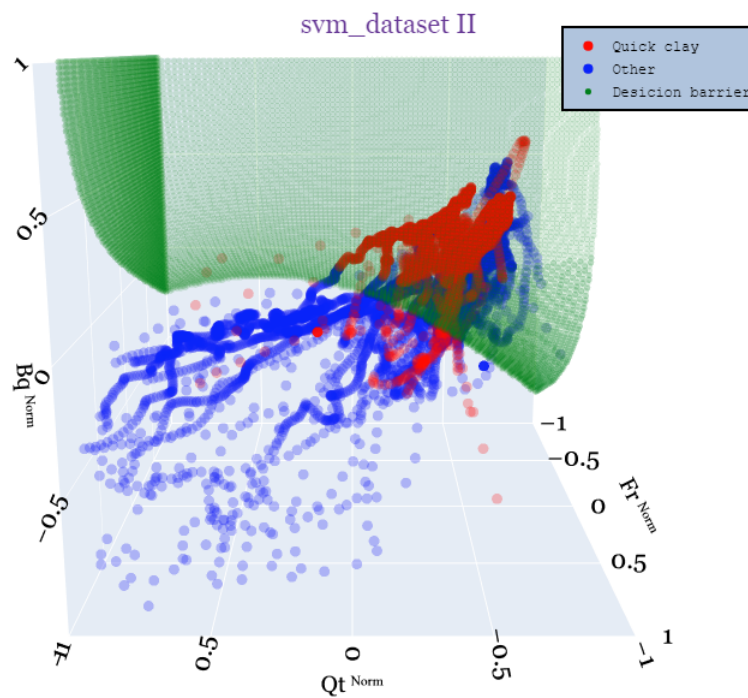


Figure 4.24: 3D plot of SVM trained on dataset II (full). Blue points are training data labeled as other material, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

To understand why XGB and RF are performing bad when the test and training datasets are different, plots of their decision barriers and a testing CPTu are useful, see figure 4.25. While the algorithm certainly separates the training dataset well, the decision boundary doesn't make for a good general model. The shape consists of sharp edges while the actual physical problem would suggest that a more gradual transition between the classes would more likely be the case. In the plot, CPTu number 32 from NGTS Tiller-Flotten is scattered in pink and green, and as it is presented all off the points end up outside the quick clay zone that the algorithm suggest. This means that the model predicts that none of these points are quick clay resulting in a true positive score of 0 % for this CPTu.

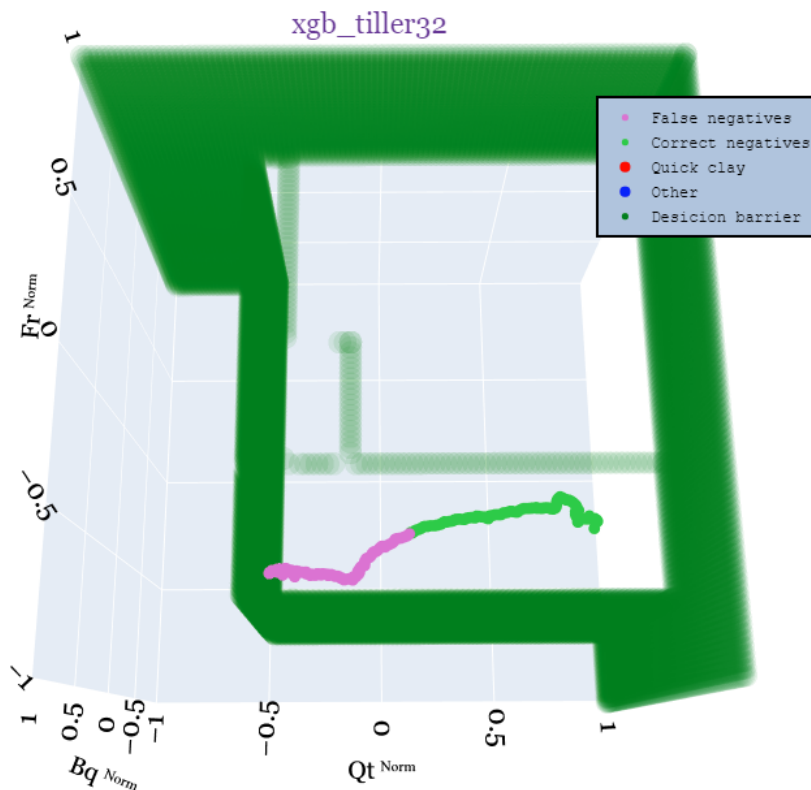


Figure 4.25: 3D plot of XGB trained on dataset II (reduced). CPTu 32 from Tiller is shown in pink and green points.

Comparison of the different algorithms versus depth are given in figure 4.26. This shows how the models trained on the reduced dataset (approach 2) classify each depth for a set of CPTus from NGTS Tiller-Flotten. These graphs give a different view on how some models persistently outperform others, while some models misclassify large parts of the profile in some cases. In this particular set CNN and SVM are classifying almost identically and are performing best in



almost all the profiles. ELM seem to do a solid job classifying the quick clay below 7.5 meters however it seem to have a tendency to misclassify the very top. The reason for this stems from the fact that the randomly generated neurons can get activated for points that are outside the -1 to 1 range. This is a problem that wasn't noticed before observing plots of the ELM model which had extended the plot range. As this fact was only observed for ELM, one proposed solution is to make sure that points outside the -1 to 1 range never should be classified as quick clay.

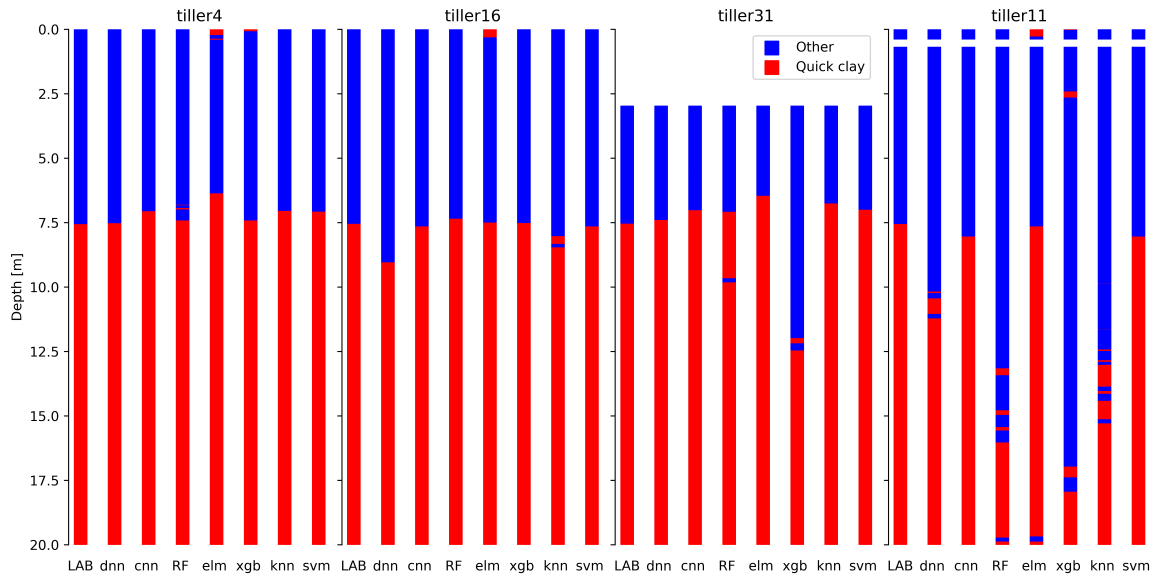


Figure 4.26: Classification profiles of a set of CPTus from Tiller. The site layering is given in the column label LAB. Models are trained on the reduced version of dataset II.

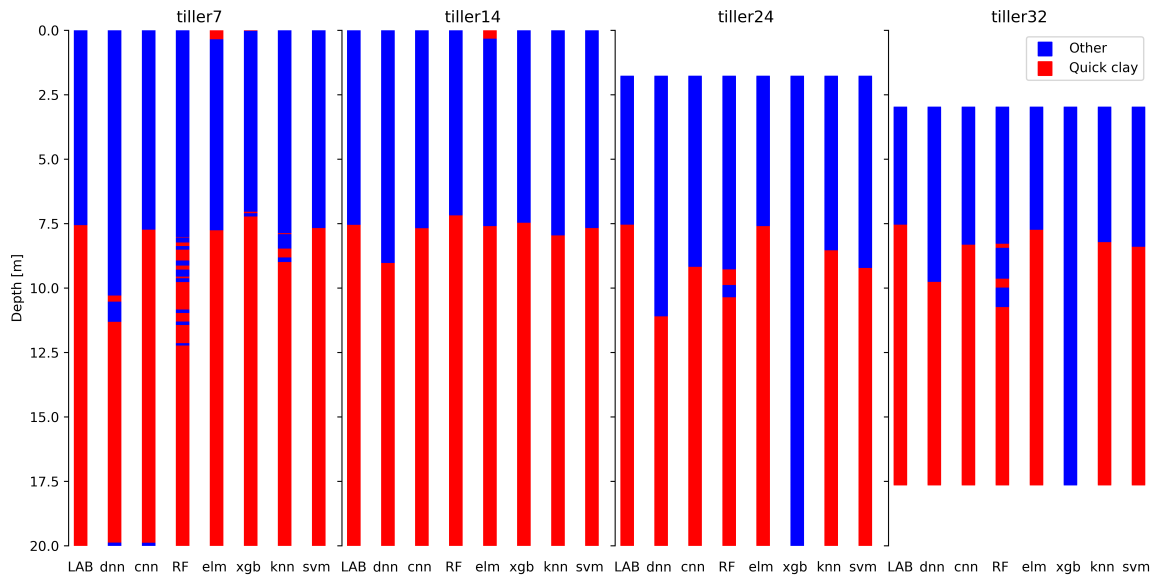


Figure 4.27: Classification profiles of a set of CPTus from Tiller. The site layering is given in the column label LAB. Models are trained on the reduced version of dataset II.

When training the models on a different dataset it is clear that the accuracy shown found in section 4.1 aren't reproducible, however the SVM algorithm come very close with a median up to 97 %. RF and XGB, both based on decision trees, perform the worst when the testing and trained set are different. While some algorithms prefer approach 1, the best performing models are found using approach 2. In addition to performing better, approach 2 also result in less complicated decision barriers, making it easier to understand why it does well or not. While it happens to be that the filtered dataset perform overall better in this case, it is not necessarily a general result.

### 4.3 Validation of trained models on a third dataset

This section will focus on validation of the machine learning models. The models can be validated by training them on one dataset and testing them on an unaffiliated dataset. Dataset II is basis for training the models, while the models are tested on dataset III. The difference between testing the models on this dataset vs NGTS Tiller-Flotten is that the Saksvik site has more heterogeneous layering. It is of interest to see whether the machine learning models are able to predict quick clay behavior in such a case.

Since the machine learning models trained on dataset II perform overall better with the second approach, these will be used when testing the Saksvik set. The classification profiles for the CPTus in the dataset are shown in figures 4.28 and 4.29.

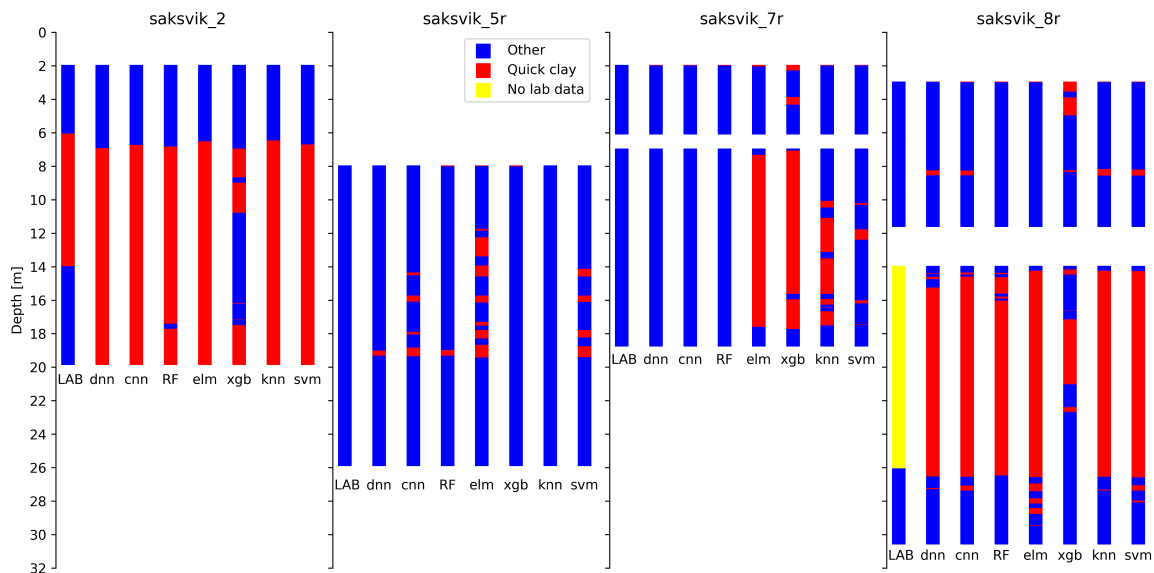


Figure 4.28: Classification profiles for CPTus at Saksvik. The site layering is given in the column label LAB. Depths marked with yellow have unknown layering. Models are trained on the reduced version of dataset II.

Due to a lack of laboratory testing for certain depths, parts of the profiles are marked as unknown in yellow. The fact that only one of the CPTus profiles have proven quick clay means that the likelihood of false positive that the models make are being tested. Giving accuracy scores for each model becomes difficult where the layering are unknown, therefore it is given as a range in table 4.6. The accuracy were calculated both as if the unknown layers were completely quick clay or if they were completely other material. Note that the XGB model perform better if the unknown layers are non-quick like at CPTu 9, however the rest obtain highest accuracy if the

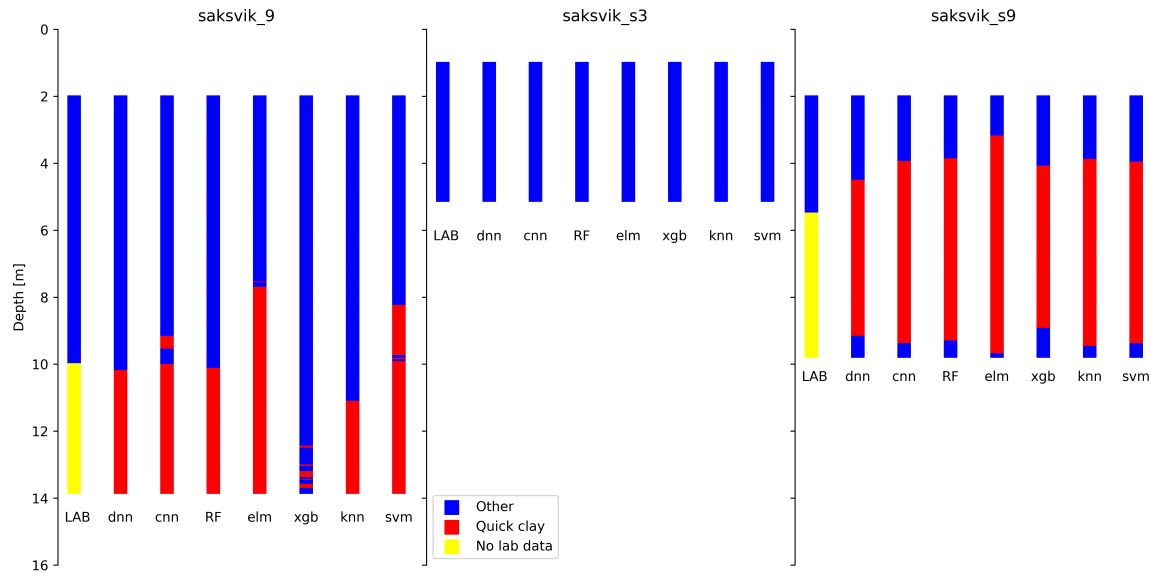


Figure 4.29: Classification profiles for CPTus at Saksvik. The site layering is given in the column label LAB. Depths marked with yellow have unknown layering. Models are trained on the reduced version of dataset II.

Table 4.6: Performance of the models trained on dataset II (reduced) and tested on dataset III. The accuracy are given as a range where the soil condition are unknown.

CPTu number	ELM	XGB	RF	SVM	KNN	DNN	CNN
2	65 %	61 %	64 %	64 %	65 %	63 %	64 %
5R	74 %	100 %	97 %	87 %	99 %	98 %	91 %
7R	34 %	28 %	100 %	91 %	59 %	100 %	99 %
S3	100 %	100 %	100 %	100 %	100 %	100 %	100 %
8R	44 - 90 %	64 - 72 %	52 - 96 %	45 - 91 %	48 - 94 %	51 - 92 %	48 - 93 %
9	47 - 80 %	73 - 95 %	68 - 99 %	53 - 85 %	77 - 91 %	69 - 99 %	64 - 96 %
S9	16 - 69 %	37 - 71 %	30 - 73 %	30 - 75 %	28 - 75 %	39 - 78 %	30 - 75 %

unknown layers are determined to be quick clay.

From figures 4.28 and 4.29 it is clear that SVM, CNN and DNN seem to classify the profiles quite similarly. However there is a pattern that suggest that SVM is more likely to classify quick clay than CNN and that CNN is more likely to classify quick clay than DNN. While this should not make a huge difference in a general case, it makes DNN the highest scoring algorithm for this dataset. Both ELM and XGB seem to have a hard time predicting correctly due to the low friction registered on some of the profiles, however assuming that CPTu 8R and 9 don't have quick clay, XGB would be outperforming the other algorithms on these profiles.

Laboratory data from CPTu profile 2 shows quick clay with  $c_{u,r}$  about 0.1 kPa at around 7 meters depth, which all the machine learning models are able to detect. At 14 meters depth however

the lab shows that the clay is no longer quick, while the models suggests that this is not the case. No more lab data is available below 15 meters so it is only assumed that the rest of the profile is not quick clay. The total sounding shown in figure 4.30 give no clearer answer to what the soil layering is as the response is very homogeneous between 5 and 20 meters depth.

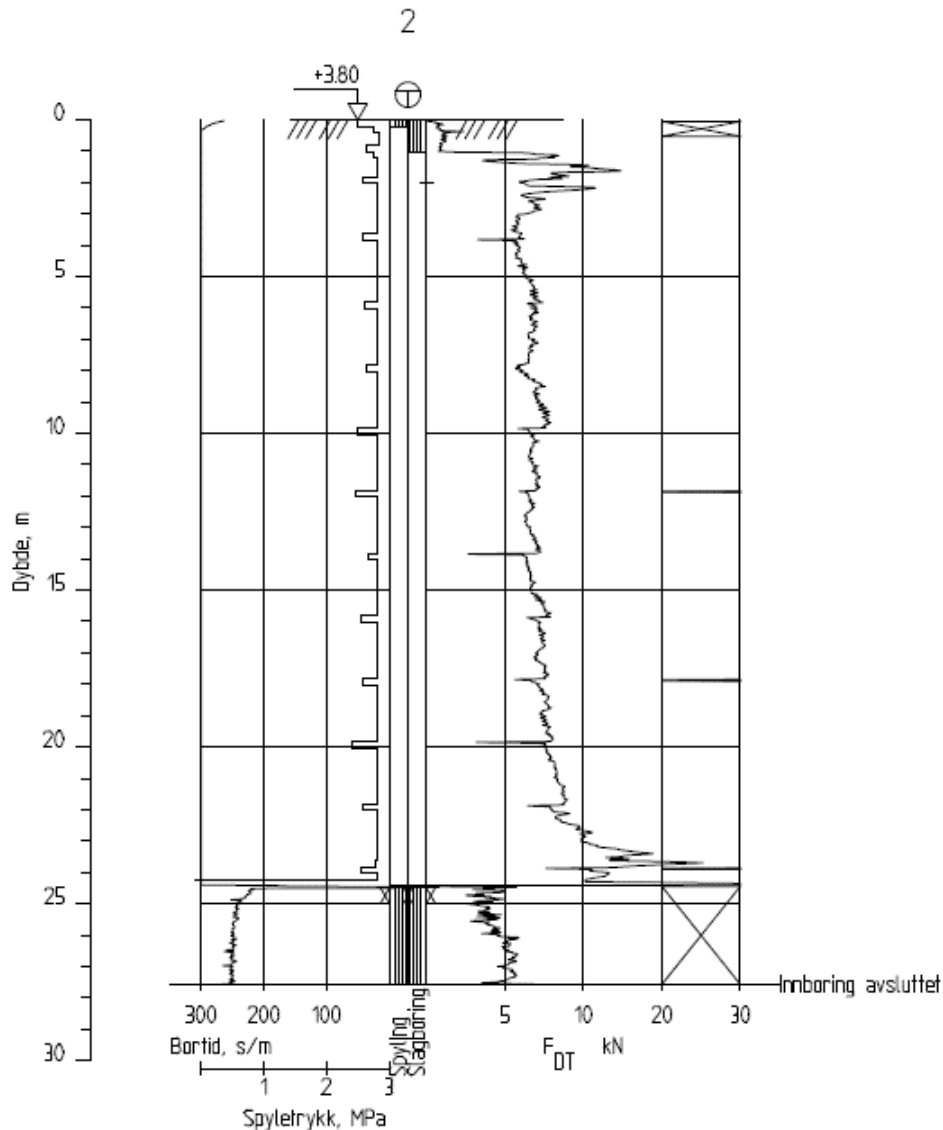


Figure 4.30: Saksvik total sounding close to CPTu profile 2.

To understand why the models aren't switching back to classifying the soil as other material under 15 meters, a plot of the decision parameters versus depth in figure 4.31 shall be studied. Between 6 and 7 meters there is a jump in  $B_q$  to about 0.9 combined with  $Q_t$  and  $F_r$  falling in value. Lower down in the profile the only sign that there's a change in behavior is trough

the slightly increasing  $F_r$ . The XGB model think there is a layer with other material below 11 meters as it requires  $F_r$  to be less than about 0.6 % to classify it as quick clay. In theory this could mean that the XGB model can be used as a lower bound way of classifying quick clay, however uncertainties in the friction response might lead to too low values being registered, giving possibilities of false positives. The CPTu 7R from Saksvik show very low values for  $F_r$ , however the  $B_q$  response is low enough that most models don't predict quick clay.

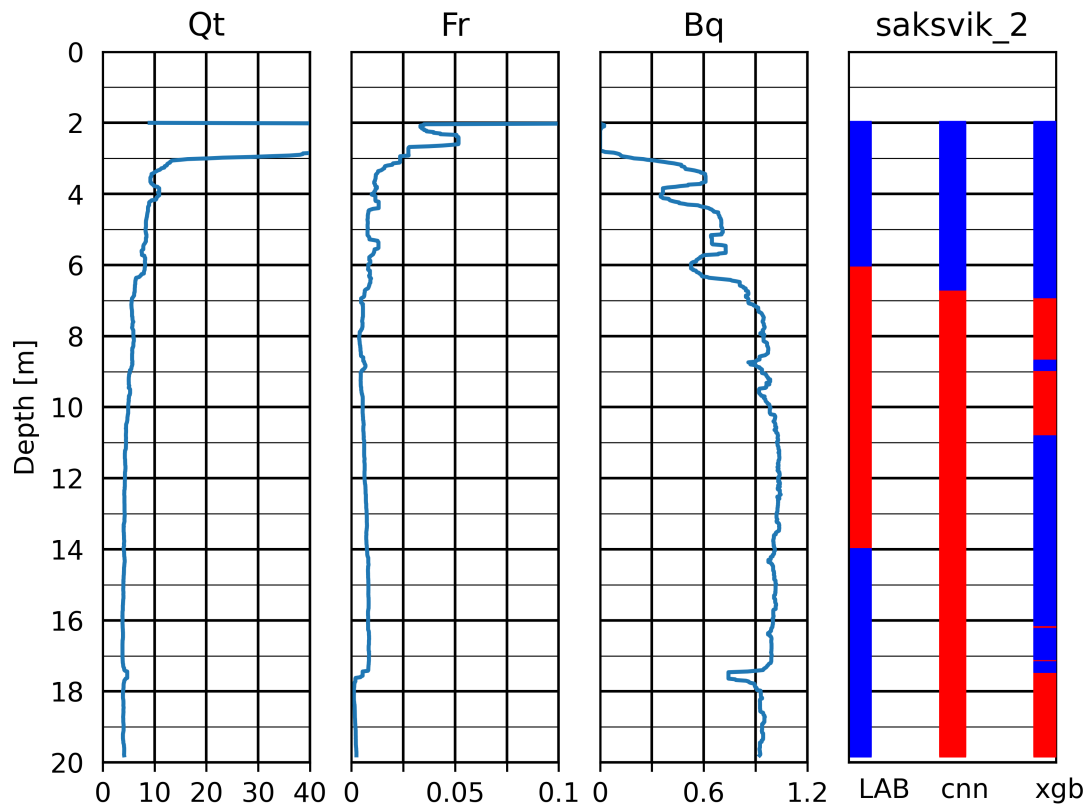


Figure 4.31: Derived values  $Q_t$ ,  $F_r$  and  $B_q$  for CPTu 2 from Saksvik plotted versus depth along with lab classification and model predictions by CNN and XGB.

Another way of visualizing why the models doesn't pick up on the other material layer below 14 meters can be done in a 3D plot. Figure 4.32 shows the CPTu scattered in colors compared to the lab. The dark red points clustered around  $-0.4 F_r^{norm}$  corresponds to depths between 14 and 17 meters, while the cluster with  $F_r^{norm}$  less than  $-0.75$  correspond to depths below 17 meters. While the 14 to 17 meter cluster show slightly higher  $F_r$ , the  $Q_t$  and  $B_q$  values are very similar to those found in the quick clay layer. Below 17 meters the friction is even lower combined with  $Q_t$

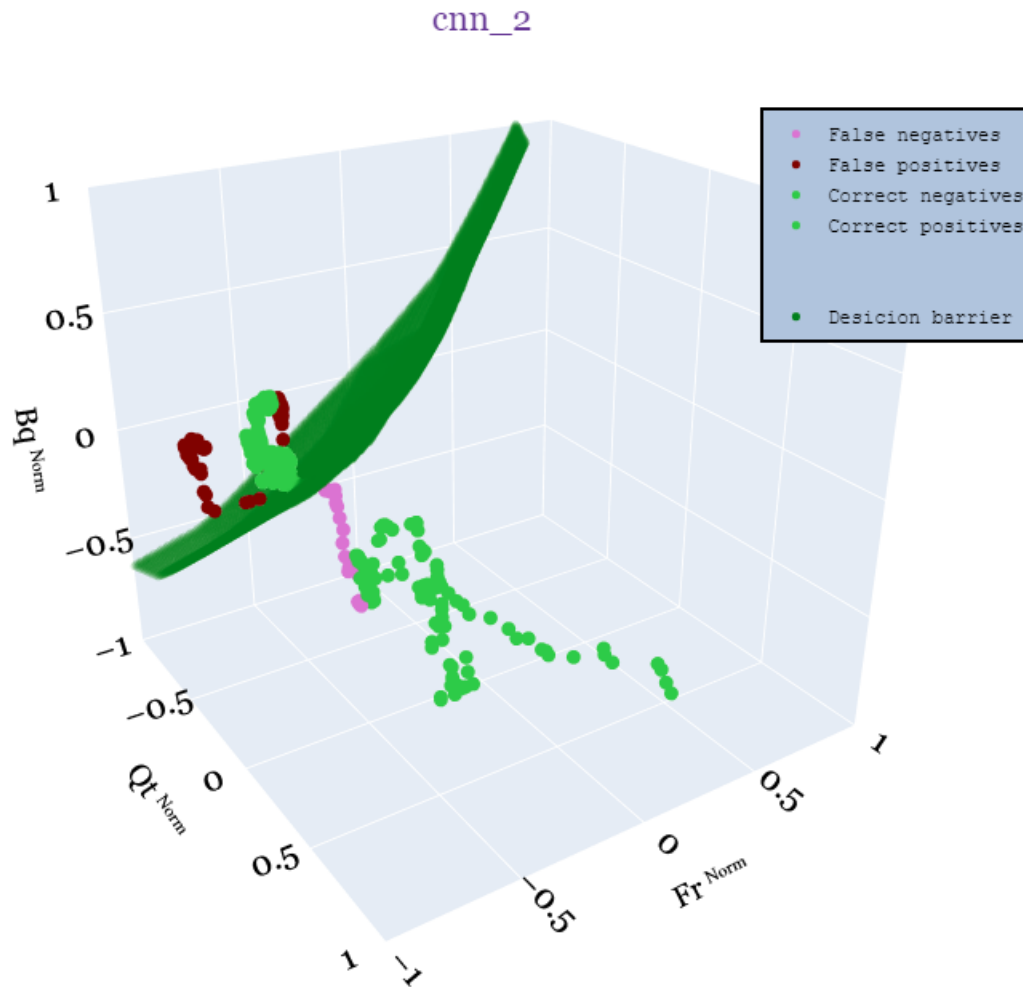


Figure 4.32: CPTu 2 from Saksvik scattered in a 3D plot. The CNN model classifying the points are shown in dark green.

being at its lowest and  $B_q$  similar to what it was around 7 meters. If a model is supposed to be able to classify the points at 7 meters as quick clay, it is therefore not a surprise that the points below 17 meters also has to be classified as such.

Since 5R, 7R and S3 is assumed to not consist of quick clay and most of the algorithms predict the profile correctly they will not be discussed in detail. However, a 3D plot summarizing the CPTus can be found in the appendix in figure [E.14](#).

In the profiles 8R, 9 and S9 almost all the models predict that there is quick clay, unfortunately however due to the lack of laboratory testing at these depth only speculations about their correctness can be made. In figure [4.33](#), the CPTus with unknown layering are plotted in 3D, where

the depths that are classified as quick by the CNN are scattered. CPTu 2 are also included to show how they lay in relation to the three others. Here it is evident that there are similarities between the quick clay layer at CPTu 2 and the response at the unknown layers in CPTu 8R, 9 and S9. Some of the points have slightly lower  $F_r$  values than the others, while one cluster shows the lowest friction measured on the whole site. Also notable is that 8R and S9 end up quite deep into the decision barrier, meaning that the CNN model is confident that there is quick clay.

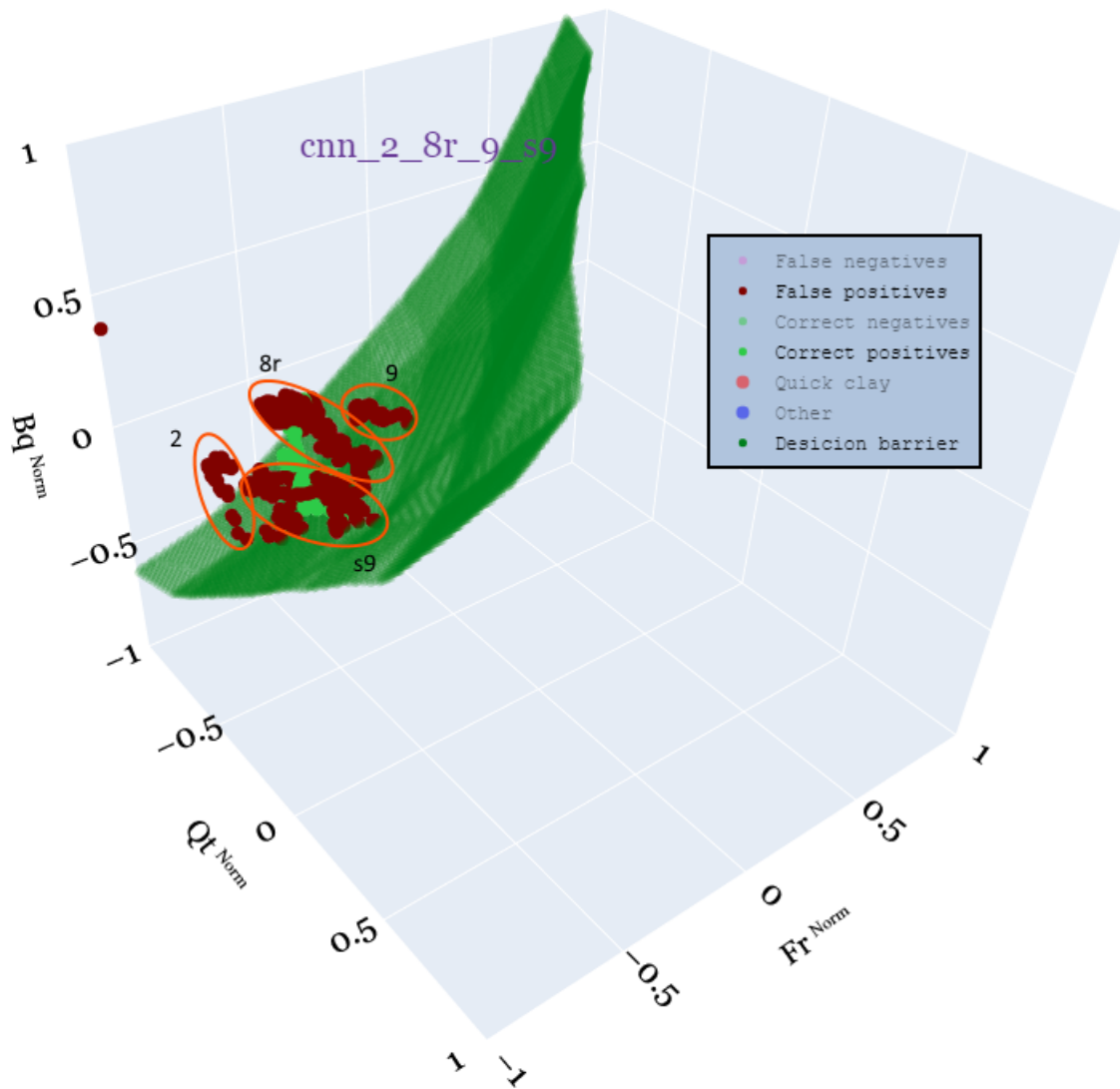


Figure 4.33: 3D plot where CPTu 2, 8R, 9 and S9 are scattered. Only depths where the CNN model classify quick clay are shown while the remaining points are hidden. Note that points marked as false positives are points with no lab data. The light green cluster are the true quick clay points at CPTu 2.



After testing the models on this dataset it is clear that they do not perform as well as on dataset I. While they catch the quick clay behavior at the start on CPTu 2, most of them keep predicting quick behavior even past the point where the lab suggest otherwise. The reason for the overprediction problem is mostly due to the CPTu response not showing any clear difference between the layers, at least for the parameter set used. If it turns out that the uncertain profiles have quick clay it would suggest that the models could be useful for predicting quick clay where lab data is missing, however more testing is required to confirm this. If the opposite is true however, it would most likely mean that the models are too eager when classifying quick clay, and may suffer from high false positive rates.

## Chapter 5

# Summary and Recommendations for Further Work

### 5.1 Summary and Conclusions

In this thesis the applicability of machine learning to detect quick clay from CPTu data was studied. Models were trained with seven machine learning algorithms on two different datasets. The models were both tested in the case where the training and testing datasets were the same and where the datasets differed.

The machine learning models were applied in python through open source libraries, while the additional code for data processing, training procedures and visualization was created specifically for this thesis.

In the case where the training and testing sets were the same, all the algorithms performed well due to the homogeneity of the dataset. The median performance of all the models were 97% or higher with the worst being ELM and the overall best being SVM with a mean error of 2.2%. Visualization of how the different algorithms separates the classes were done by plotting the decision barriers in two and three dimensional plots. The tree based algorithms gave barriers with straight edges while the neural networks resulted in smoother transitions. SVM separated the data with a line in 2D and a plane in 3D, while KNN drew the boundary according to the nearest points in the dataset.

The difference in performance when using two parameters instead of three were not that big on dataset I. Some models preferred two parameters while some did improve when increasing to three parameters. The average performance seem to favor models with three parameters which is why it was preferred when training the algorithms on dataset II.

When testing the models on a dataset not a part of the training set, the performance took an expected hit. However, when testing the algorithms on dataset I, the performance is relatively good since the site is very homogeneous. Neural networks such as CNN and DNN generally performed well, with median scores between 86-94% and mean error between 6.4-13.9%. ELM were one of the top performers in both approaches with a median performance of 94% and a low mean error. Decision tree algorithms such as extreme gradient boost and random forest were the least accurate, where XGB performed worst overall. SVM showed the best performance overall with 95-97% median and mean error of 4.8-6.2% from both approaches.

The two different approaches used to train the models on dataset II showed that training on a reduced version of the dataset where the labels were certain, resulted in a models with simpler decision boundaries than for the full dataset. This ended up also giving higher accuracies which is why it became the preferred approach.

Testing the algorithms on dataset III resulted in a much higher variance in performance. XGB and ELM are the algorithms that seem to differ the most from the others, both overfitting and underfitting. SVM, CNN and DNN all end with very similar classification profiles with SVM being the most likely to classify quick clay and DNN being the least likely. RF struggle somewhat to predict correctly on dataset I, but end up predicting very similar to CNN on dataset III, despite having different architectures.

The results found in this thesis indicate that an approach using machine learning to detect quick clay can give good accuracy. The observed performance is dependent on what algorithm is used, parameter tuning and the dataset it is trained on. While it can give indication of the existence of quick clay, it is not able to prove it as this requires the measurement of remolded shear strength. The machine learning models might be useful for the geotechnical engineer when deciding where sampling are needed and where it can be skipped.

## 5.2 Discussion

One of the factors that limits the scope of the results of this thesis are the relative small training and testing datasets. To obtain models that can be applied in practice, more data has to be gathered both for training and testing.

When expanding the datasets however it becomes difficult for the creator of the models to have an overview of each individual CPTu in the set. This might increase the occurrence of faulty and wrongly labeled points, which can influence the performance of the models.

While very high accuracy was found when training and testing the models on the same dataset, the practicality of such an approach is quite limited. For an average project the time required to perform the necessary training is probably too high, however for bigger projects where quick clay is a concern it might be worthwhile. Especially in the case where field testing is done in several rounds it might become useful to select where more sampling is required.

In the case where only a certain part of a clay layer is quick, it becomes difficult for the models to single out the correct depths. The reason for this stems from the fact that the registrations from the CPTu does not necessarily show a clear change in behavior between the quick and non-quick clay.

The measured  $Q_t$ ,  $F_r$  and  $B_q$  values in quick clay might be significantly different depending on the site. Soft clays might in some cases show lower  $Q_t$  and  $F_r$  values than found in quick clay. As the CPTu tend to be limited by three measured parameters, it simply is not able to describe all the necessary details of a soil to determine whether it is quick or not. The performance of the machine learning models is naturally bounded by how distinguishable the CPTu data is, and sometimes quick clay and other material points overlap each other, resulting in misclassifications.

In the practice of machine learning it is possible to do what is called parameter optimization to obtain higher accuracy (Bergstra, 2012). This is often done by testing a wide range of different values for the model parameters. This process requires the models to be cross validated on data not part of the training set to avoid overfitting. Due the lack of a diverse testing dataset, model parameters were set manually with emphasis on how they affected the decision boundaries.

### 5.3 Recommendations for Further Work

Machine learning is a relatively new concept in geotechnical engineering and may with time be broadly implemented in the field. Cone penetration testing is a good match with machine learning since each CPTu sounding creates a lot of raw data ready for interpretation. Further development of the methods however must be investigated more deeply in order to find all the possibilities and increase the performance of the algorithms. Presented below is a list of recommendations for further work on the topic of machine learning in geotechnical engineering:

1. Incorporate a bigger dataset for training the algorithms, containing CPTu data from all across Norway, both sensitive and nonsensitive material, in order to get more generalizable predicting models.
2. Test the different machine learning algorithms on sites with more complex soil layering across Norway. It could also be interesting to research if the algorithms could detect thin quick clay layers in between more solid material.
3. Extend the machine learning algorithms in this thesis to classification of other soil types than quick and brittle clay, for example sand, silt and clay. [Erharter et al. \(2021\)](#) at Graz University of Technology uses neural networks to classify six soil types. Other machine learning algorithms such as SVM and CNN could be explored concerning this problem.
4. Extend the machine learning algorithms to other sounding methods. Total sounding is a frequently used ground investigation method in Norway that produces a substantial amount of data. If the database is big enough, it might be possible to adapt the machine learning algorithms to make predictions based on the total sounding data. Although it might prove itself difficult to create such models due to the rough nature of the sounding which includes rotation, flushing and hammering. An useful parameter in detecting quick clay with total sounding could be whether the tip resistance is decreasing with depth.
5. Use machine learning algorithm to determine undrained shear strength and overconsolidation ratio in the soil based on CPTu data. Raw data such as corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$  together with laboratory tests such as oedometer, uniaxial and triaxial tests can be gathered into datasets, with implementation of ML

algorithms to derive correlations between the raw data, shear strength and OCR. Such correlations based only on raw data from CPTu might be difficult to define as shear strength and overconsolidation ratio often rely on variables such as plasticity index and water content.

6. Perform a parameter study on which CPTu parameters gives the best prediction performance on a given site. This thesis used  $Q_t$ ,  $B_q$  and  $F_r$ , although there are many other derived parameters from CPTu. [Valsson \(2019\)](#) researched which features provided the best prediction. However, soil conditions vary from site to site and the best fit parameters might change depending on the site.
7. Implement machine learning algorithms to the computer at the boring rig in order to obtain in-situ information about soil layering in real time. The earlier the investigation indicates quick clay the earlier the consulting engineers can adapt.
8. Create a global model or program that is able to determine in-situ soil layering from CPTu soundings on new site investigations based on a database containing data from all around Norway.

# Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. <https://www.tensorflow.org/>.

Aizerman, M. A. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837.

Akusok, A., Bjørk, K., Miche, Y., and Lendasse, A. (2015). High-performance extreme learning machines: A complete toolbox for big data applications. *Access, IEEE*, 3:1011–1025.

Alpaydin, E. (2020). *Introduction to machine learning Fourth edition*. Cambridge, Massachusetts: The MIT Press, 2020.

Ayodele, T. O. (2010). Types of machine learning algorithms. *New advances in machine learning*, 3:19–48.

Bergstra, J. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Berrum, M. and Skaar, H. (2020). Application of correlations and machine learning on cptu. *Geotechnical engineering, Specialization Project TBA4510*.

Breiman, L. (2001). *Random Forests*, chapter 1, pages 5–32. Kluwer Academic Publishers, Netherlands. Statistics Department, University of California, Berkeley, CA 94720.

- Bæverfjord, M. G., Døssland, T., Eknes, A., Hagberg, K., Handberg, A., Jønland, J., Nerland, O., Rundmo, O. E., and Sandven, R. (2010). Veiledning for utførelse av trykksondering. *NGF melding 5*.
- CEN (1992). *Eurocode 2: Design of concrete structures - Part 1-1: General rules and rules for building*. NS-EN 1992-1-1:2004+A1:2014+NA:2018.
- Chen, T. and Guestrin, C. (2016a). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, pages 785–794, New York, NY, USA. ACM.
- Chen, T. and Guestrin, C. (2016b). XGBoost documentation: Parameters. <https://xgboost.readthedocs.io/en/latest/parameter.html>.
- Chollet, F. et al. (2015). Keras. <https://github.com/fchollet/keras>.
- Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, 27.
- Ding, S., Su, C., and Yu, J. (2011). An optimizing bp neural network algorithm based on genetic algorithm. *Artificial intelligence review*, 36(2):153–162.
- Erharter, G., Oberhollenzer, S., Fankhauser, A., Marte, R., and Marcher, T. (2021). Learning decision boundaries for cone penetration test classification. *Comput Aided Civ*, 36:489–503.
- Godoy, C. (2019). Site characterization using kriging and machine learning approaches. *Master's thesis in Geotechnics and Geohazards*.
- Godoy, C., Depina, I., and Thakur, V. (2020). Application of machine learning to the identification of quick and highly sensitive clays from cone penetration tests. *Journal of Zhejiang University-SCIENCE A*, 21(6):445–461.
- Gundersen, A. S., Hansen, R. C., Lunne, T., L'Heureux, J. S., and Strandvik, S. O. (2019). Characterization and engineering properties of the ngts onsøy soft clay site. *AIMS Geosciences*, 5(3):665–703.
- IBM (2020). Neural networks. <https://www.ibm.com/cloud/learn/neural-networks>.



- Kartverket (2021). Norgeskart. <https://norgeskart.no/#!?project=norgeskart&layers=1002&zoom=3&lat=7197864.00&lon=396722.00>.
- Lai, J., Wang, X., Li, R., Song, Y., and Lei, L. (2020). Bd-elm: A regularized extreme learning machine using biased dropconnect and biased dropout. *Hindawi*, page 7.
- L'Heureux, J.-S. (2013). Revurdering av faregraden for 20 kvikkleiresoner i strandsonen. *NIFS*, page 161.
- L'Heureux, J.-S., Gundersen, A. S., D'Ignazio, M., Smaavik, T., Kleven, A., Rømoen, M., Karlsrud, K., Paniagua, P., and Hermann, S. (2018). Impact of sample quality on cptu correlations in clay - example from rakkestad clay. *NGI*.
- L'Heureux, J.-S., Lindgård, A., and Emdal, A. (2019). The tiller-flotten research site: Geotechnical characterization of a very sensitive clay deposit. *AIMS Geosciences*.
- Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.
- Love, B. C. (2002). Comparing supervised and unsupervised category learning. In *Psychonomic Bulletin & Review*, volume 4, pages 829–835. Psychonomic Society, Inc.
- Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.
- NGU (2021). Superficial deposits - national database. [http://geo.ngu.no/kart/losmasse\\_mobil/](http://geo.ngu.no/kart/losmasse_mobil/).
- Noble, W. S. (2006). *What is a support vector machine?*, volume 24, pages 1565 – 1567. Nature Biotechnology.
- O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, page 11.
- Pal, M. (2005). Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222.
- Paniagua, P., L'Heureux, J.-S., Carroll, R., Kåsin, K., and Sjørusen, M. (2019). Evaluation of sample disturbance of three norwegian clays. *Norwegian geotechnical institute*.

- 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., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Raileanu, L. E. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. In *Annals of Mathematics and Artificial Intelligence*, volume 41, pages 77–93. Kluwer Academic Publishers.
- Robertson, P. (2016). Cone penetration test (cpt)-based soil behaviour type (sbt) classification system — an update. *Canadian Geotechnical Journal*, 53(12):1910–1927.
- Rolnick, D. and Tegmark, M. (2018). The power of deeper networks for expressing natural functions. *ICLR*, page 14.
- Sanderson, G. (2017). Gradient descent, how neural networks learn | deep learning, chapter 2. <https://www.youtube.com/watch?v=IHZwWFHwa-w>.
- Sandven, R., Gylland, A., Montafia, A., Pfaffhuber, A., Kåsin, K., and Long, M. (2015). Detektering av kvikkleire-sluttrapport. *Norges Geologiske Undersøkelse*.
- Sandven, R., Gylland, A. S., Wangen, P. A., Solberg, I.-L., Montafia, A., Tørym, E., Kåsin, K., and Valsson, S. M. (2019). Veiledning for detektering av sprøbruddmateriale. *NGF melding 12*, page 17.
- Seidl, T., Liu, L., and Tamer, M. (2009). *Nearest Neighbor Classification*, pages 1885–1890. Springer US, Boston, MA.
- Shyamel, P. and Pingel, J. (2017). Introduction to deep learning: What are convolutional neural networks? <https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>.
- Starmer, J. (2020). Xgboost part 2 (of 4): Classification. <https://www.youtube.com/watch?v=8b1JEDvenQU>.
- Valsson, S. (2019). Machine learning to detect sensitive materials with cptu in norway. *ECSMGE-2019 - Proceedings*.

- Valsson, S., Degago, S., and Haugen, E. (2018). Detecting highly sensitive materials with cptu in norway using machine learning. *26th European Young Geotechnical Engineers Conference*.
- Van Rossum, G. and Drake Jr, F. L. (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.
- Wickham, H. and Stryjewski, L. (2012). 40 years of boxplots. *had.co.nz*.
- Winston, P. (2015). Lecture 12a: Neural nets. <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/lecture-12a-neural-nets/>.
- Yuan, G.-X., Ho, C.-H., and Lin, C.-J. (2012). Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584–2603.
- Zhang, L., Zhou, W., and Jiao, L. (2004). Wavelet support vector machine. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):34–39.
- Zhong, Y. (2016). The analysis of cases based on decision tree. *IEEE Xplore*, pages 142–147.

## **Appendix A**

### **CPTu raw data graphs**

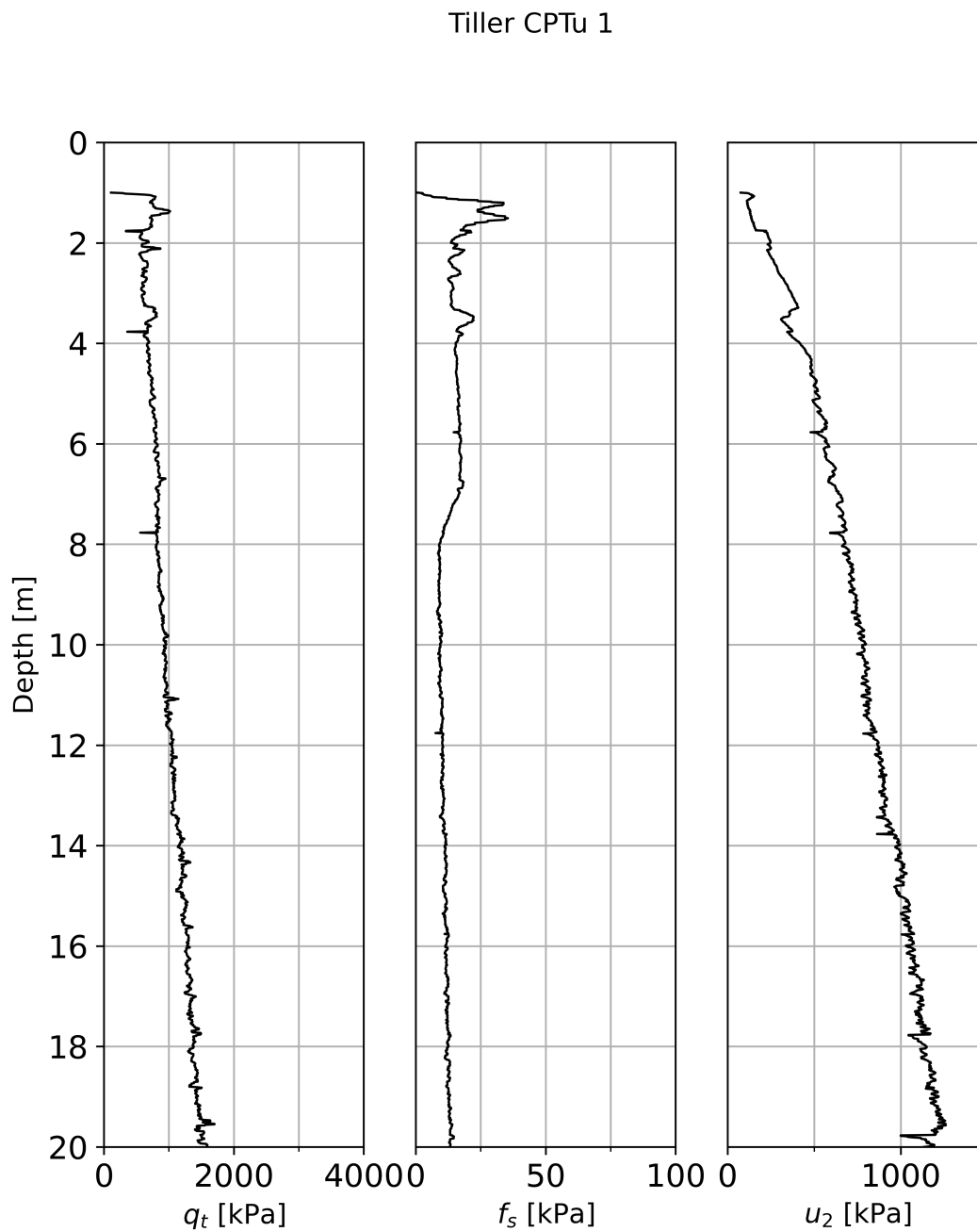


Figure A.1: Raw data graphs of NGTS Tiller-Flotten CPTu 1, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

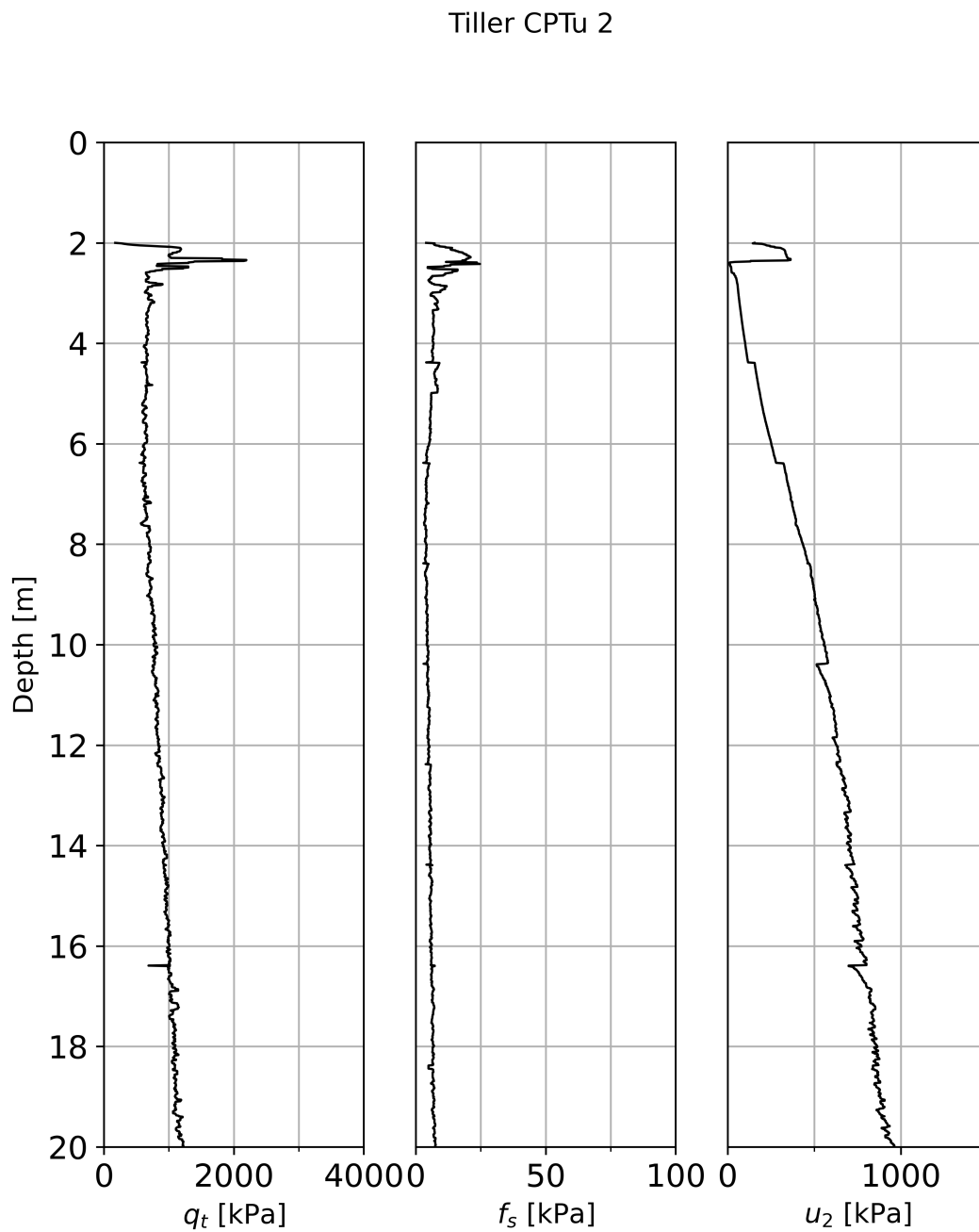


Figure A.2: Raw data graphs of NGTS Tiller-Flotten CPTu 2, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

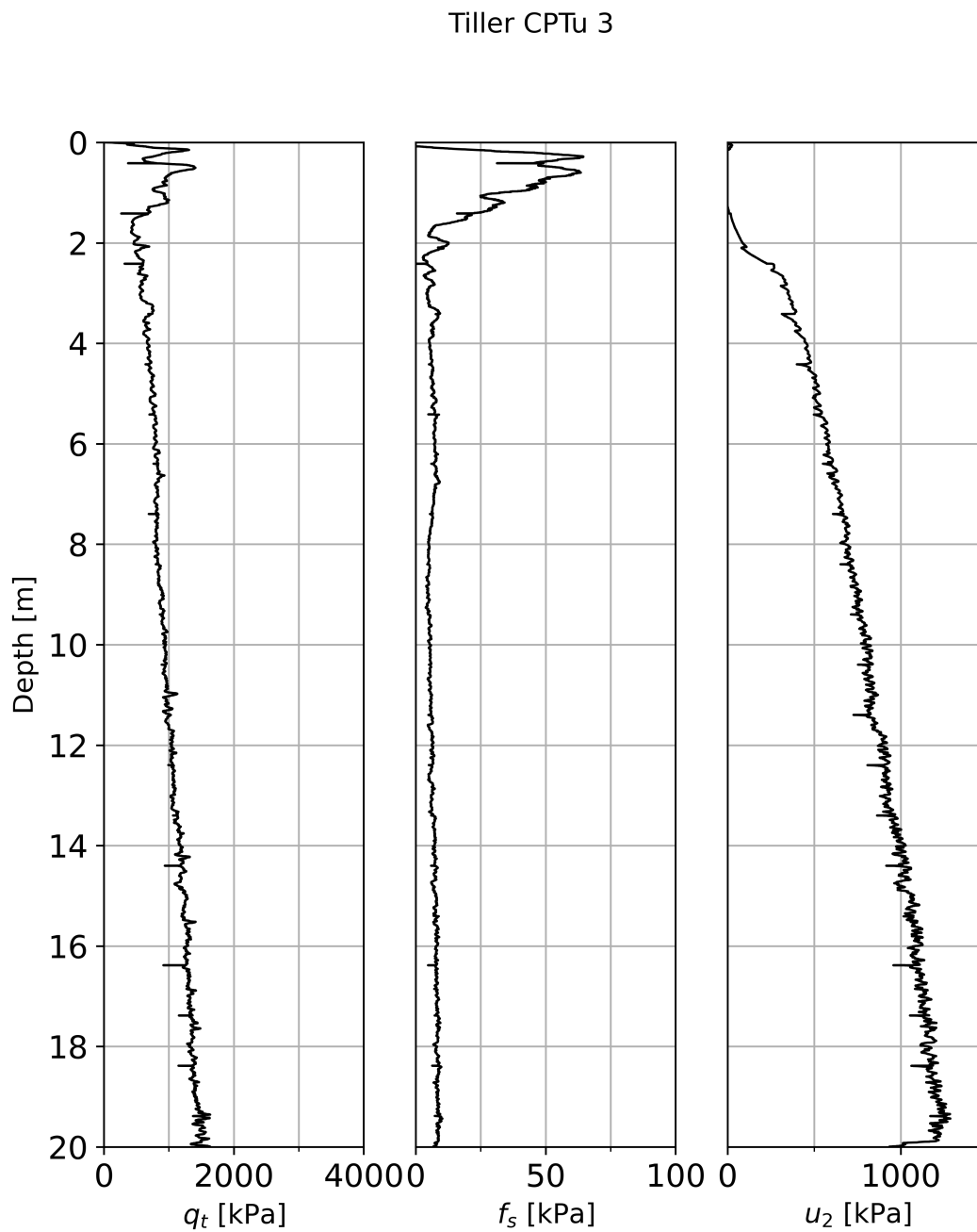


Figure A.3: Raw data graphs of NGTS Tiller-Flotten CPTu 3, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 4

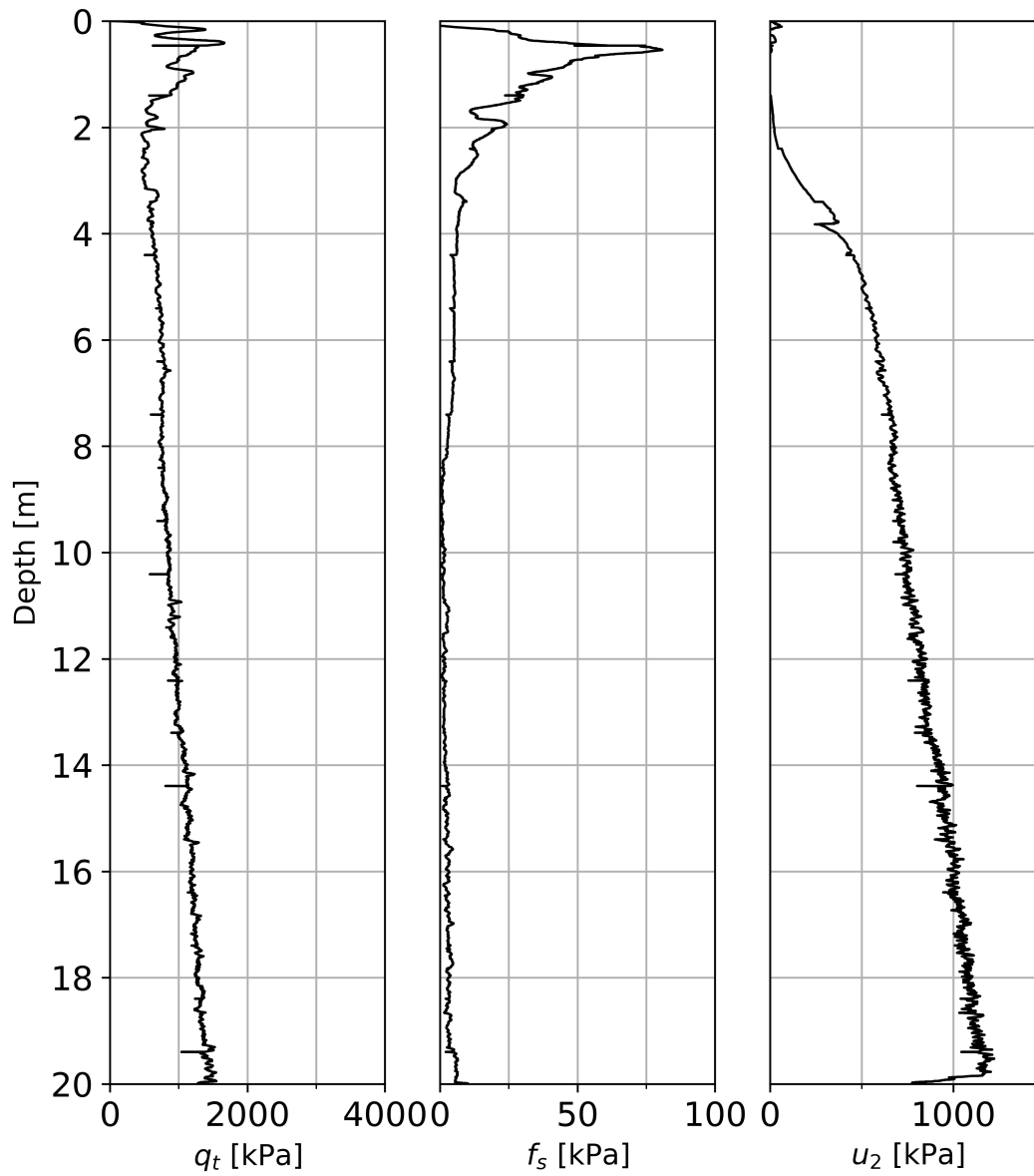


Figure A.4: Raw data graphs of NGTS Tiller-Flotten CPTu 4, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .



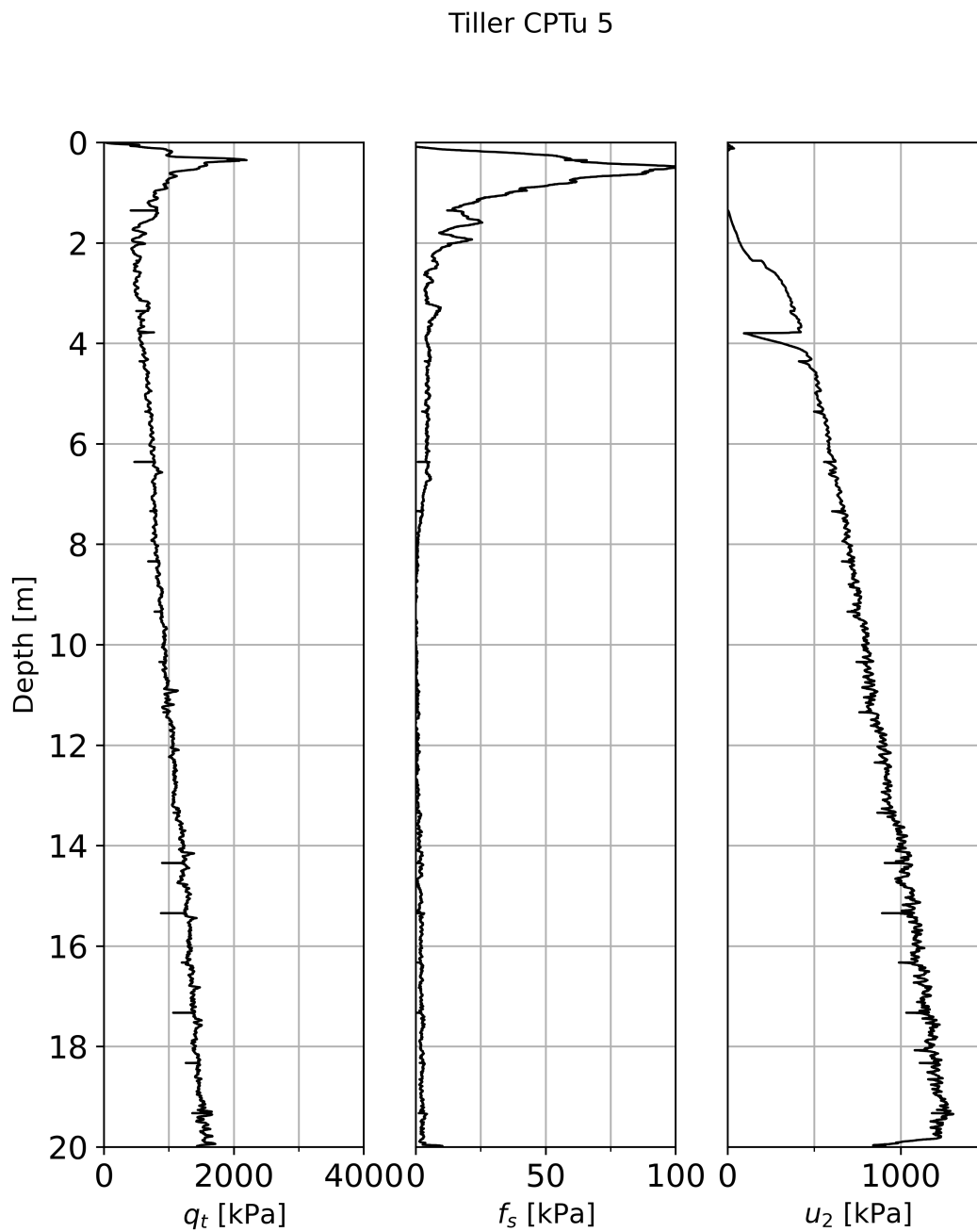


Figure A.5: Raw data graphs of NGTS Tiller-Flotten CPTu 5, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 6

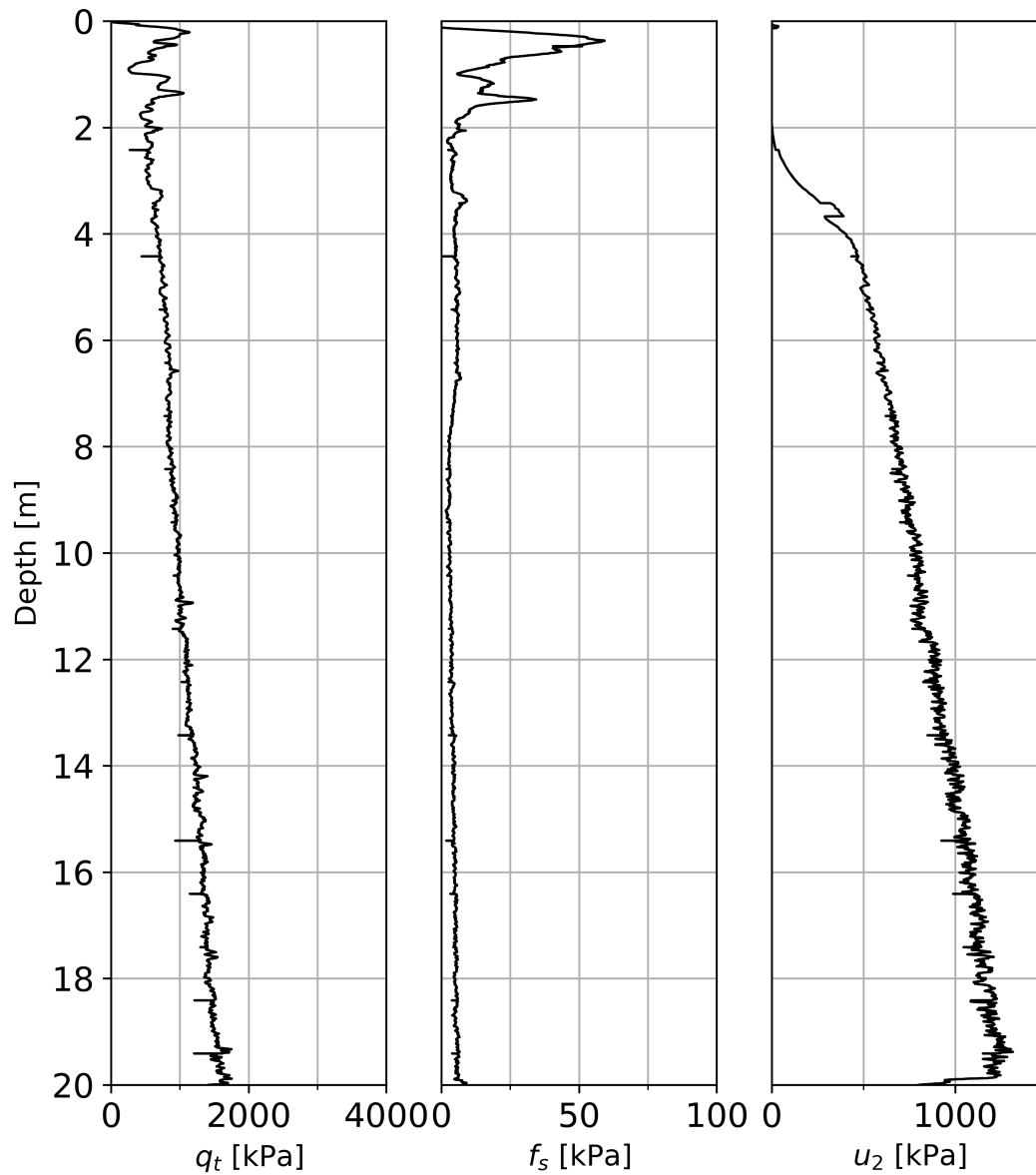


Figure A.6: Raw data graphs of NGTS Tiller-Flotten CPTu 6, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

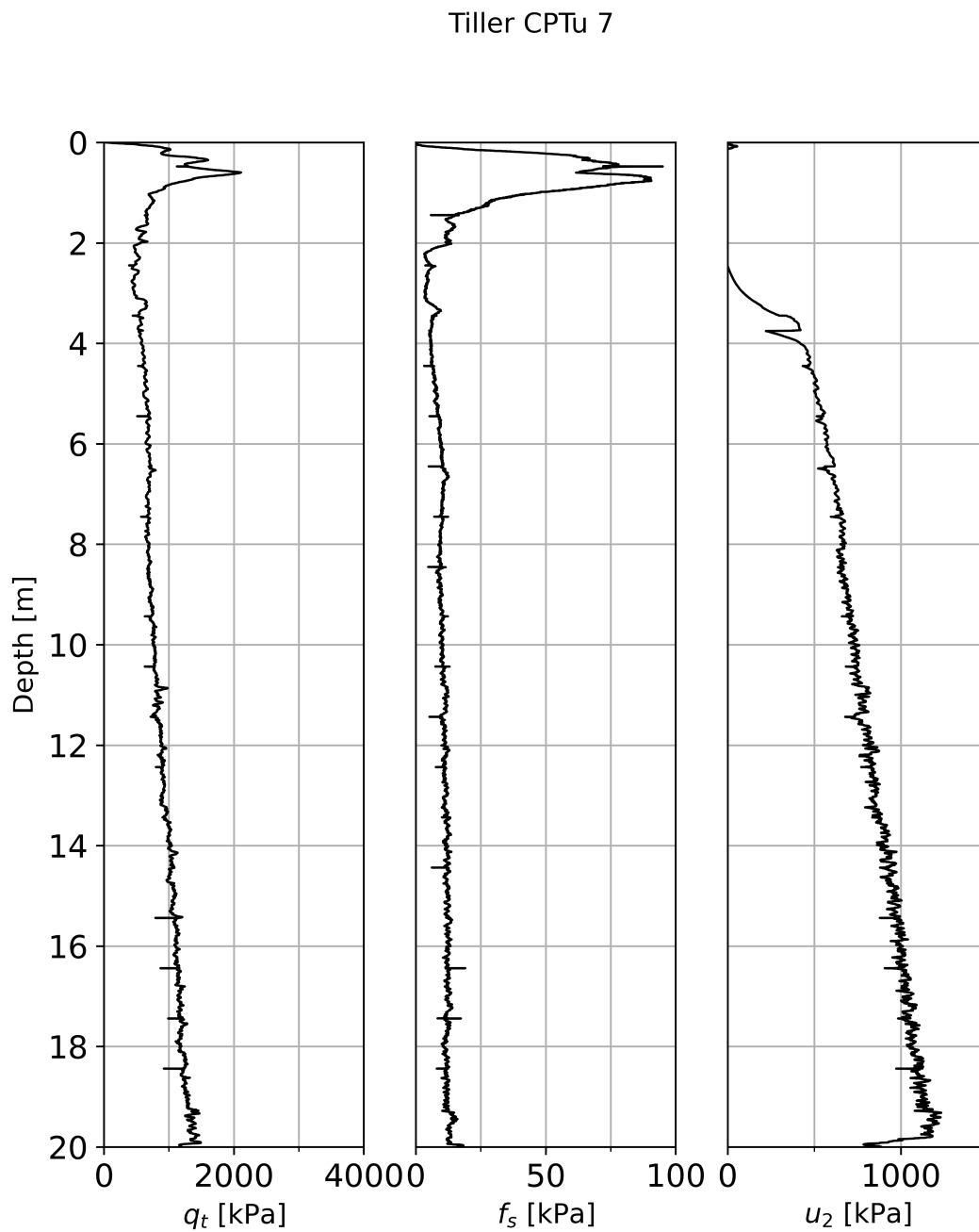


Figure A.7: Raw data graphs of NGTS Tiller-Flotten CPTu 7, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 8

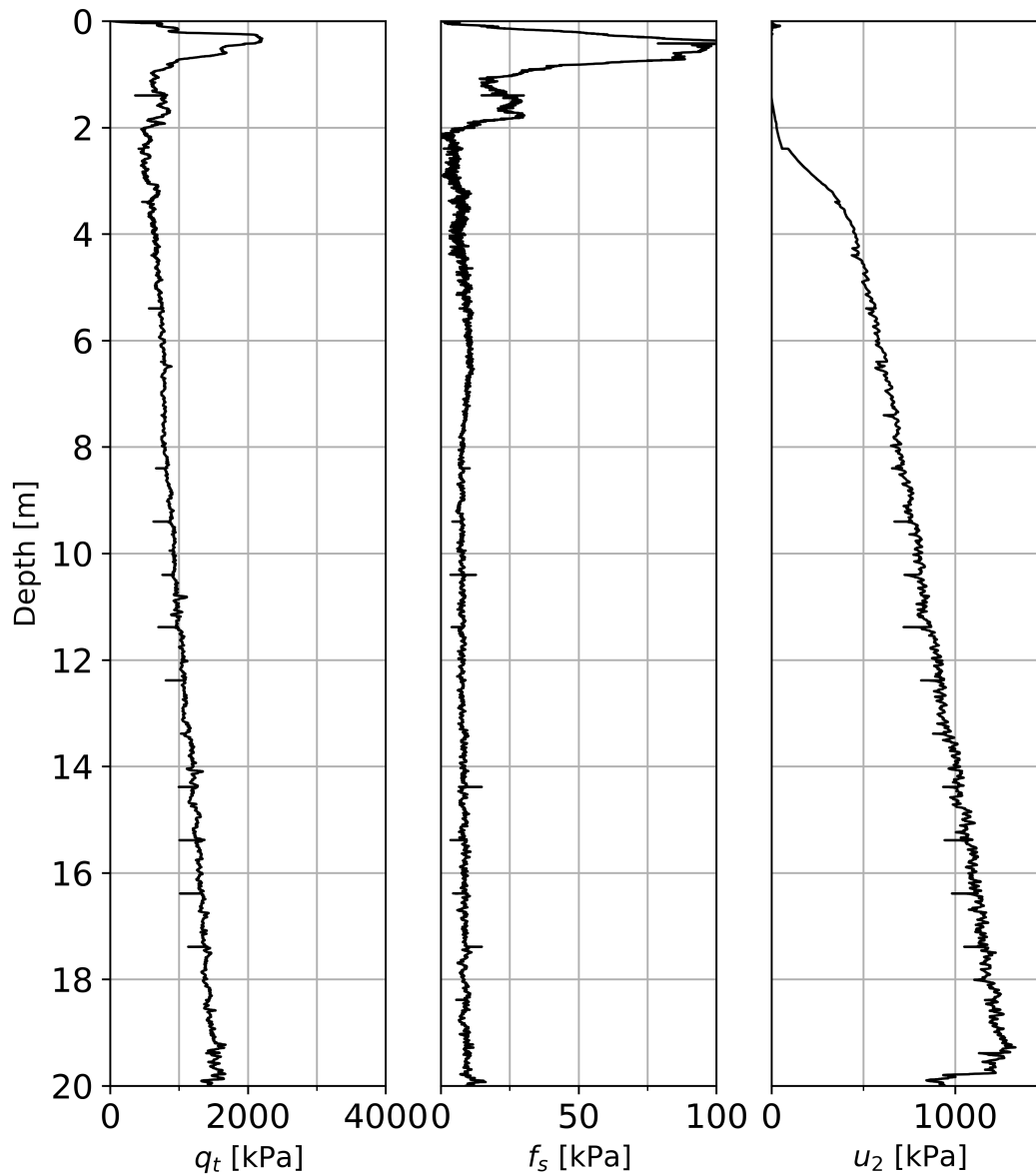


Figure A.8: Raw data graphs of NGTS Tiller-Flotten CPTu 8, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 9

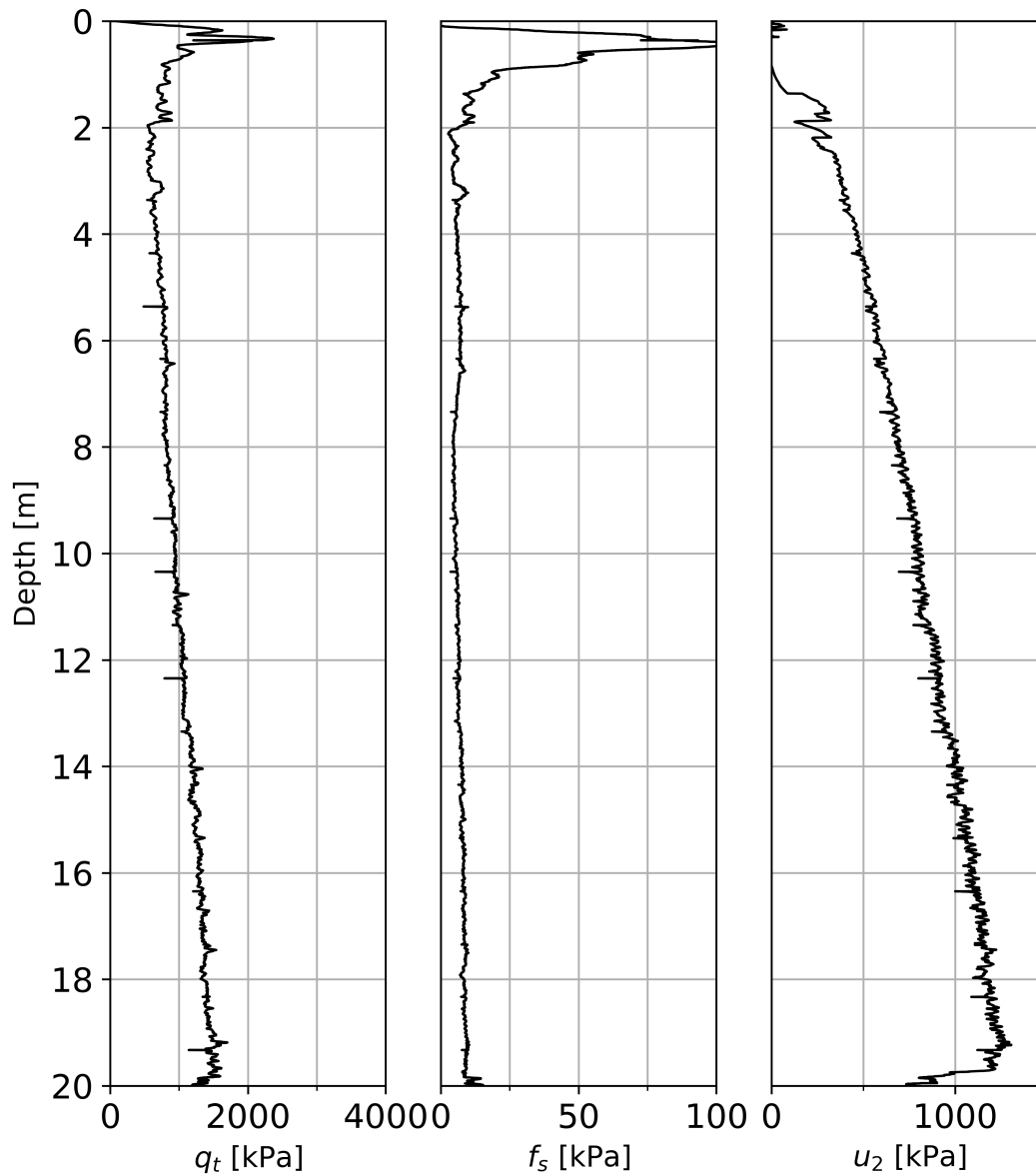


Figure A.9: Raw data graphs of NGTS Tiller-Flotten CPTu 9, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 10

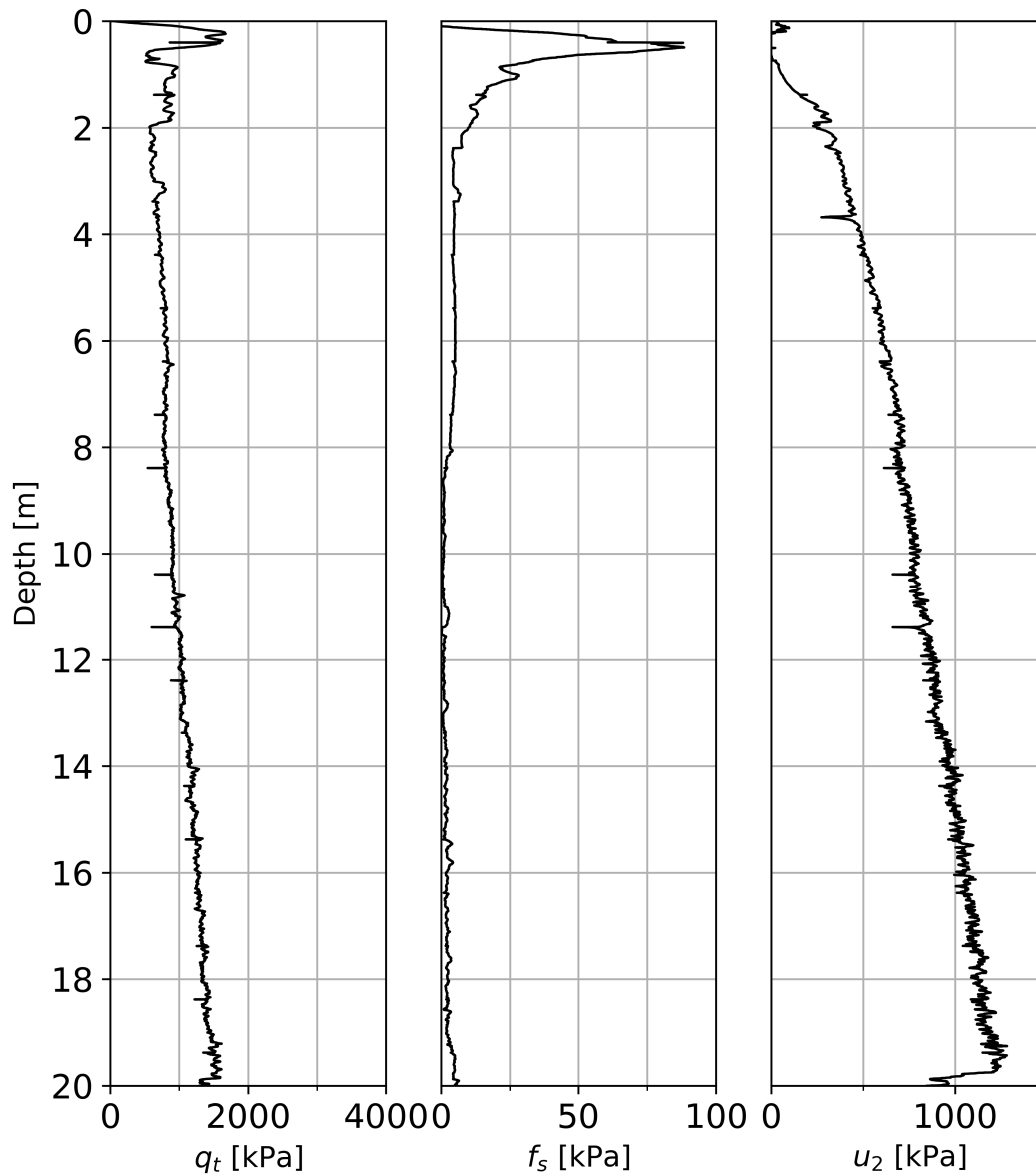


Figure A.10: Raw data graphs of NGTS Tiller-Flotten CPTu 10, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 11

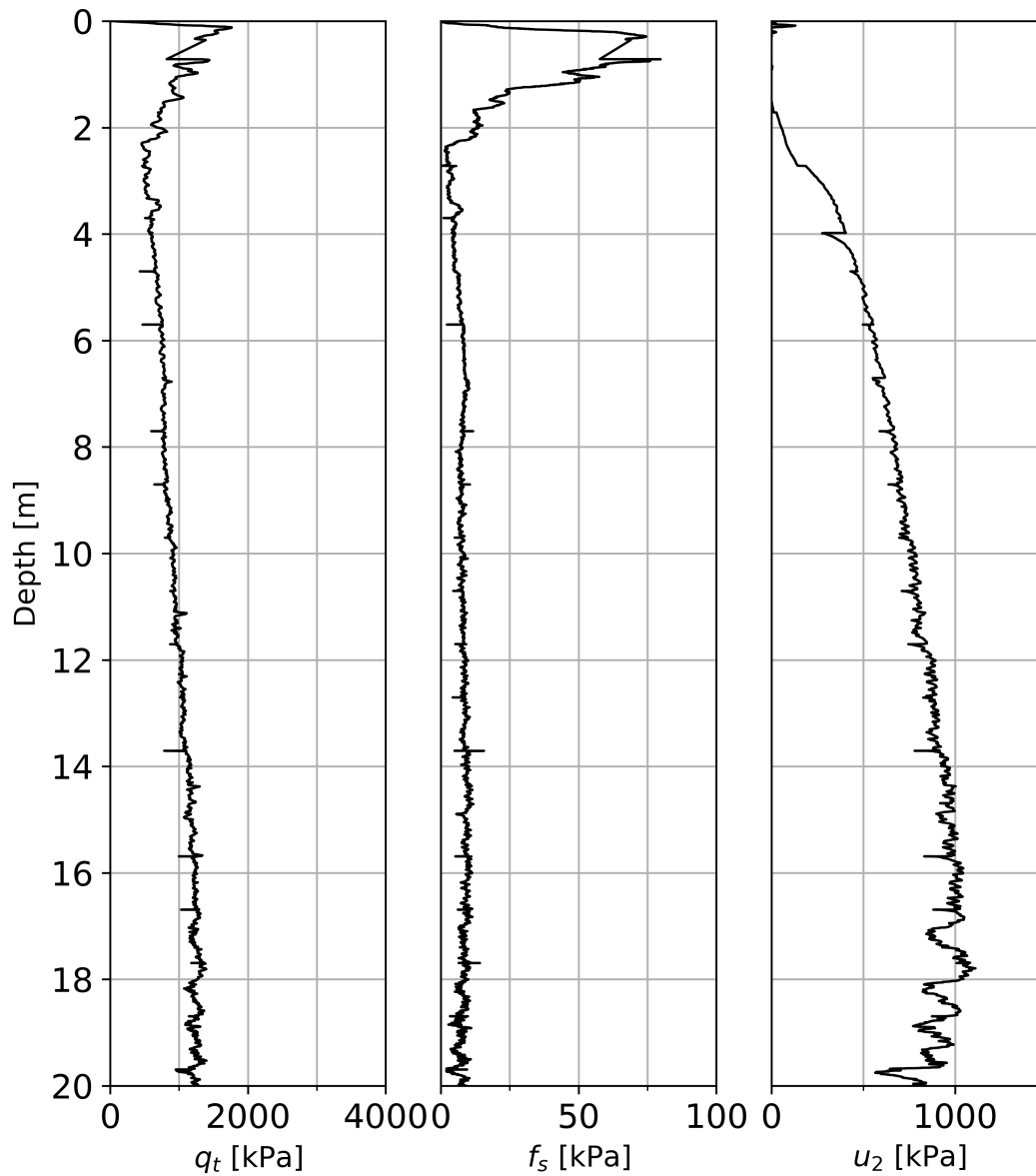


Figure A.11: Raw data graphs of NGTS Tiller-Flotten CPTu 11, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 12

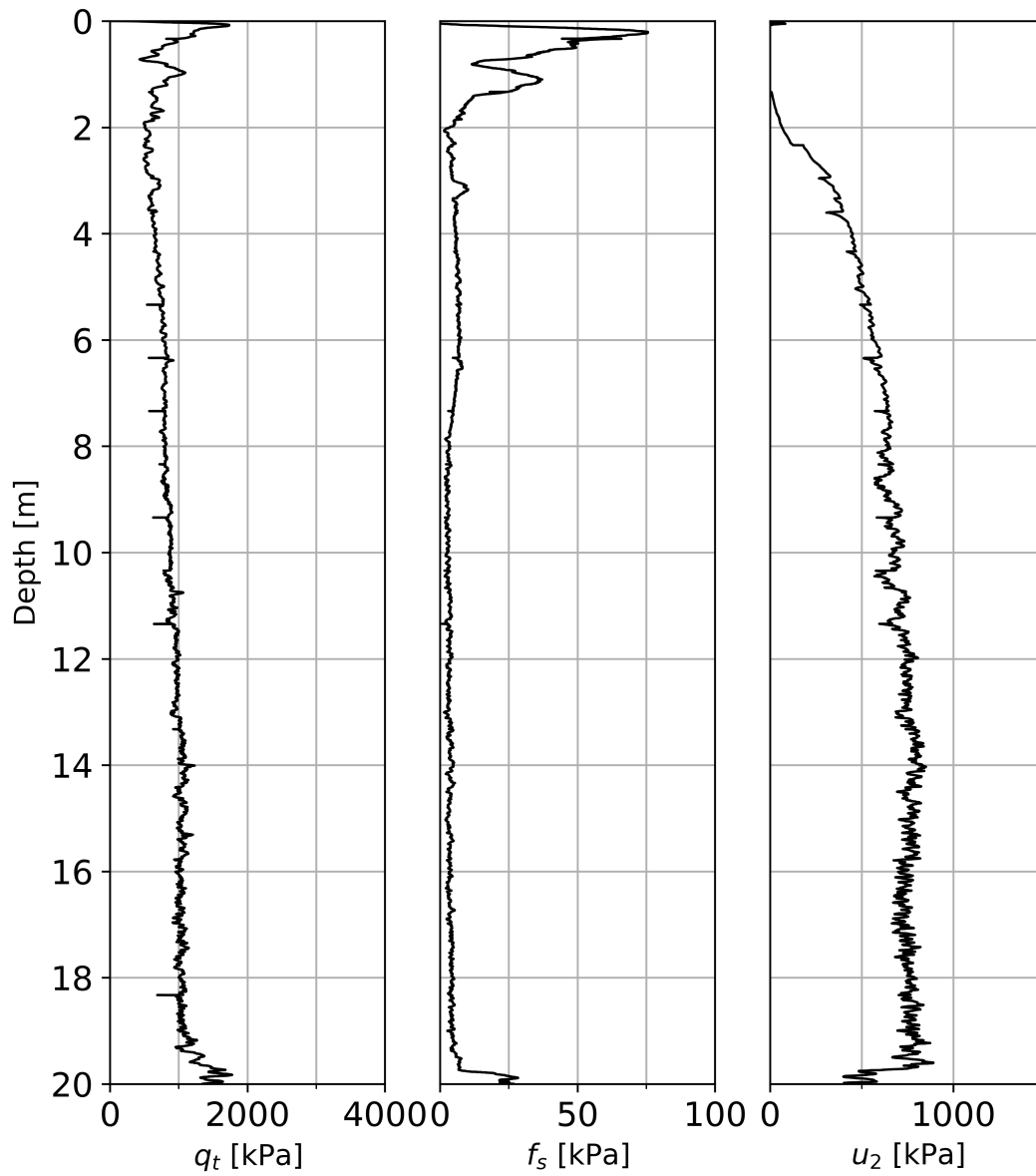


Figure A.12: Raw data graphs of NGTS Tiller-Flotten CPTu 12, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .



## Tiller CPTu 13

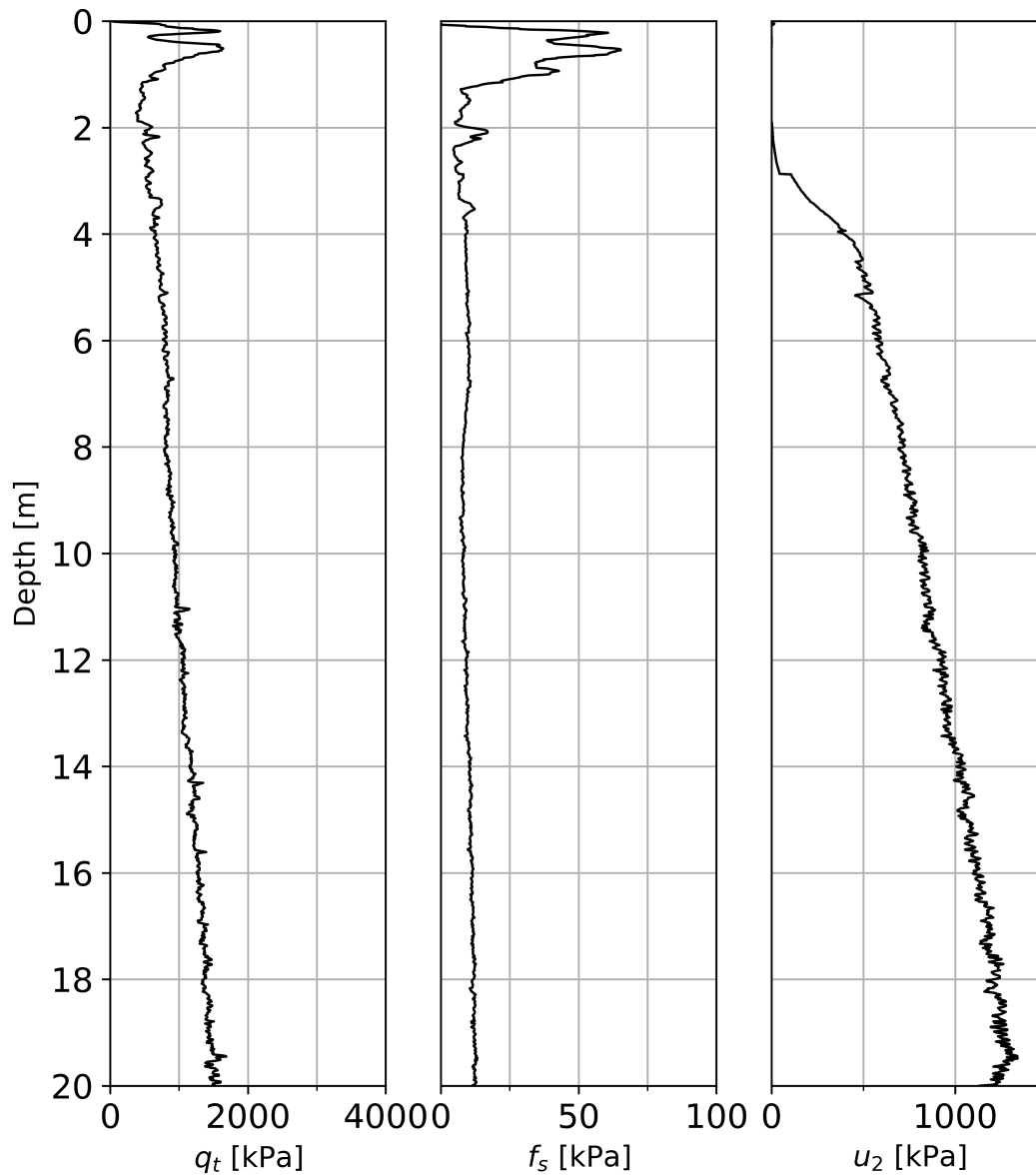


Figure A.13: Raw data graphs of NGTS Tiller-Flotten CPTu 13, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 14

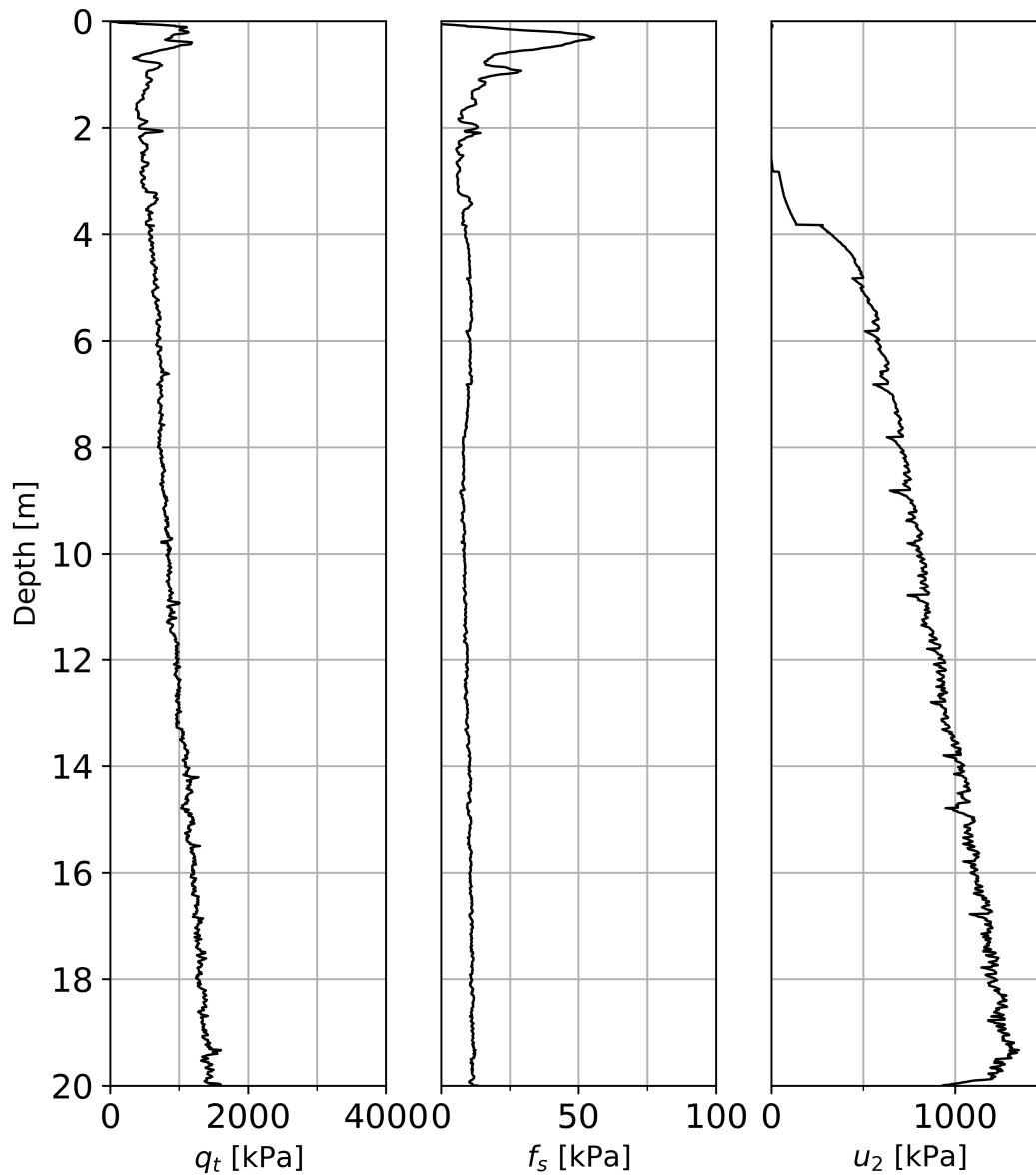


Figure A.14: Raw data graphs of NGTS Tiller-Flotten CPTu 14, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 15

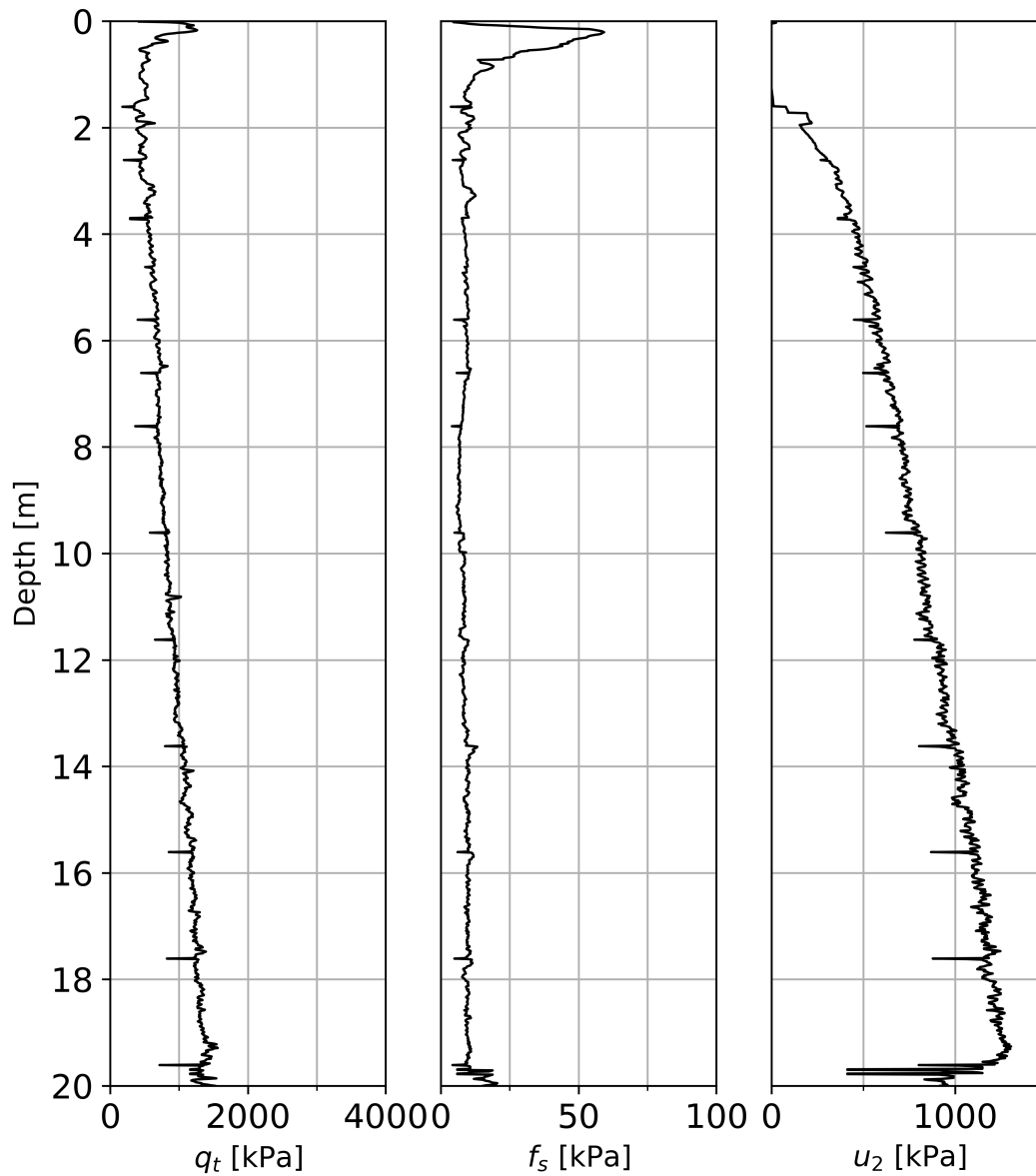


Figure A.15: Raw data graphs of NGTS Tiller-Flotten CPTu 15, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 16

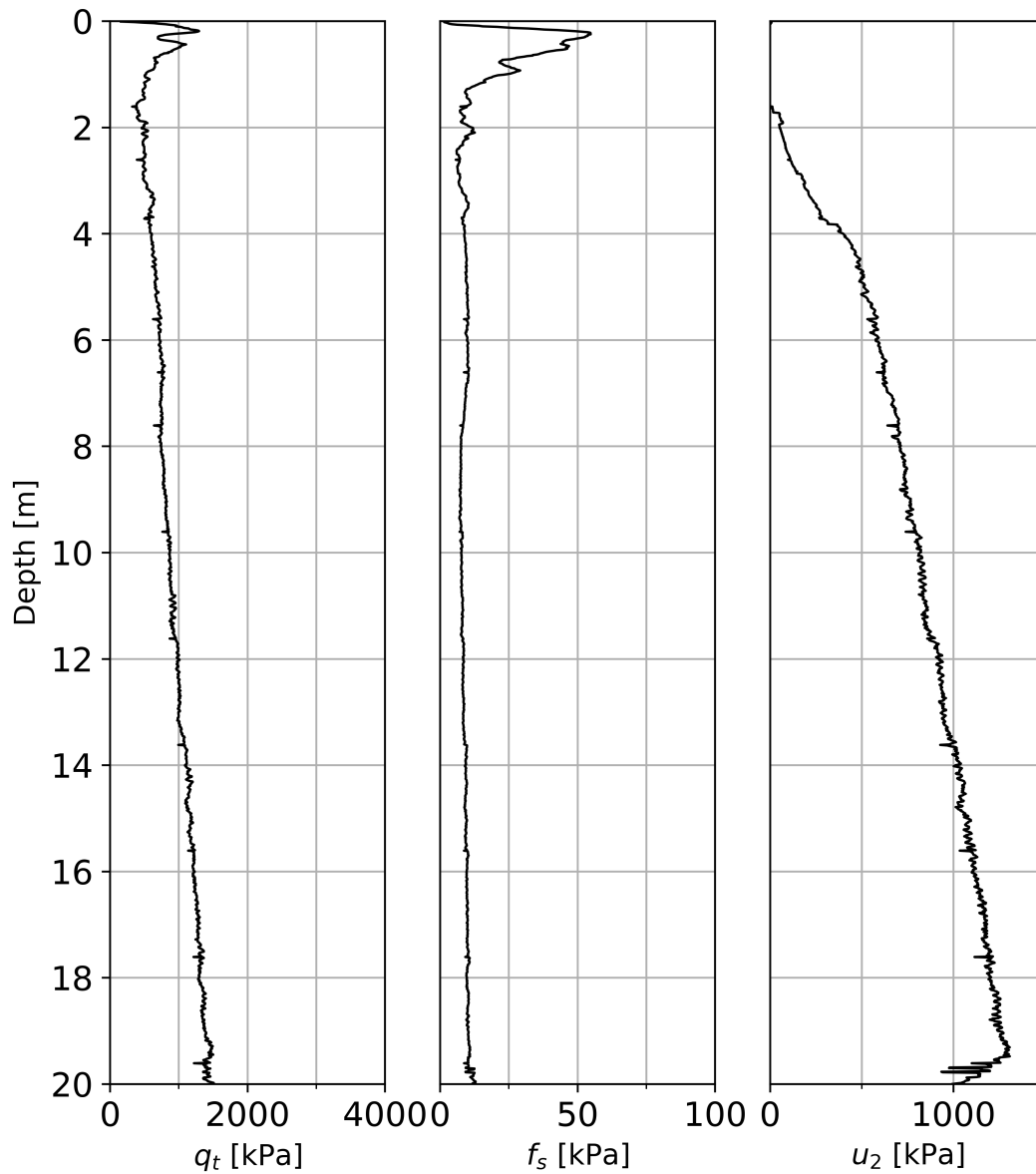


Figure A.16: Raw data graphs of NGTS Tiller-Flotten CPTu 16, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 17

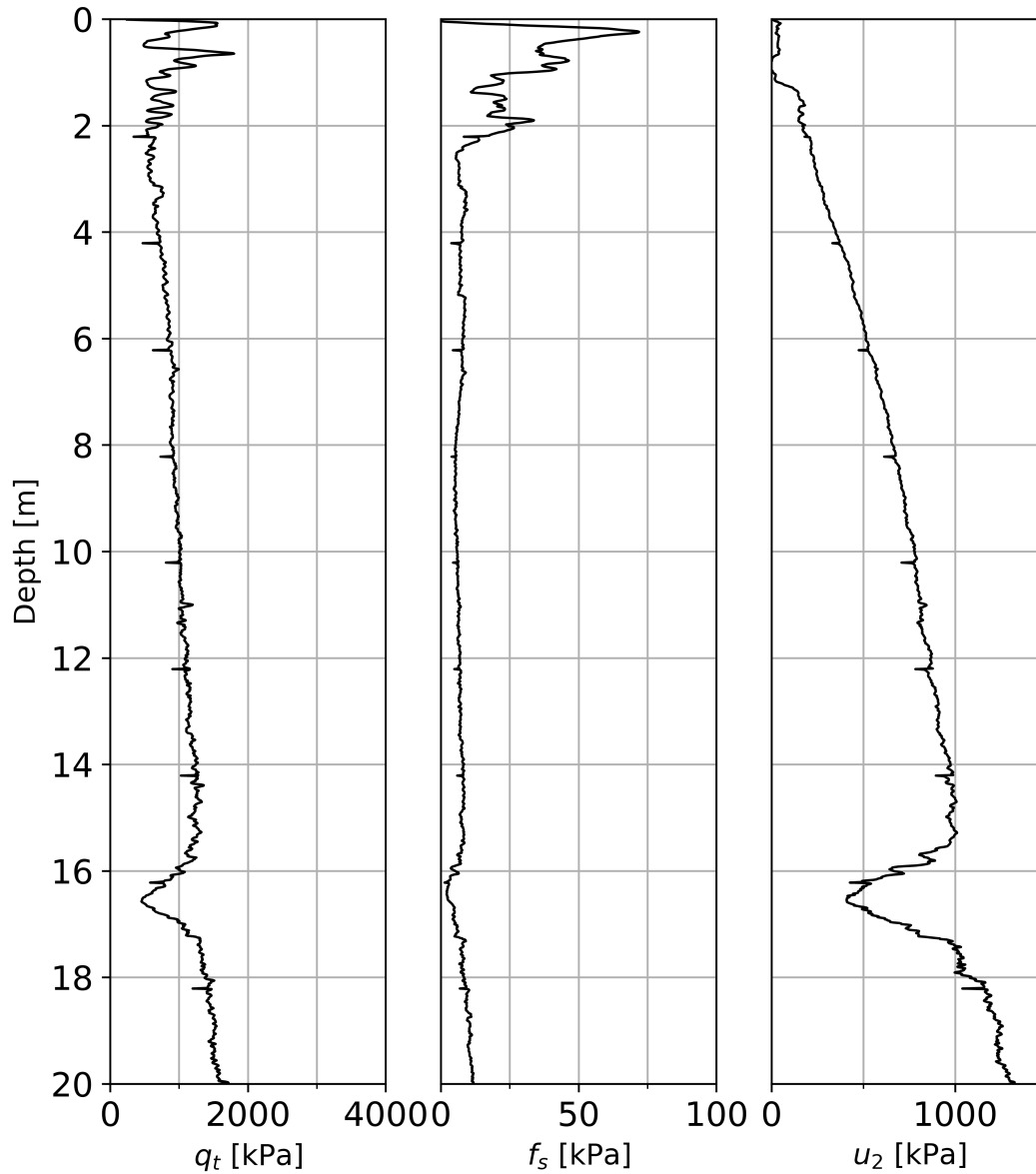


Figure A.17: Raw data graphs of NGTS Tiller-Flotten CPTu 17, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 19

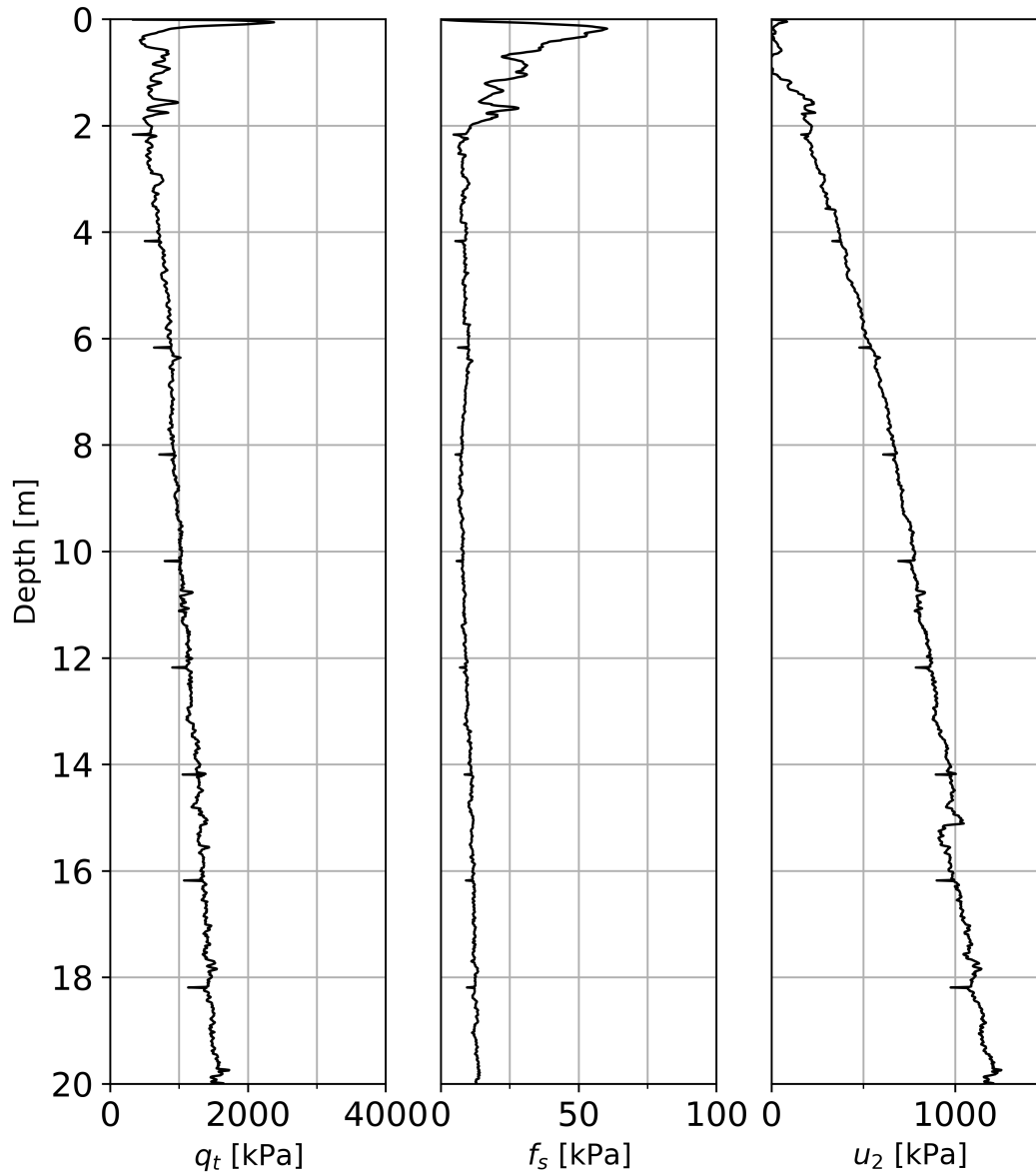


Figure A.18: Raw data graphs of NGTS Tiller-Flotten CPTu 19, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 20

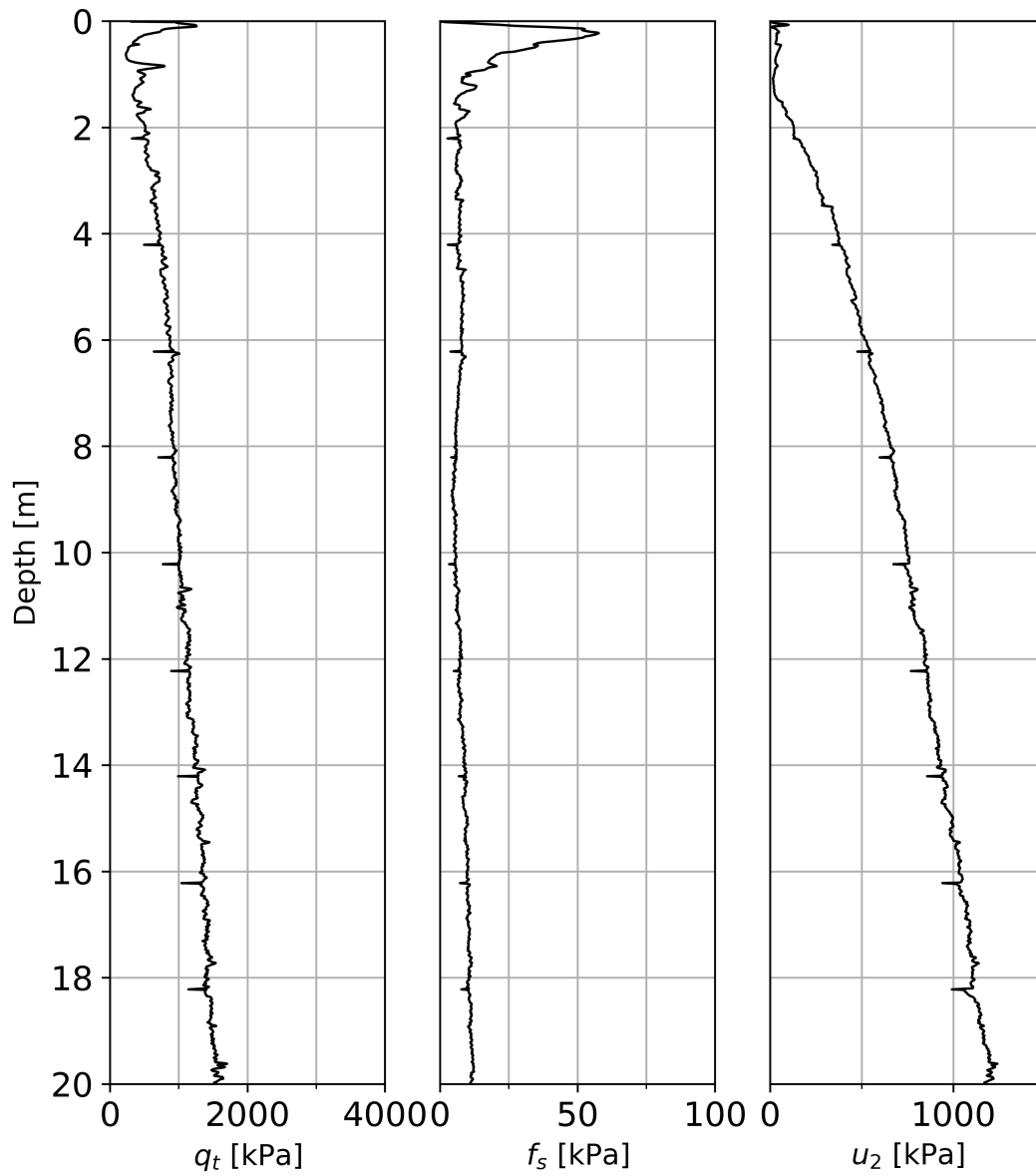


Figure A.19: Raw data graphs of NGTS Tiller-Flotten CPTu 20, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 22

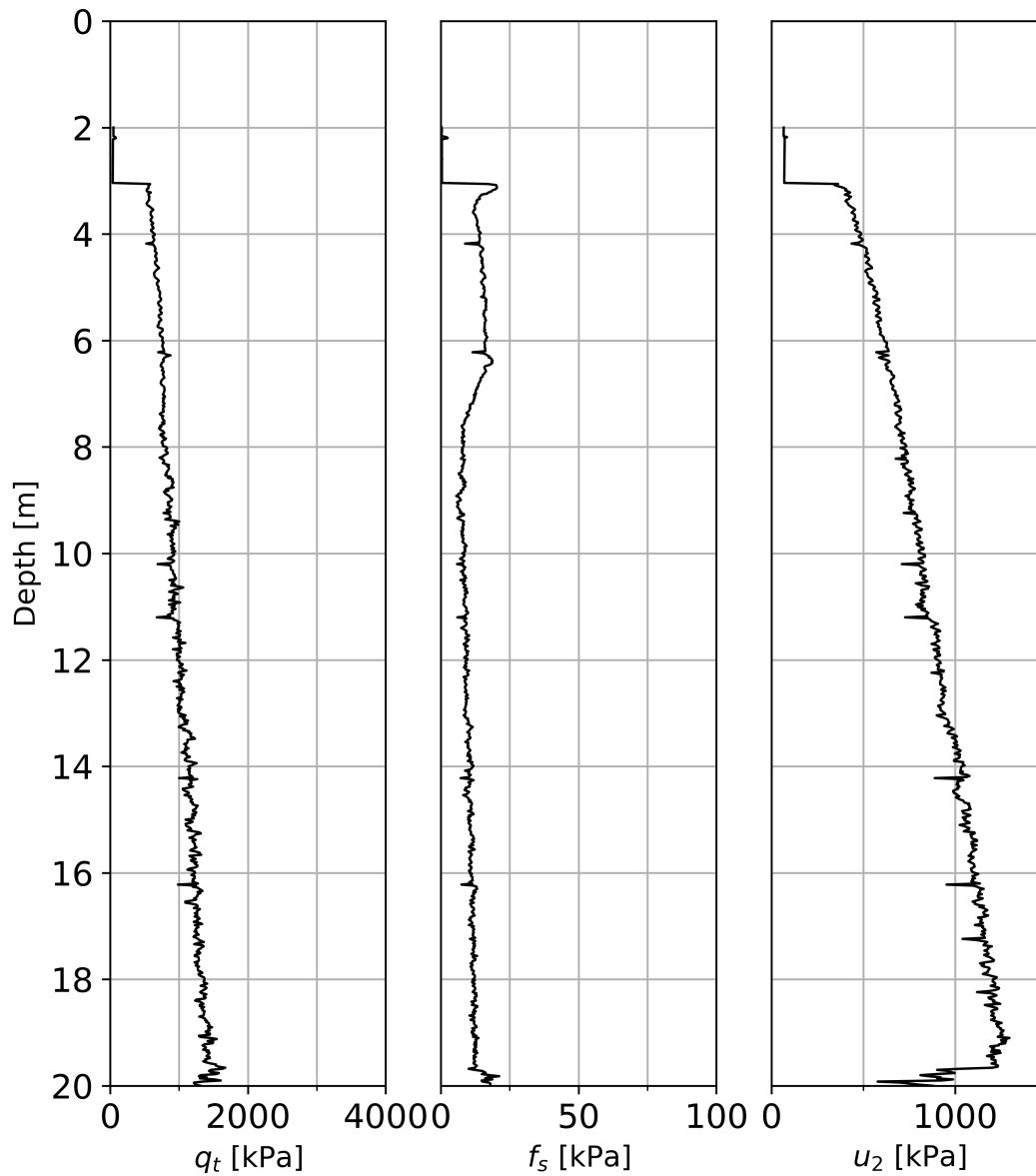


Figure A.20: Raw data graphs of NGTS Tiller-Flotten CPTu 22, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .



## Tiller CPTu 23

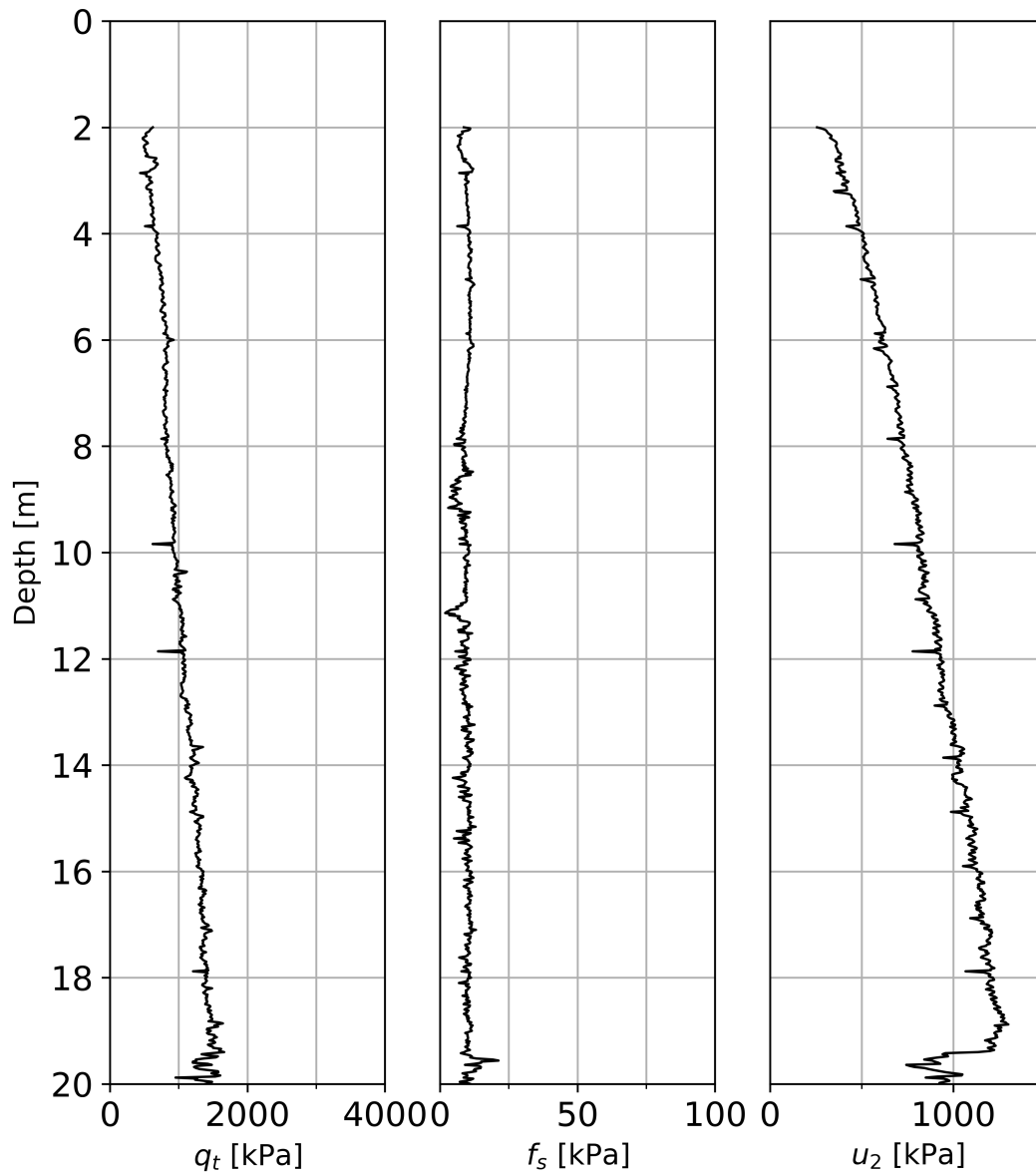


Figure A.21: Raw data graphs of NGTS Tiller-Flotten CPTu 23, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 24

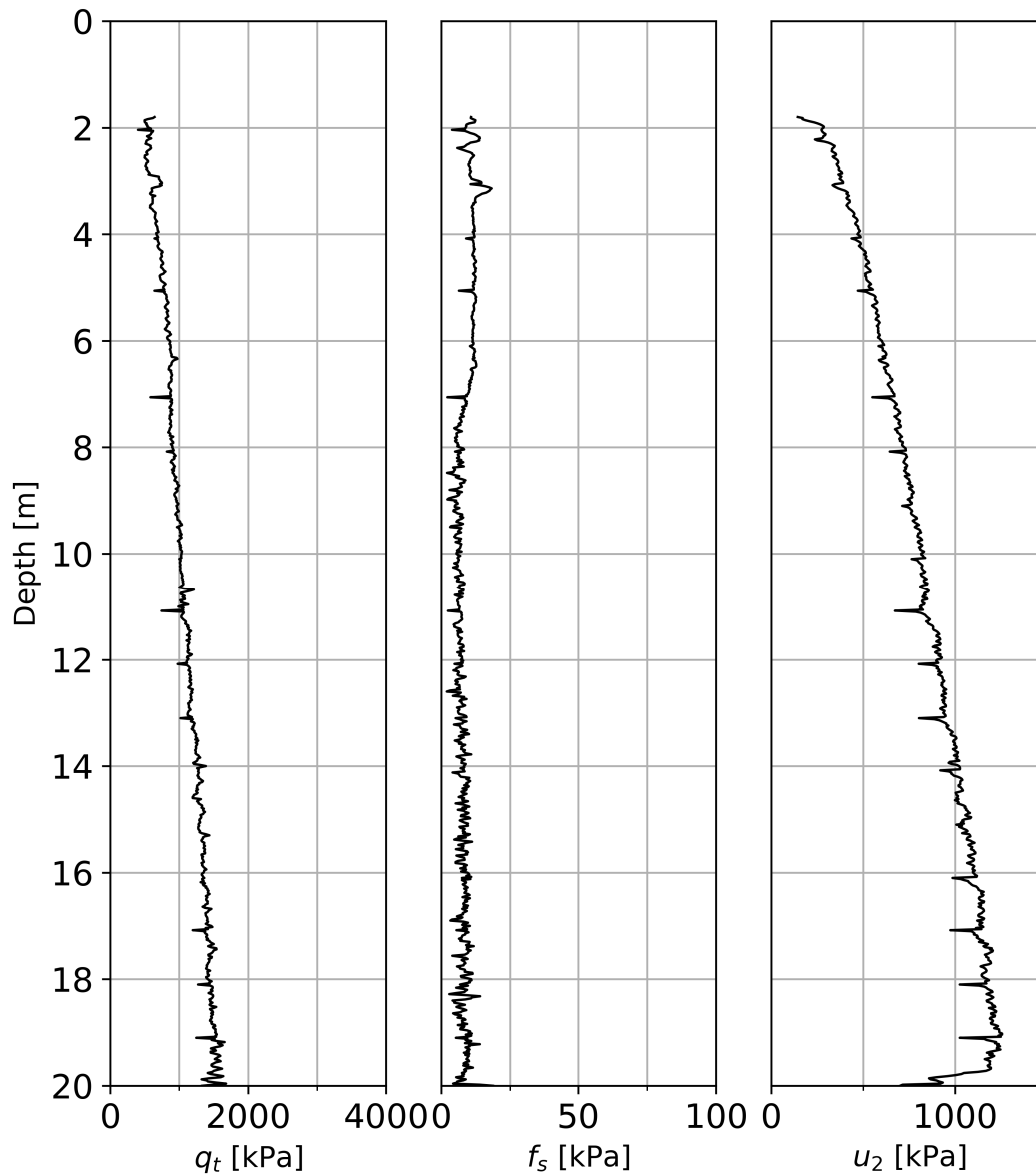


Figure A.22: Raw data graphs of NGTS Tiller-Flotten CPTu 24, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 25

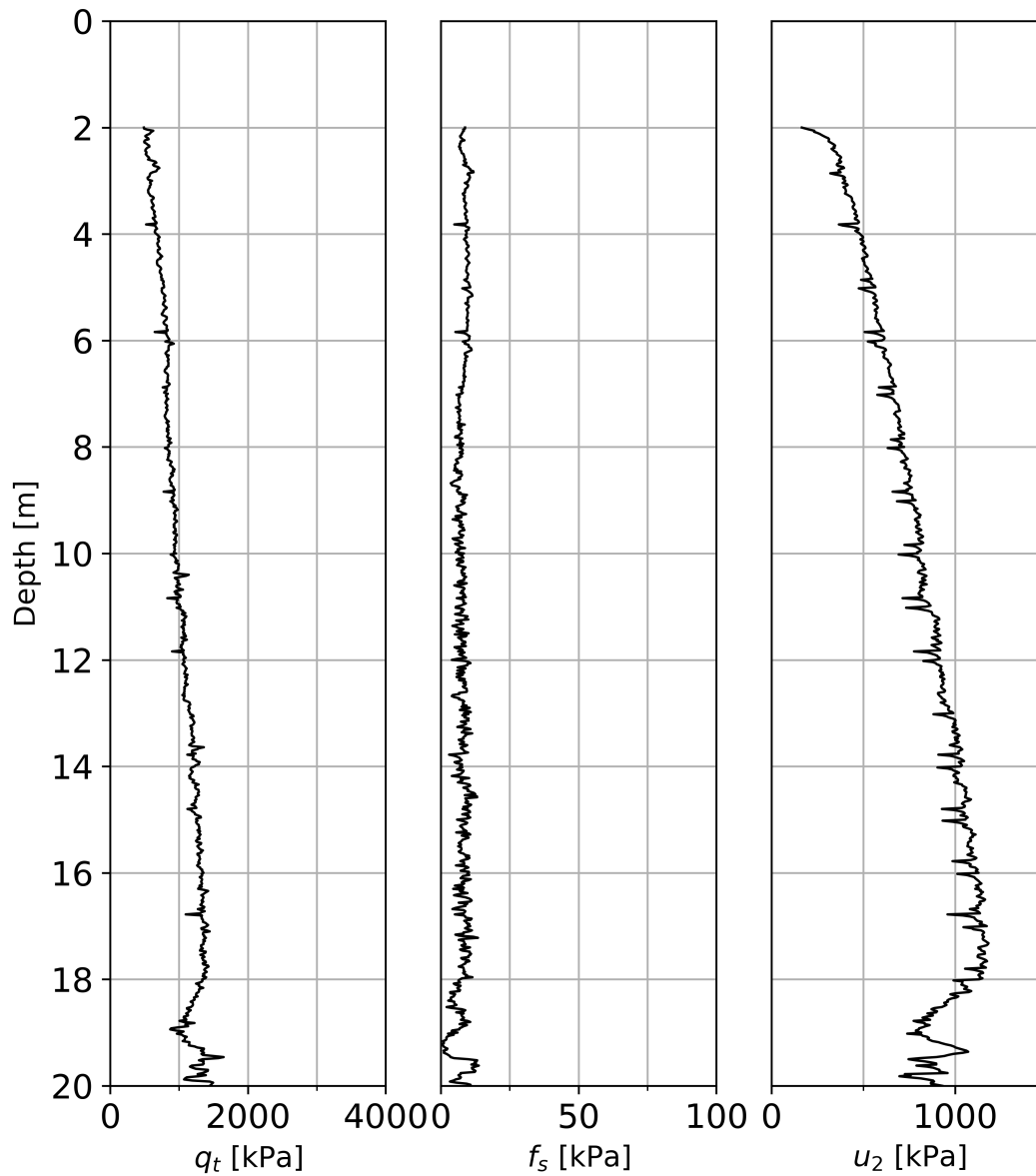


Figure A.23: Raw data graphs of NGTS Tiller-Flotten CPTu 25, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 26

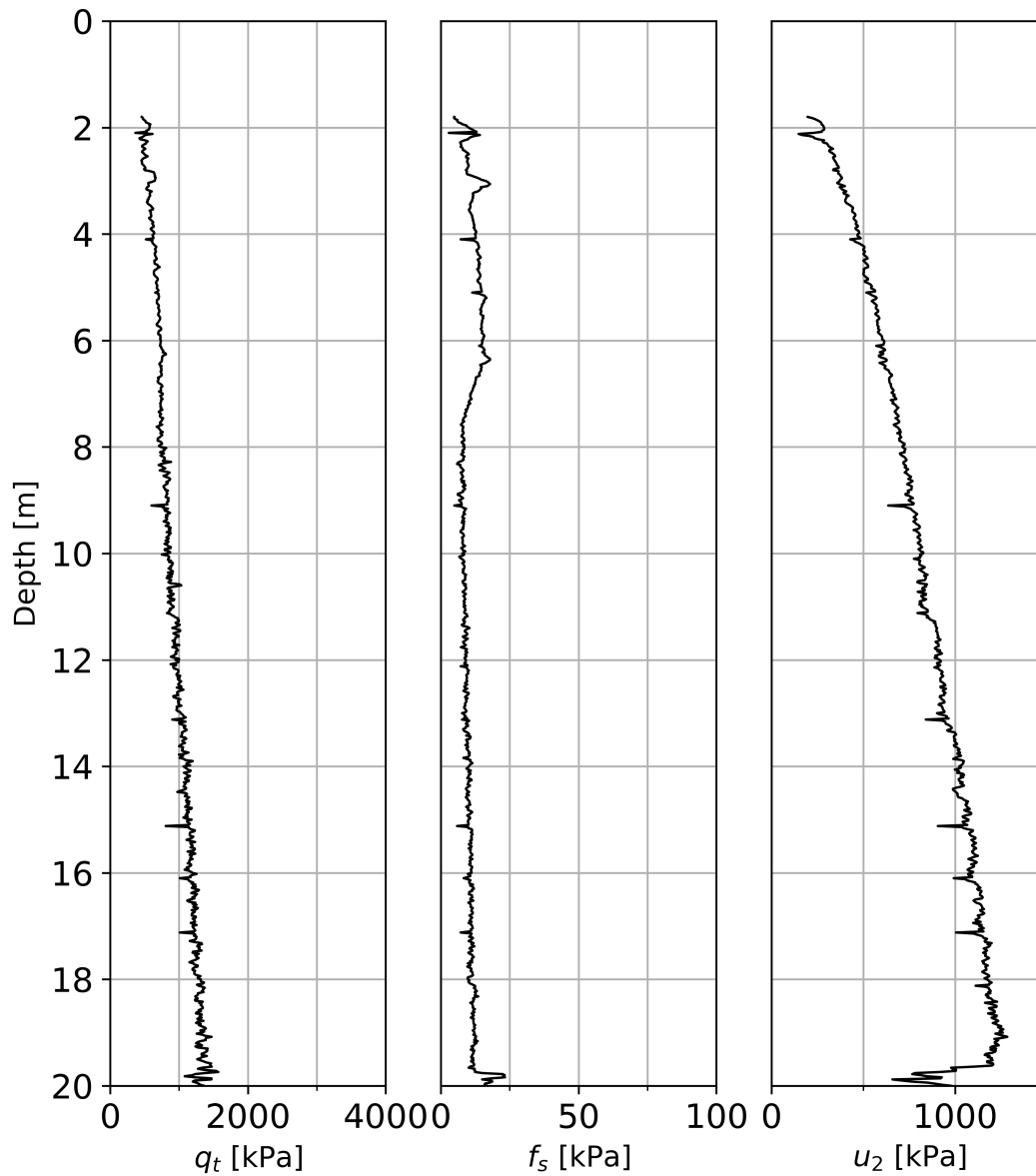


Figure A.24: Raw data graphs of NGTS Tiller-Flotten CPTu 26, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 27

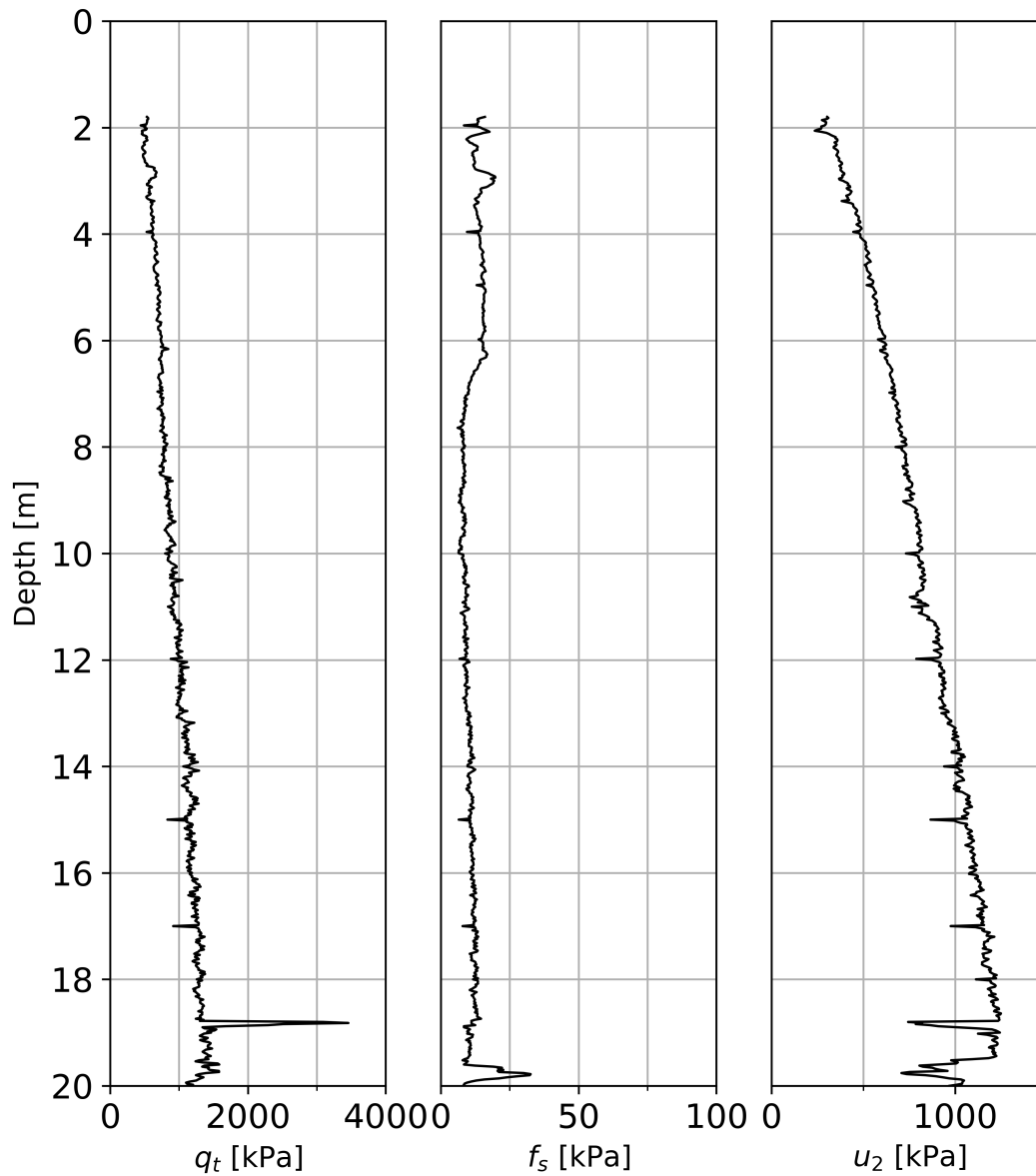


Figure A.25: Raw data graphs of NGTS Tiller-Flotten CPTu 27, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 28

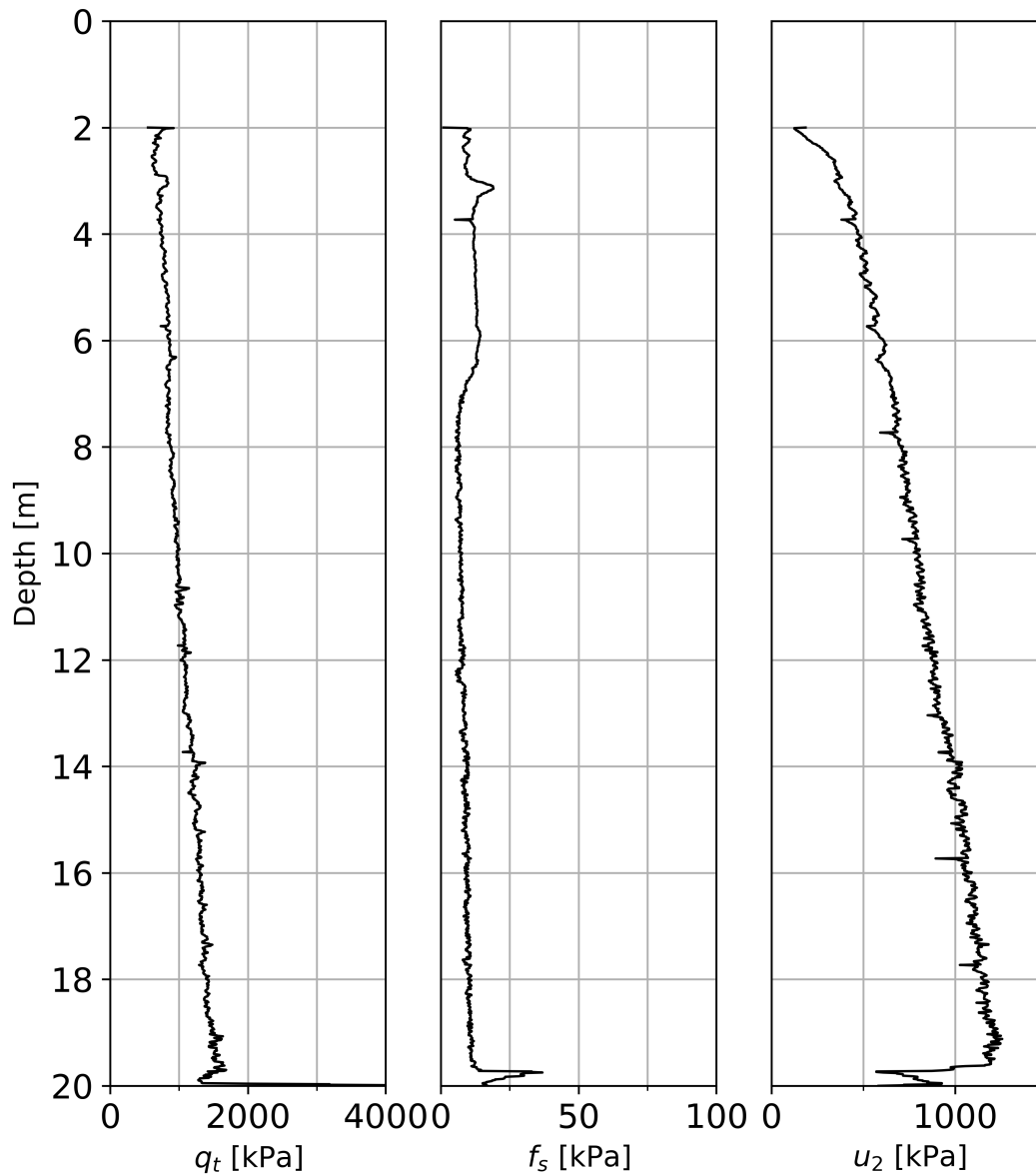


Figure A.26: Raw data graphs of NGTS Tiller-Flotten CPTu 28, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 29

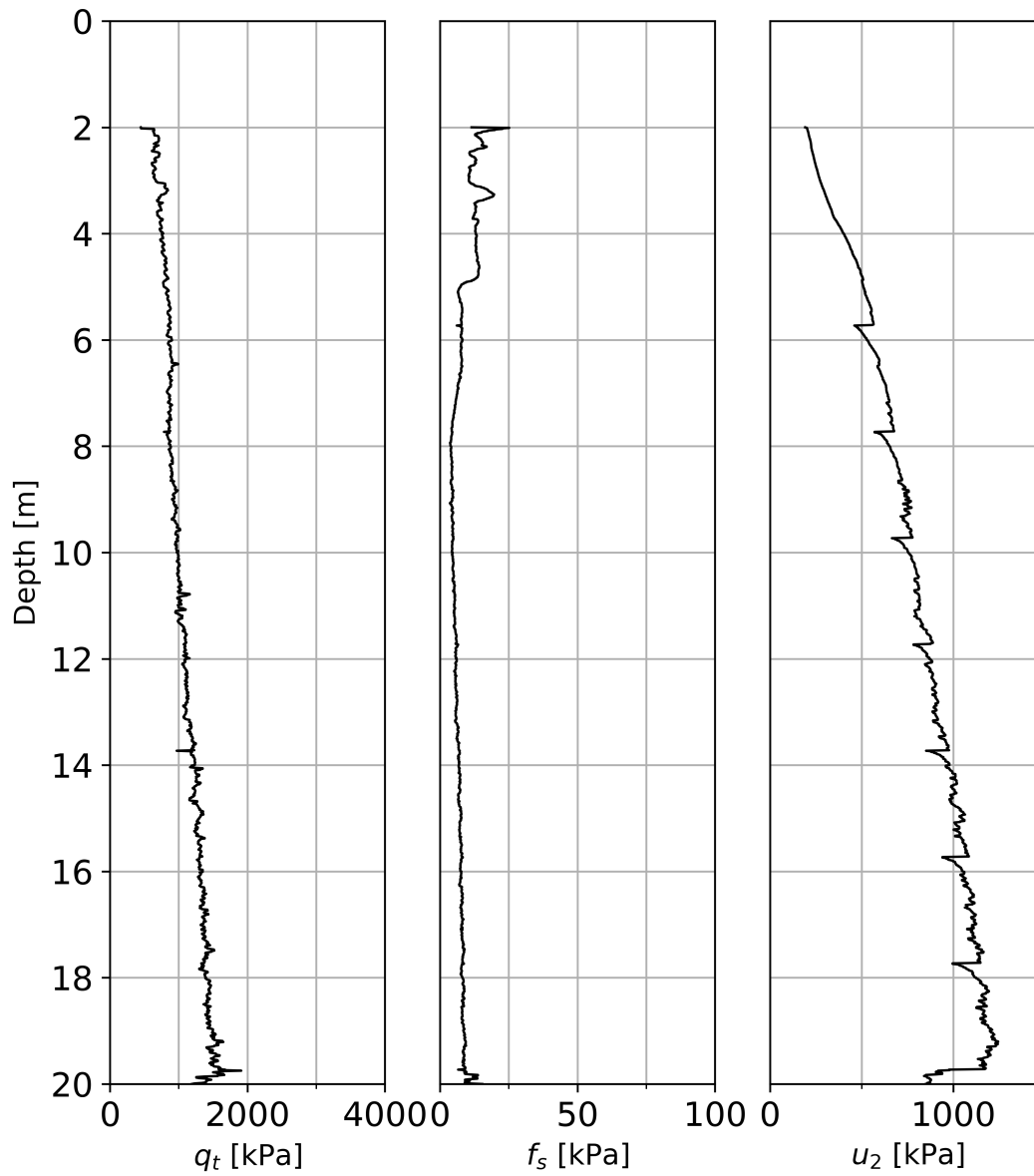


Figure A.27: Raw data graphs of NGTS Tiller-Flotten CPTu 29, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

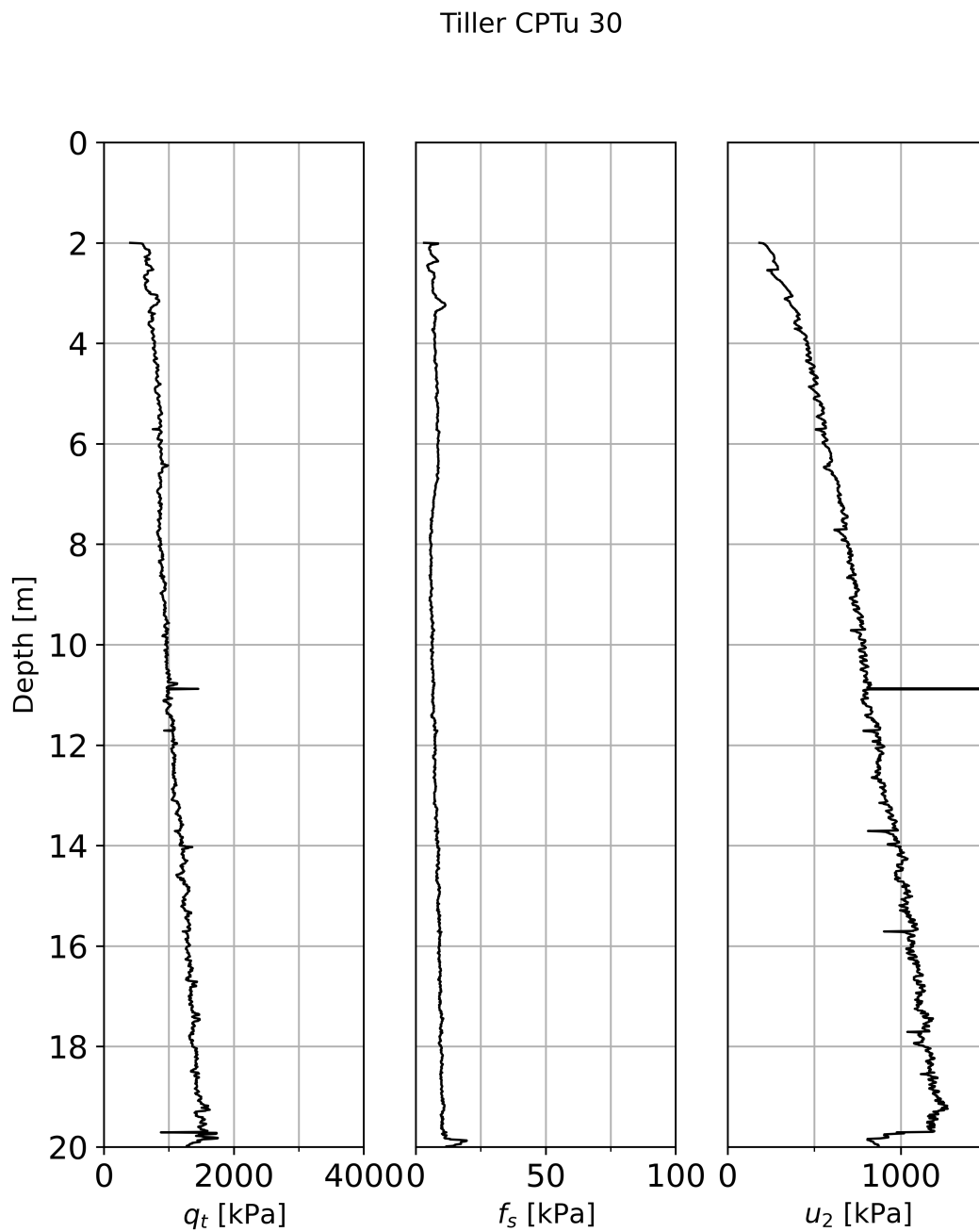


Figure A.28: Raw data graphs of NGTS Tiller-Flotten CPTu 30, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .



## Tiller CPTu 31

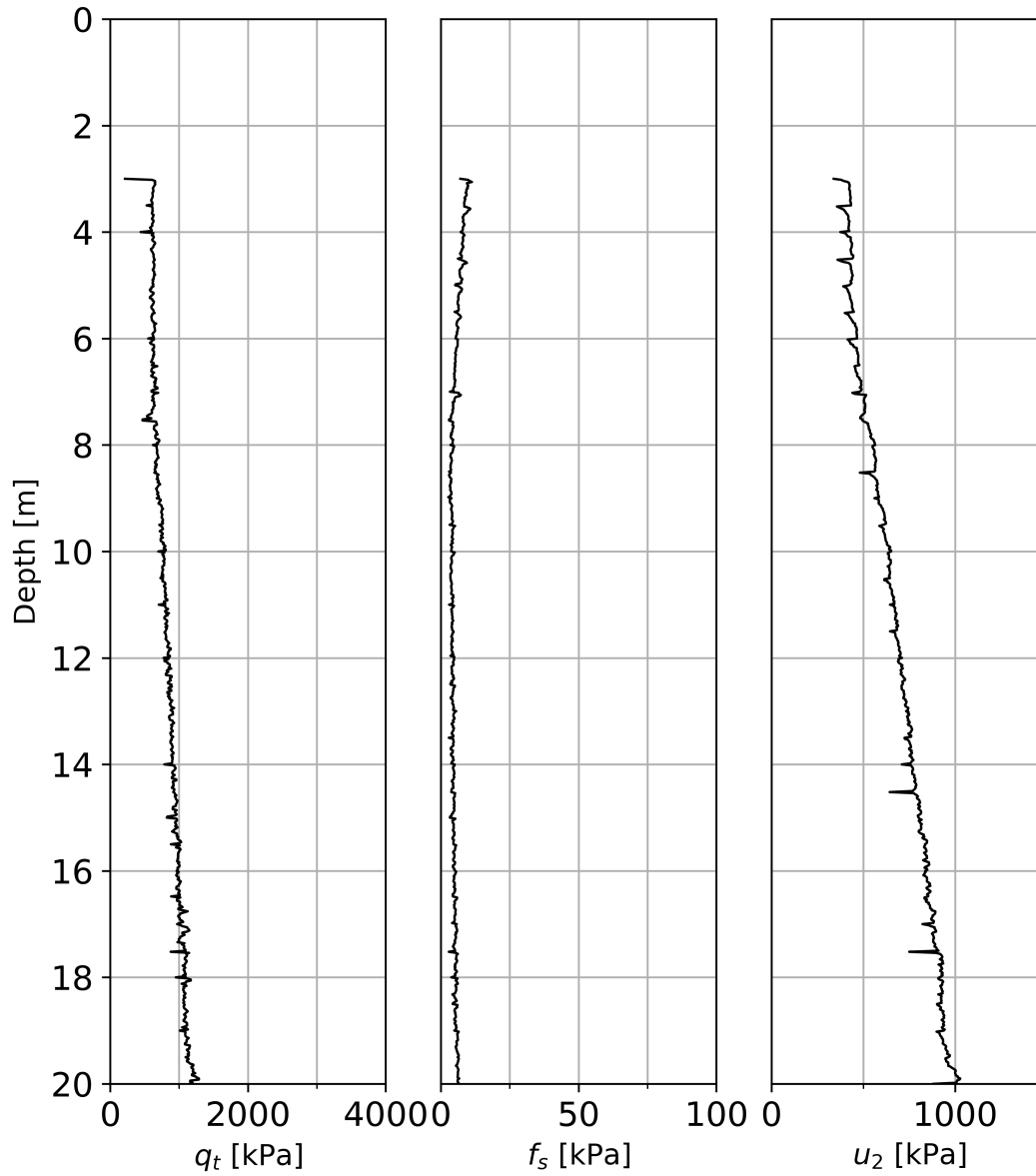


Figure A.29: Raw data graphs of NGTS Tiller-Flotten CPTu 31, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 32

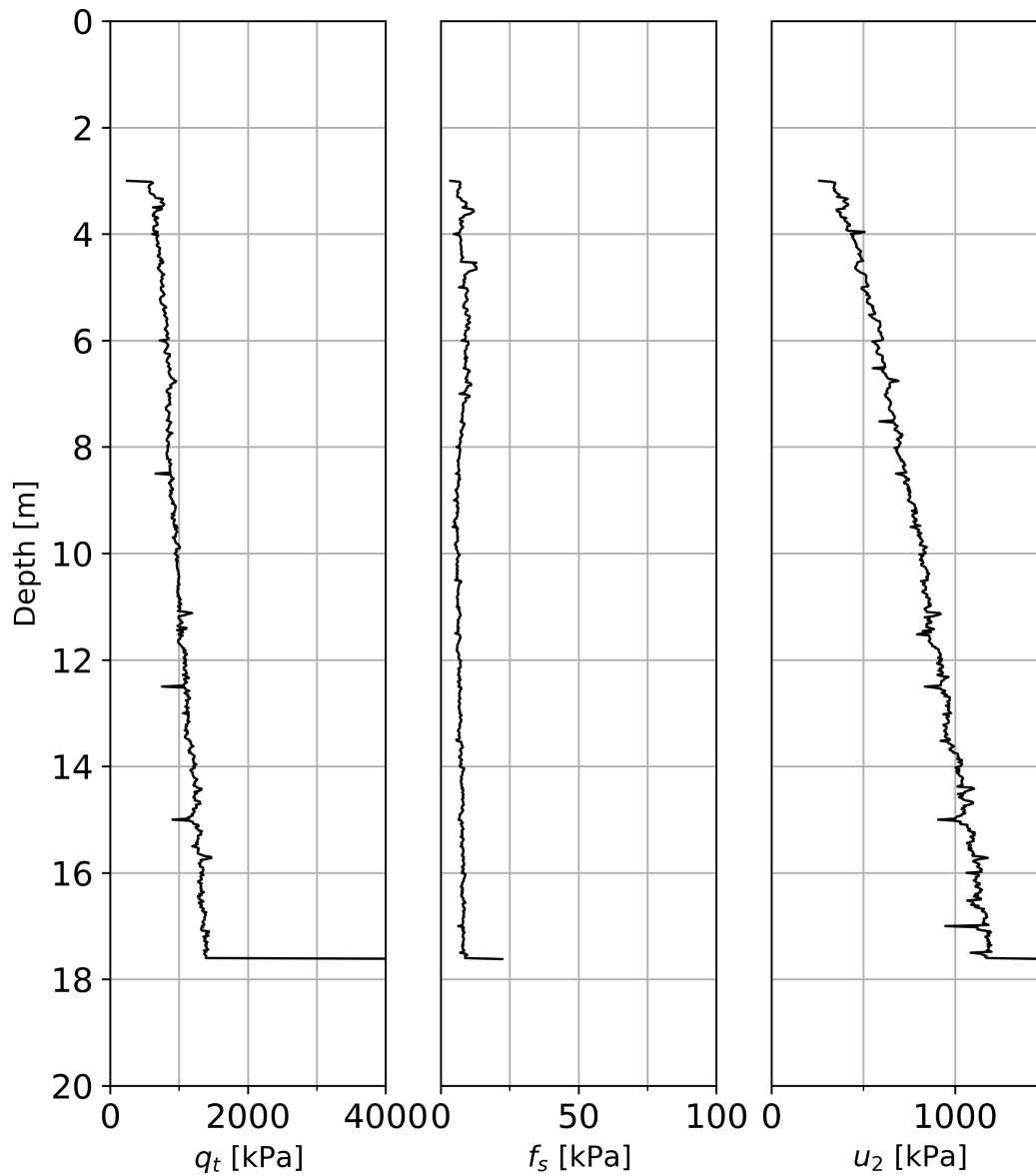


Figure A.30: Raw data graphs of NGTS Tiller-Flotten CPTu 32, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 33

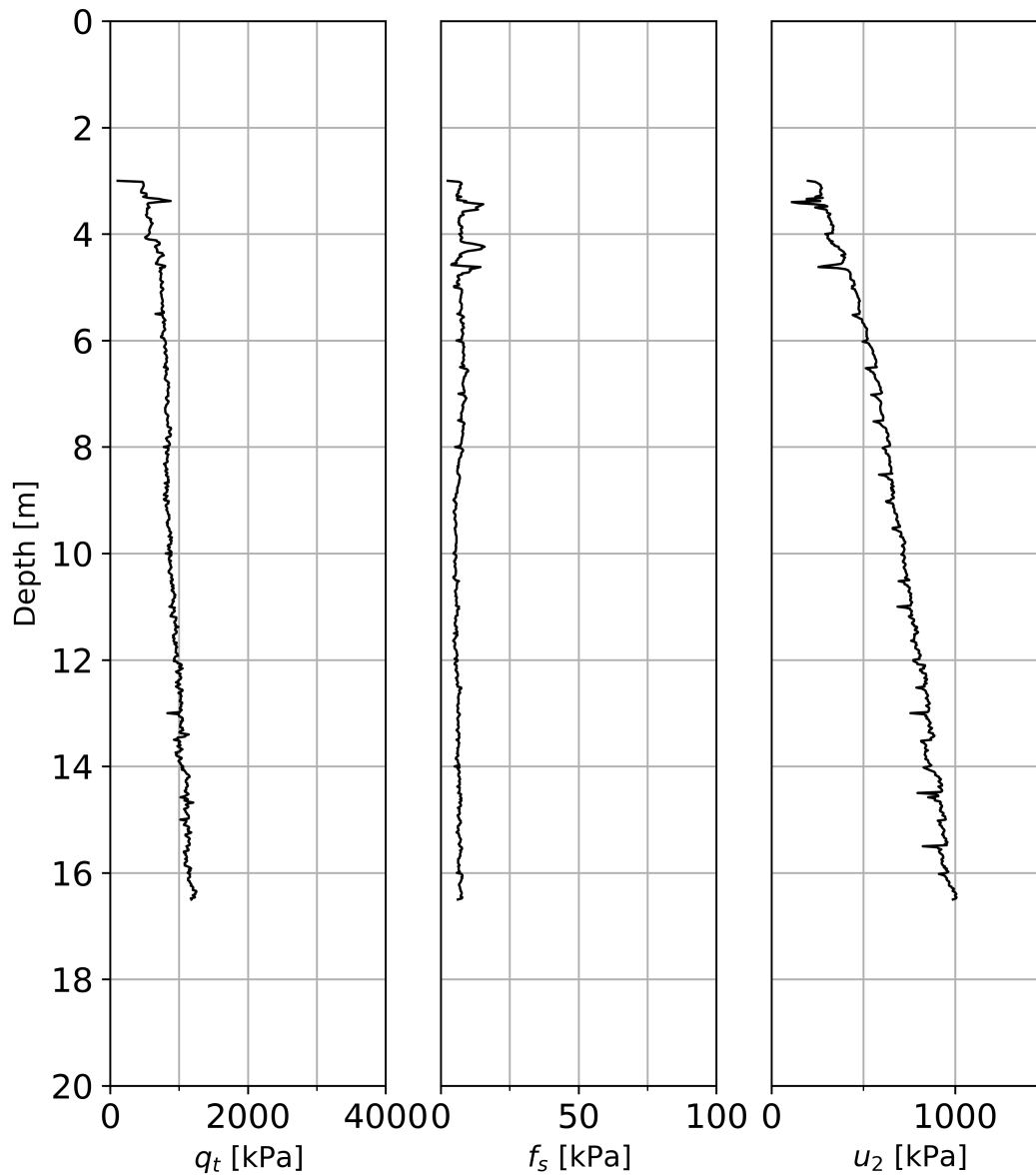


Figure A.31: Raw data graphs of NGTS Tiller-Flotten CPTu 33, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Tiller CPTu 34

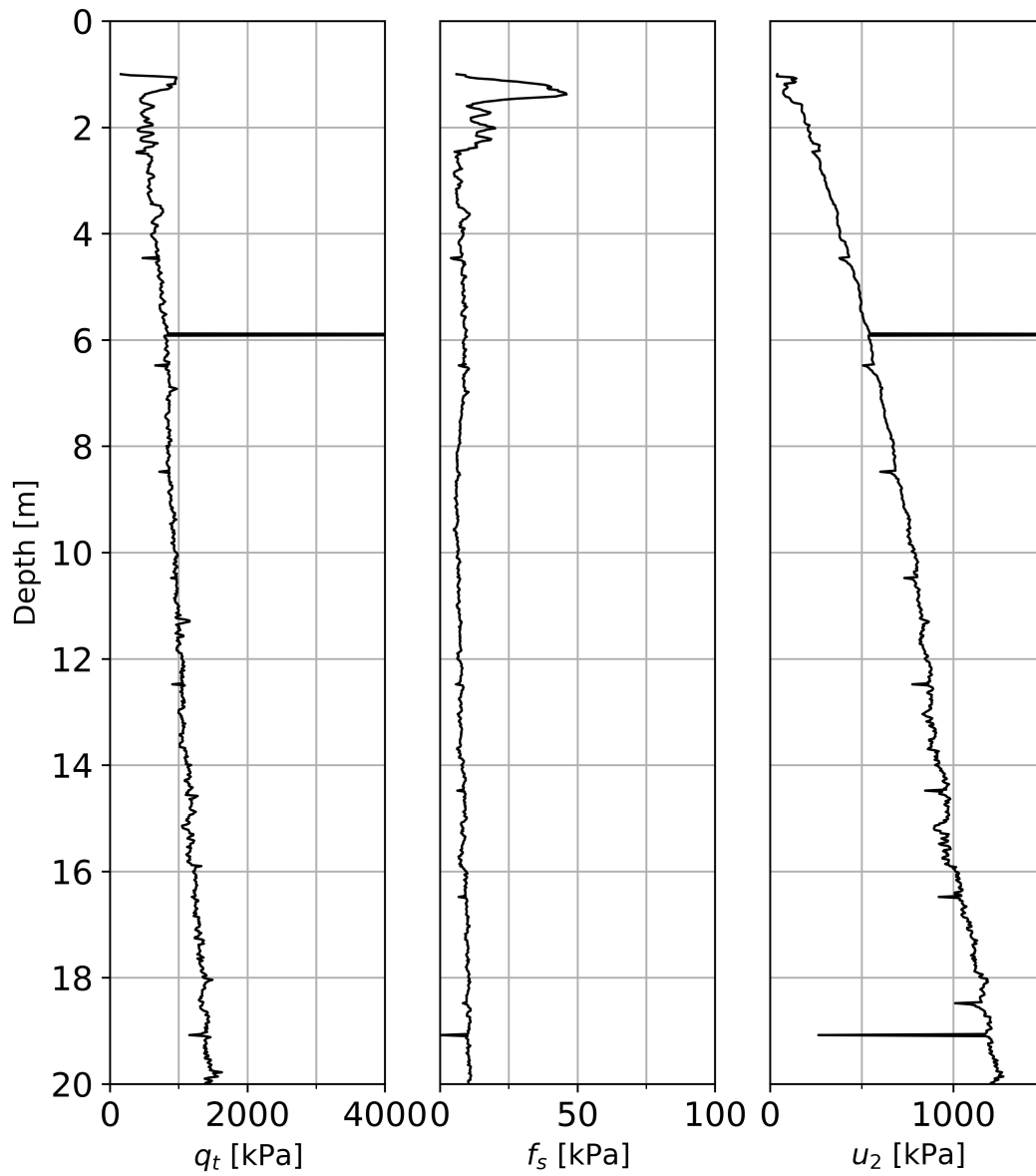


Figure A.32: Raw data graphs of NGTS Tiller-Flotten CPTu 34, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

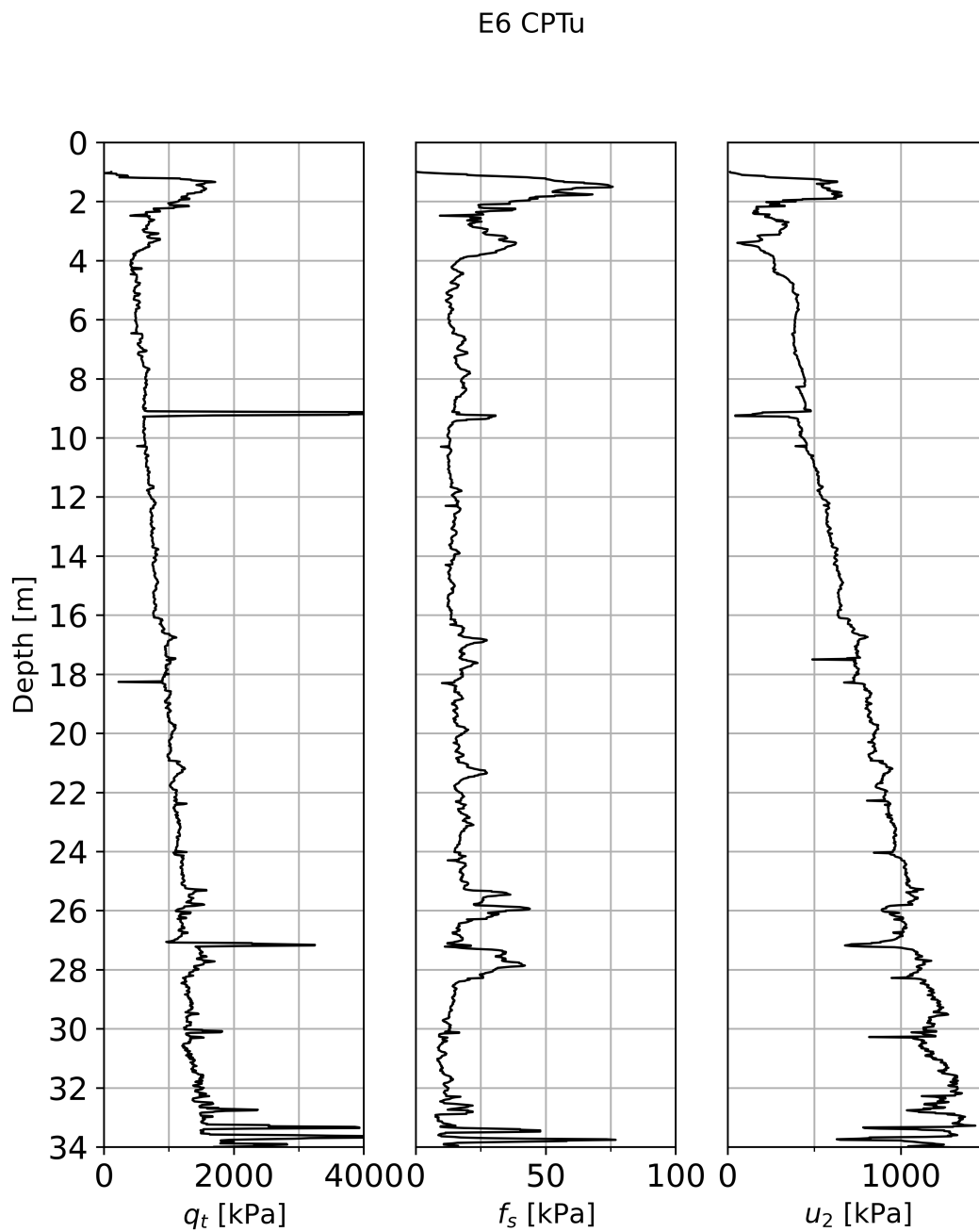


Figure A.33: Raw data graphs of E6 Kvithammer-Åsen CPTu, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

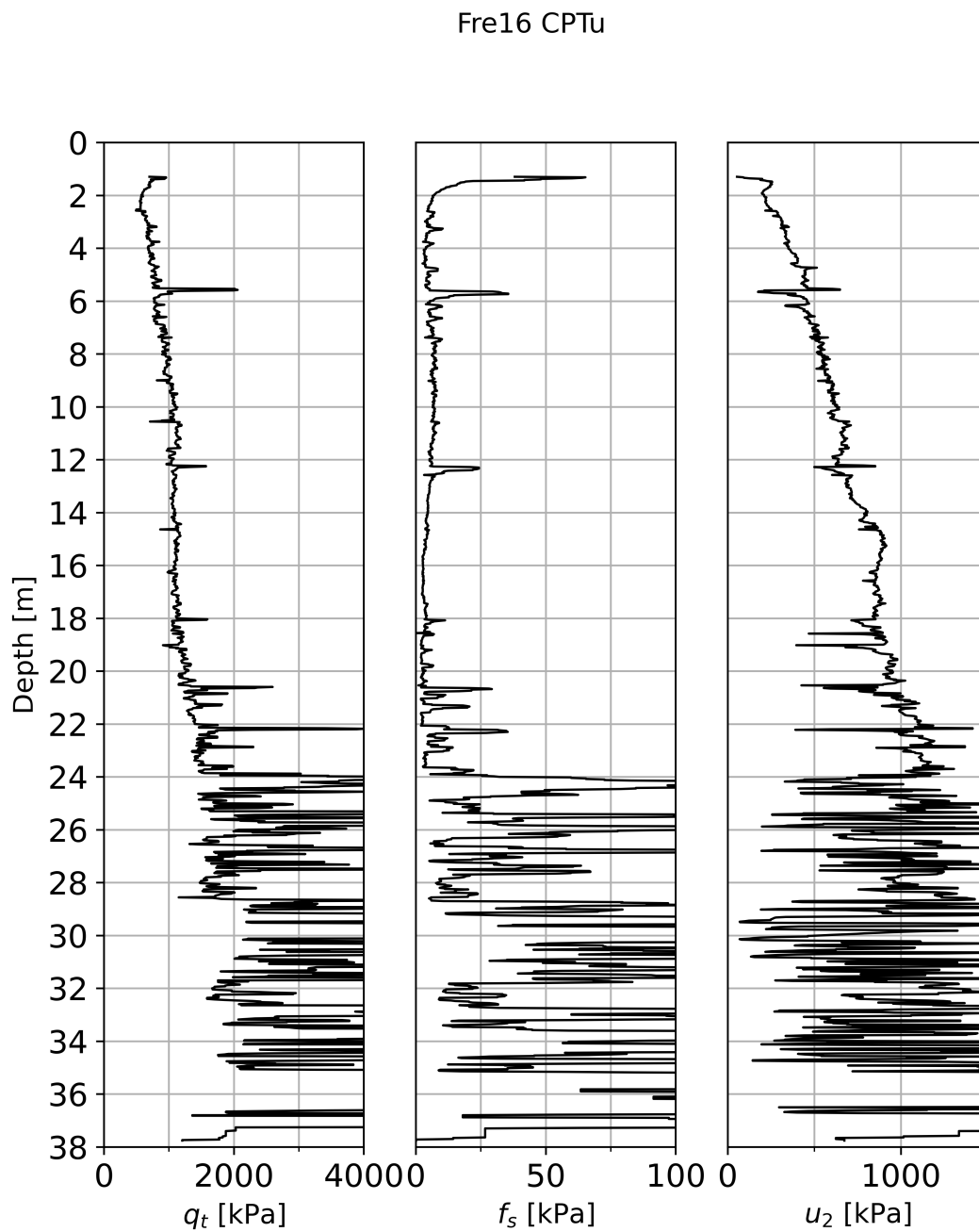


Figure A.34: Raw data graphs of Fre16 (Ringeriksbanen and E16 - the joint railway & road project) CPTu, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

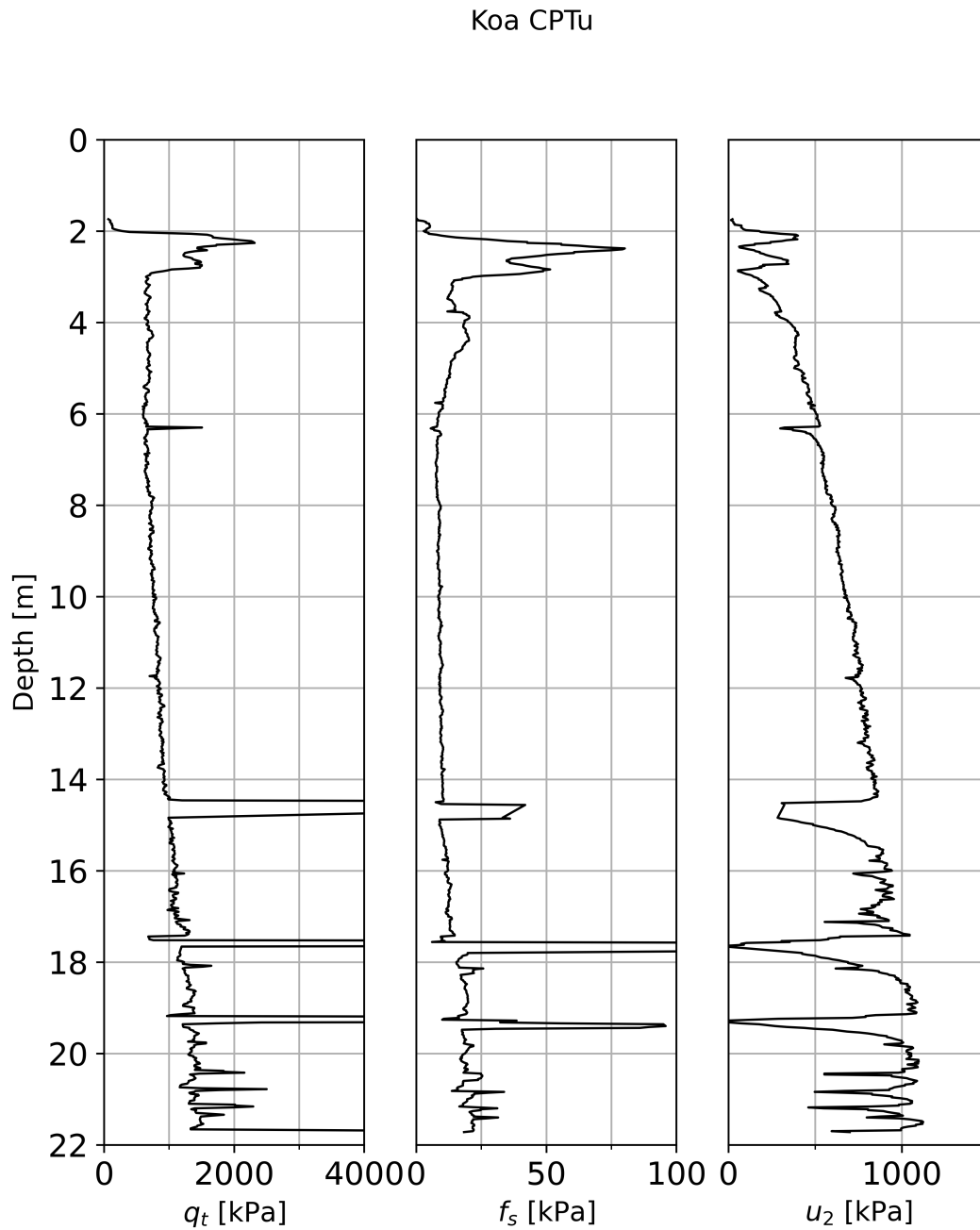


Figure A.35: Raw data graphs of Koa CPTu, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Nybakk-slomarka CPTu

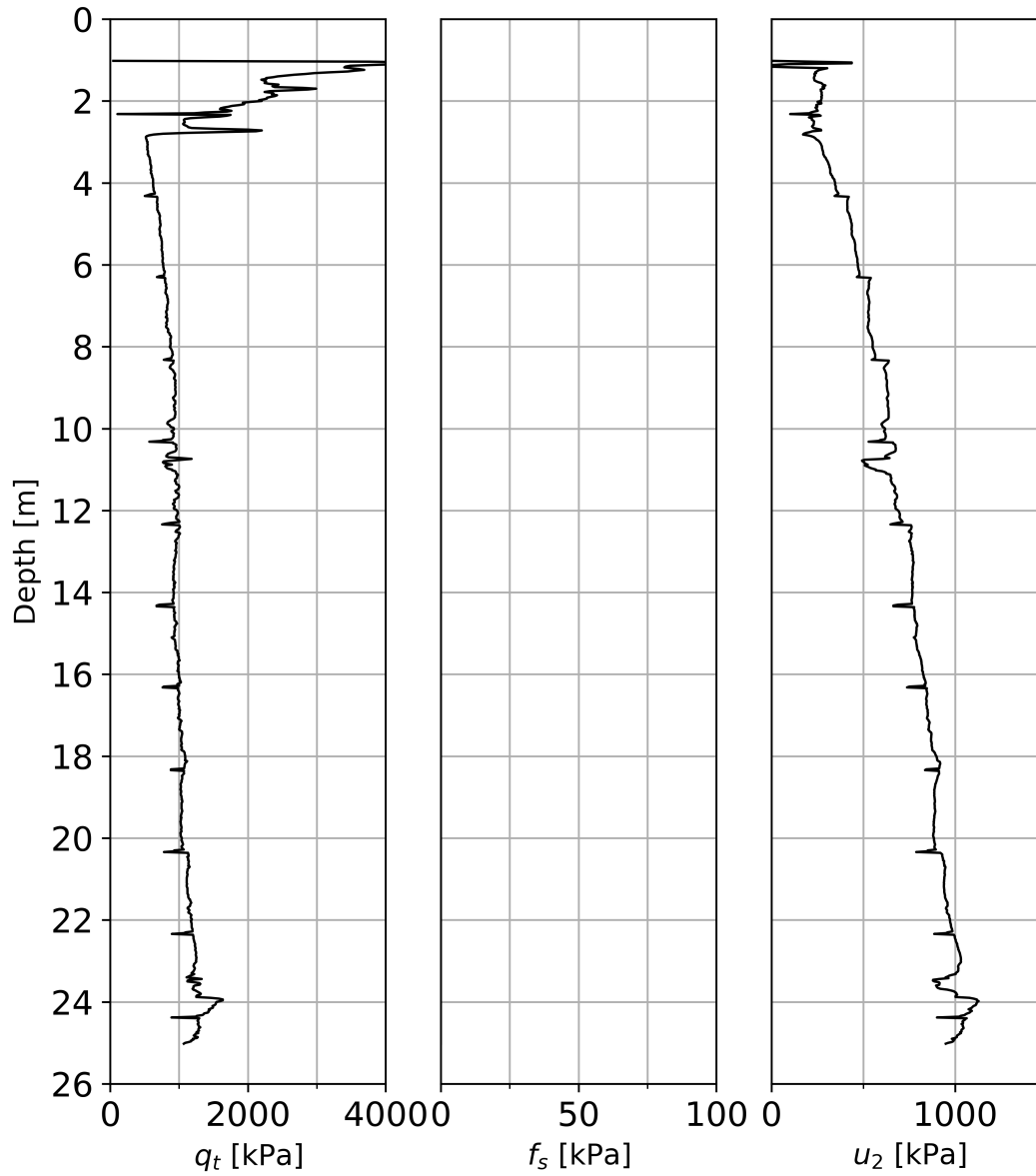


Figure A.36: Raw data graphs of Nybakk-Slomarka, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ . No sleeve friction data was registered from this CPTu.



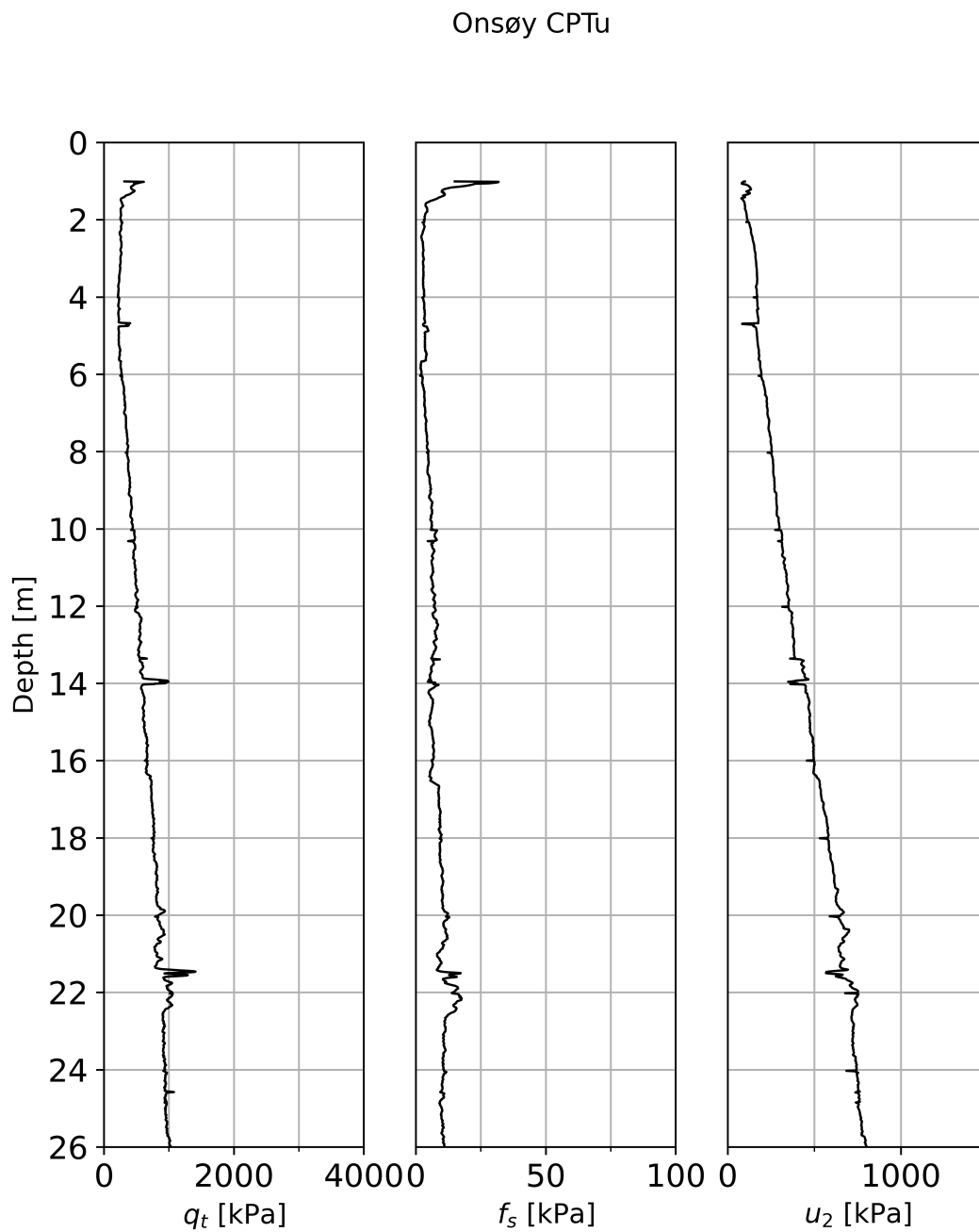


Figure A.37: Raw data graphs of Onsøy CPTu, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

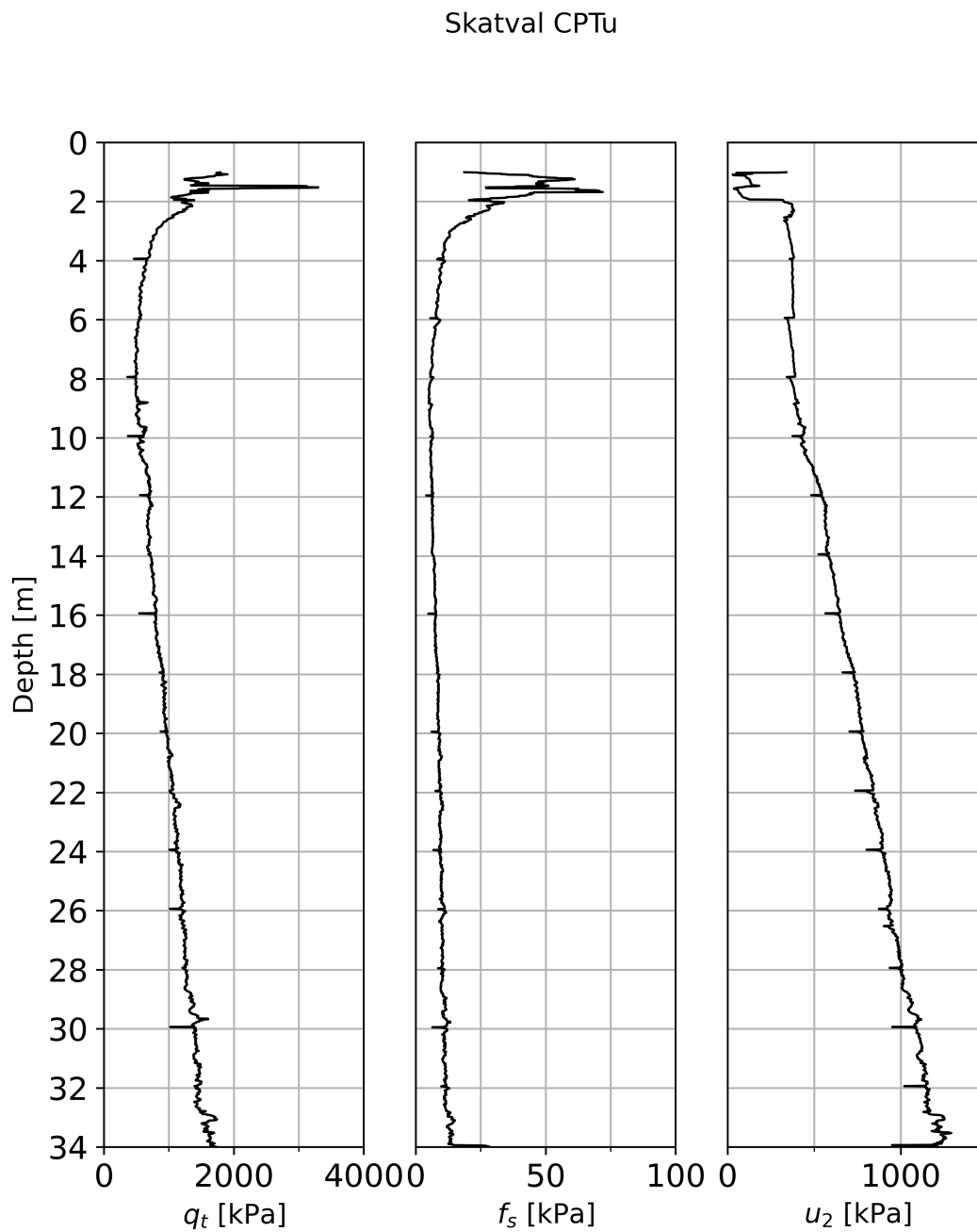


Figure A.38: Raw data graphs of Skatval CPTu, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Saksvik CPTu 2

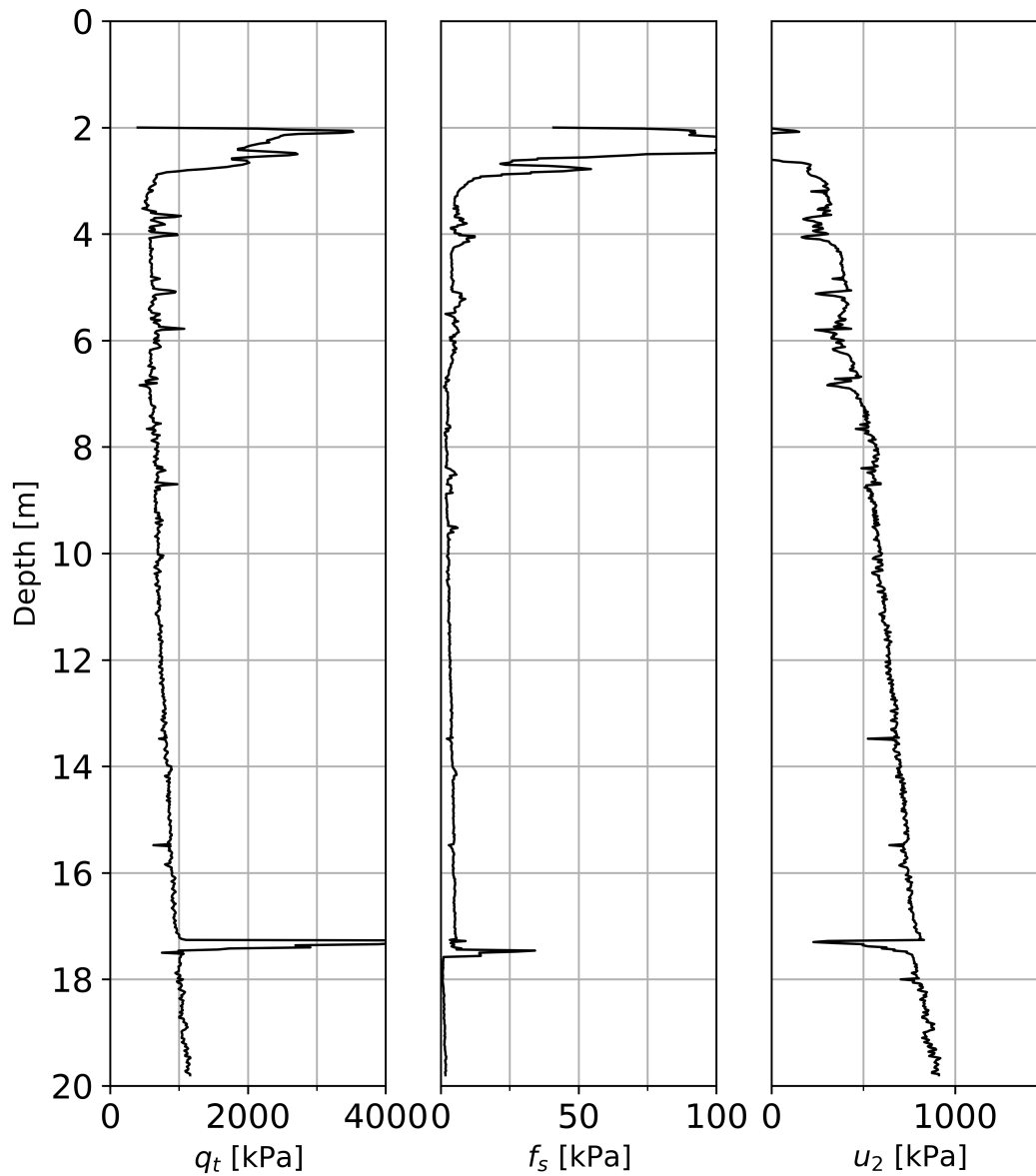


Figure A.39: Raw data graphs of Saksvik CPTu 2, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Saksvik CPTu 5r

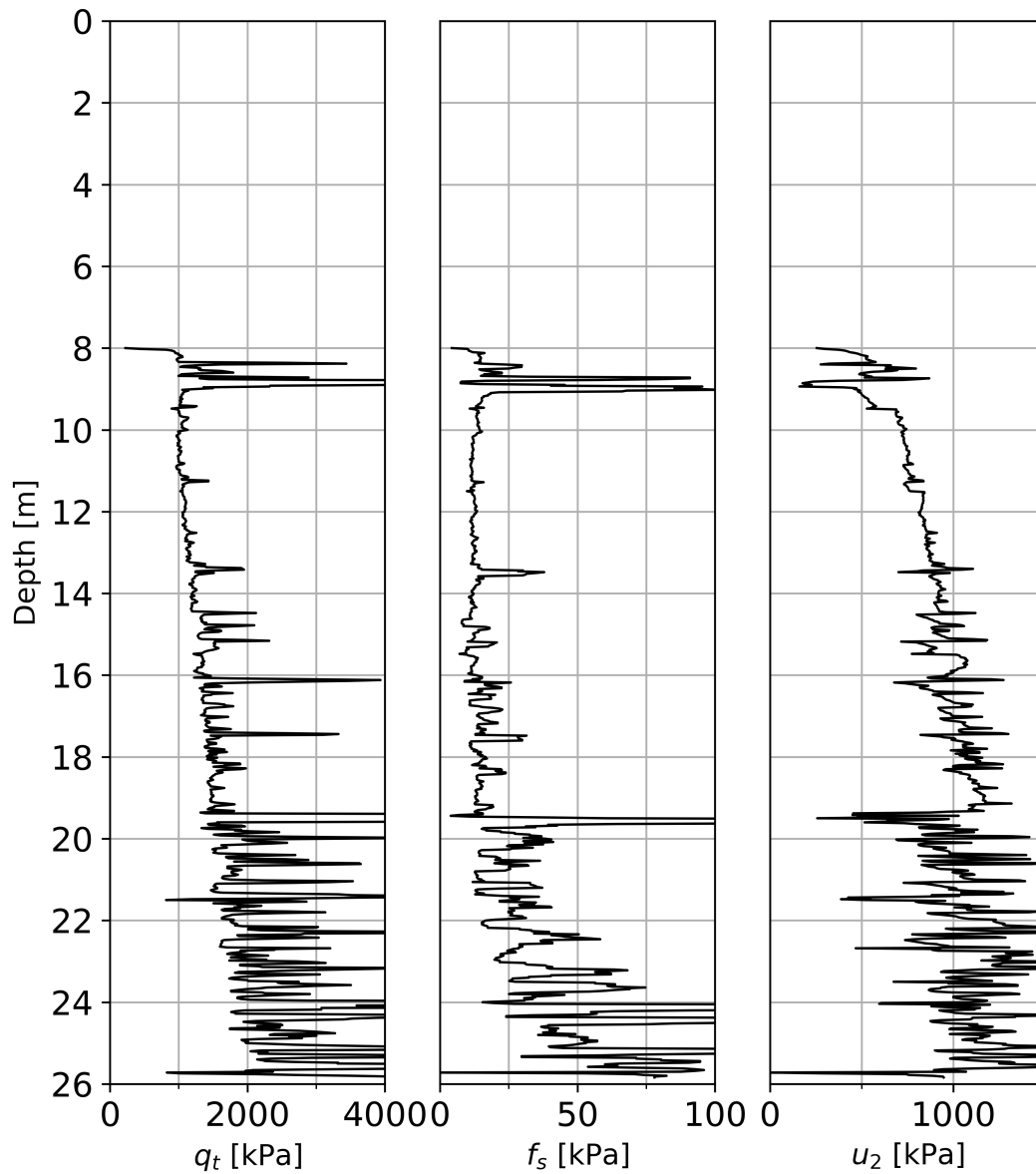


Figure A.40: Raw data graphs of Saksvik CPTu 5R, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Saksvik CPTu 7r

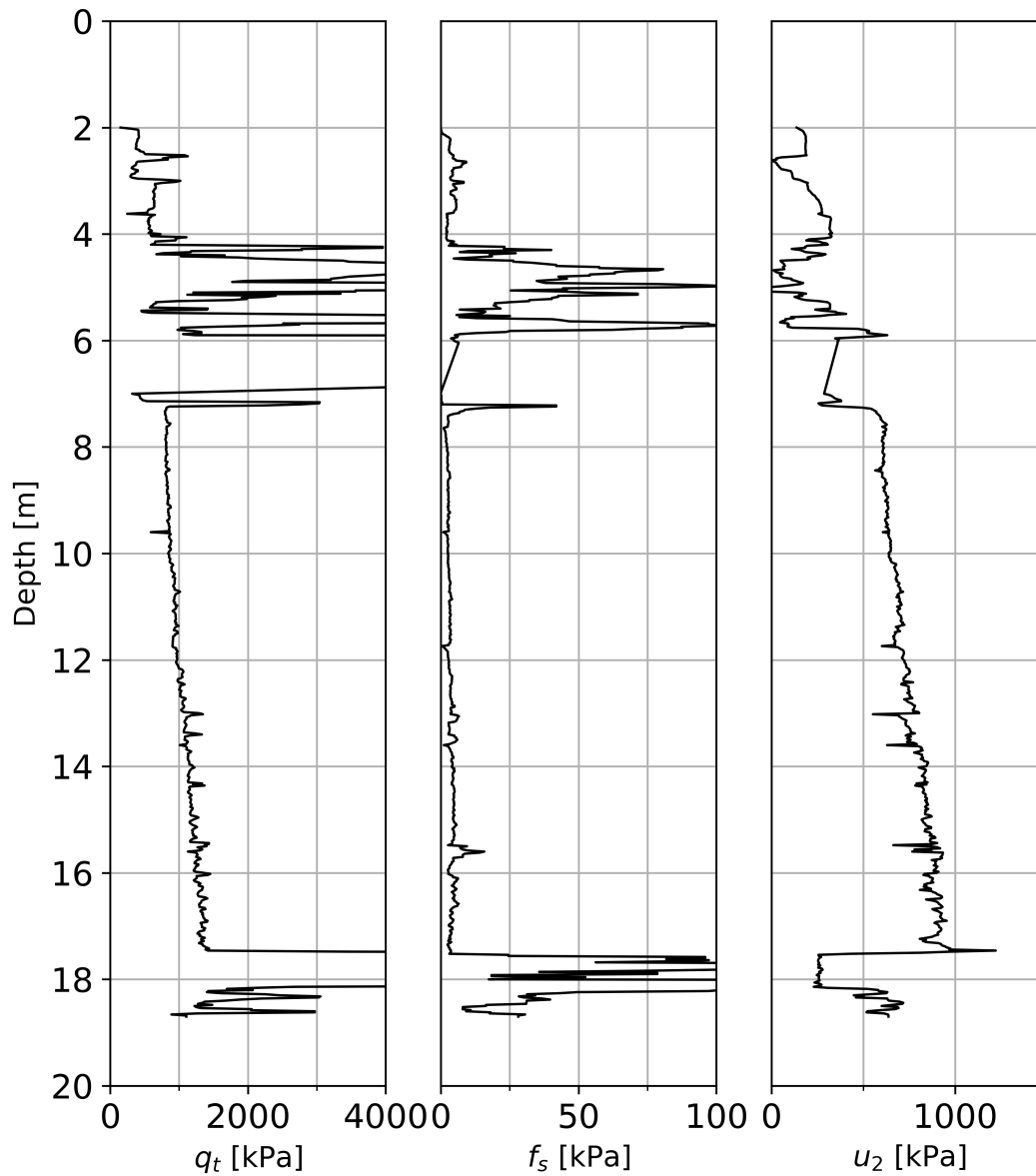


Figure A.41: Raw data graphs of Saksvik CPTu 7R, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Saksvik CPTu 8r

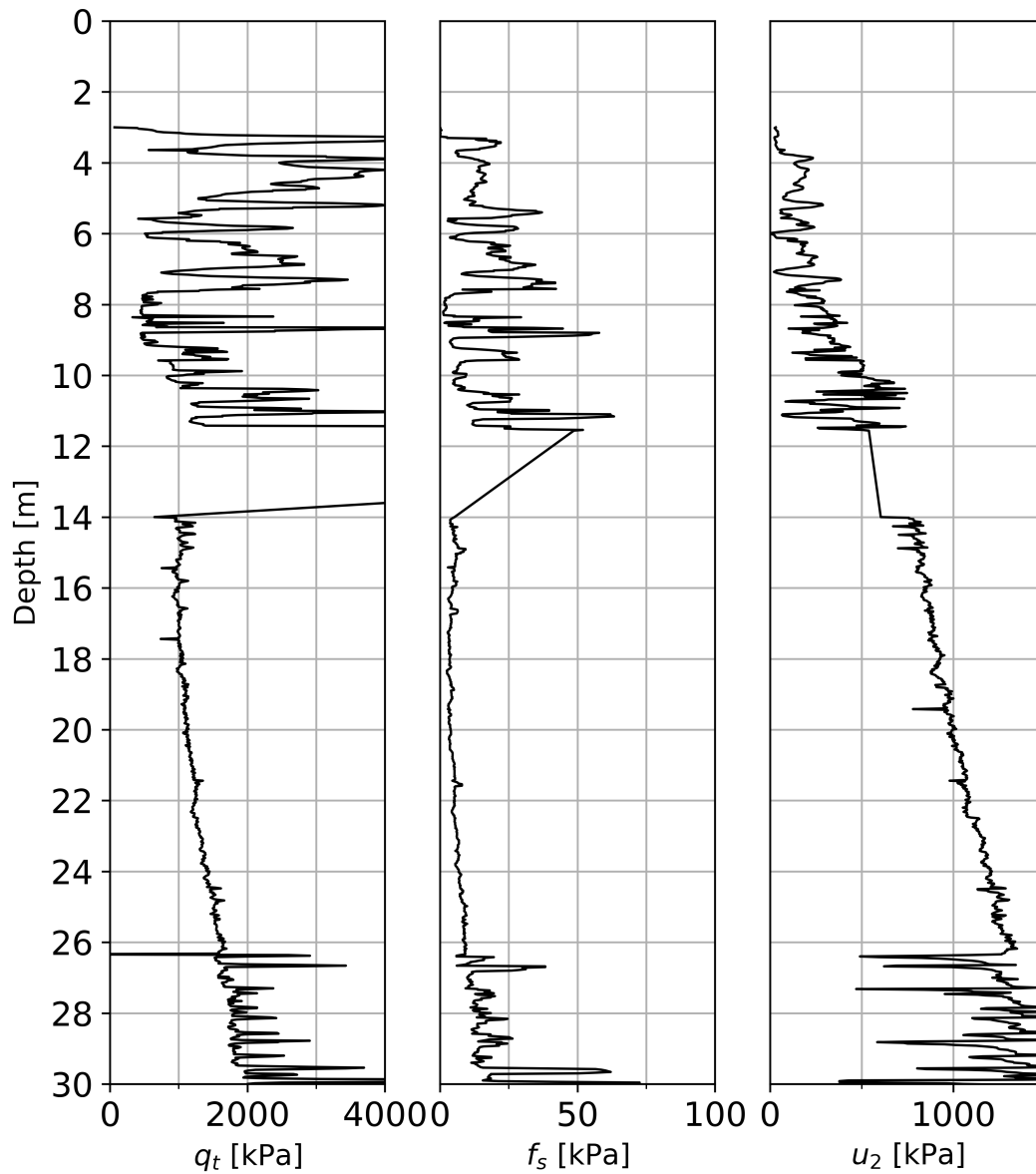


Figure A.42: Raw data graphs of Saksvik CPTu 8R, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Saksvik CPTu 9

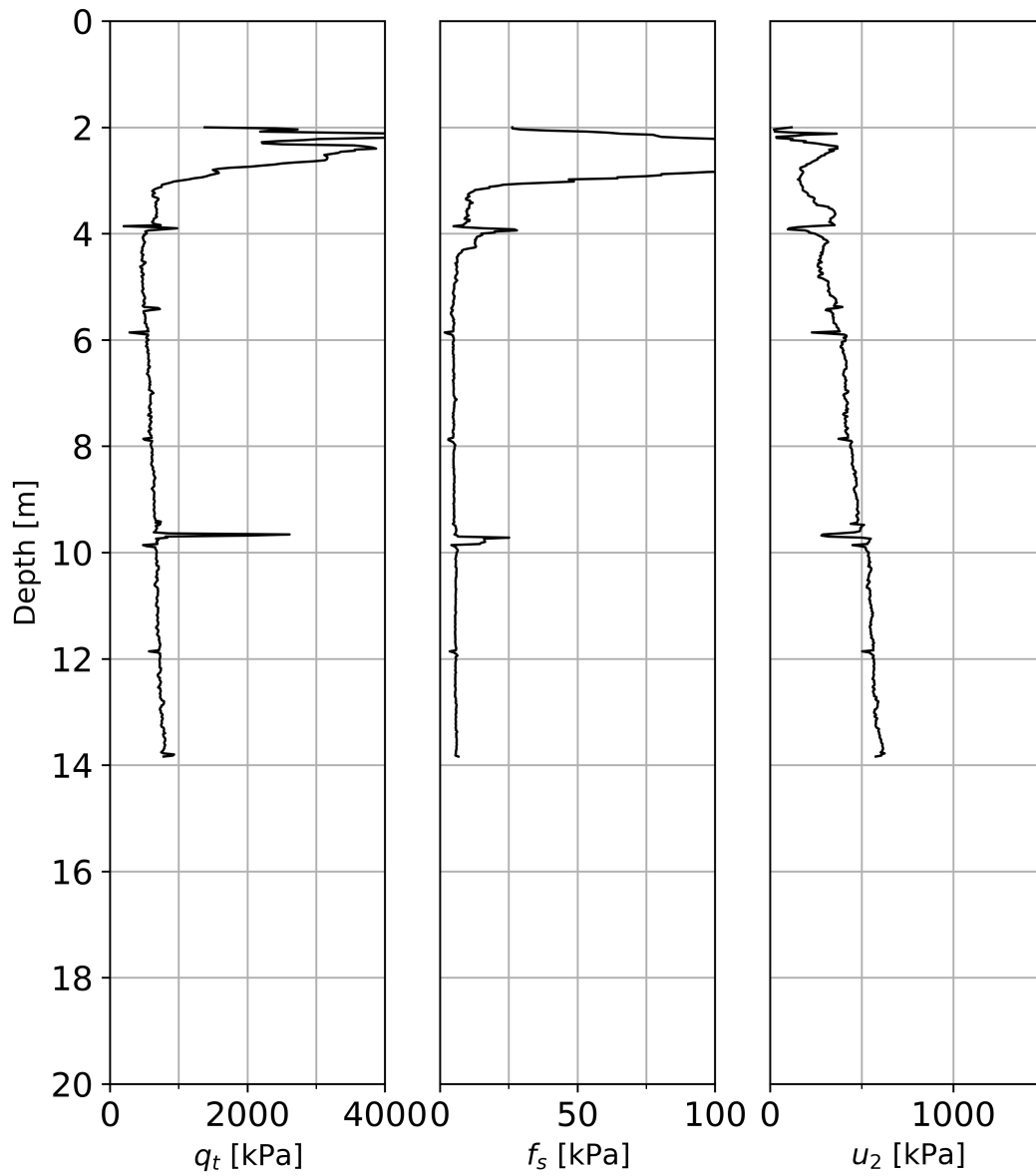


Figure A.43: Raw data graphs of Saksvik CPTu 9, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## Saksvik CPTu s9

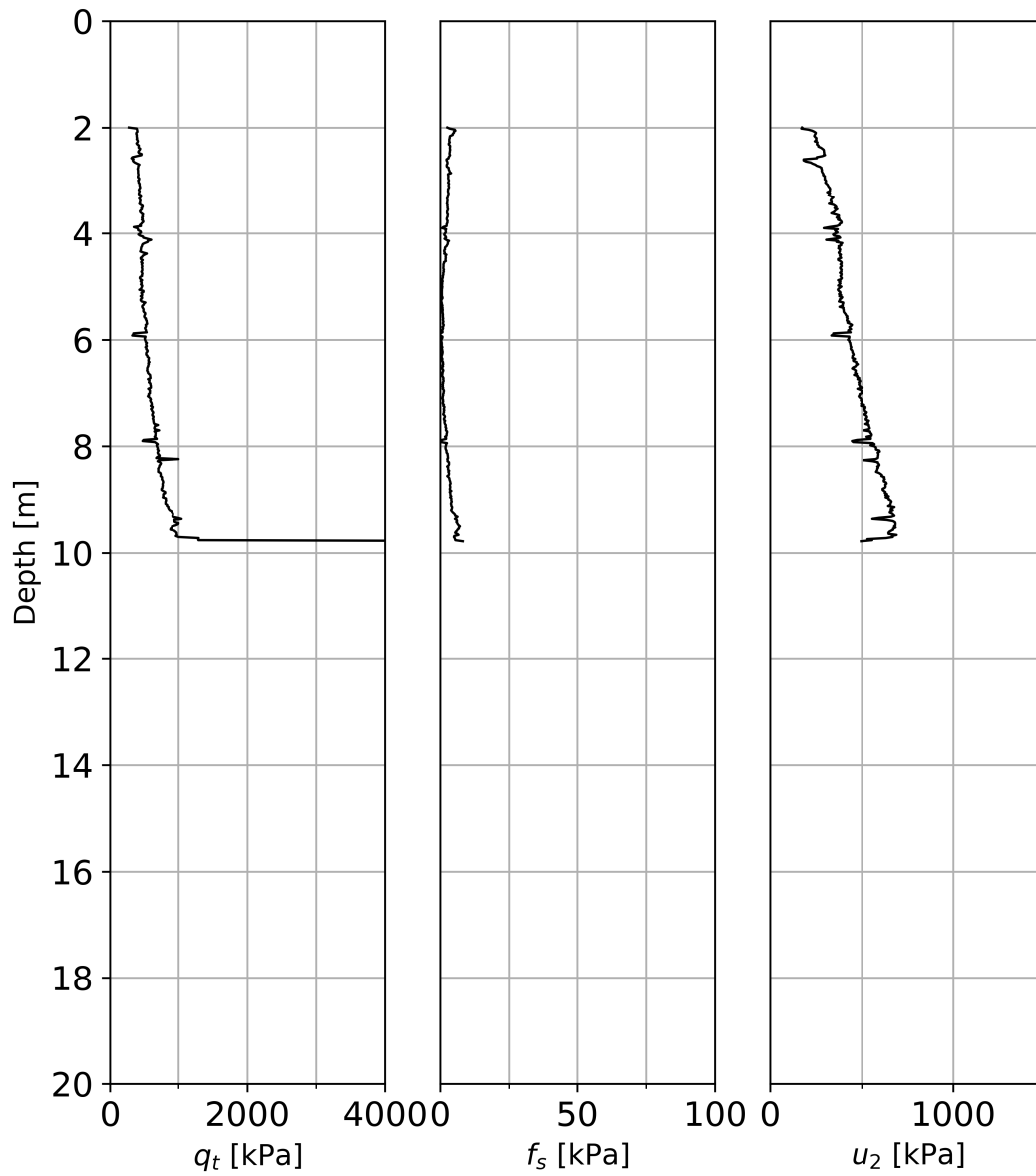


Figure A.44: Raw data graphs of Saksvik CPTu S9, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .



## Saksvik CPTu s3

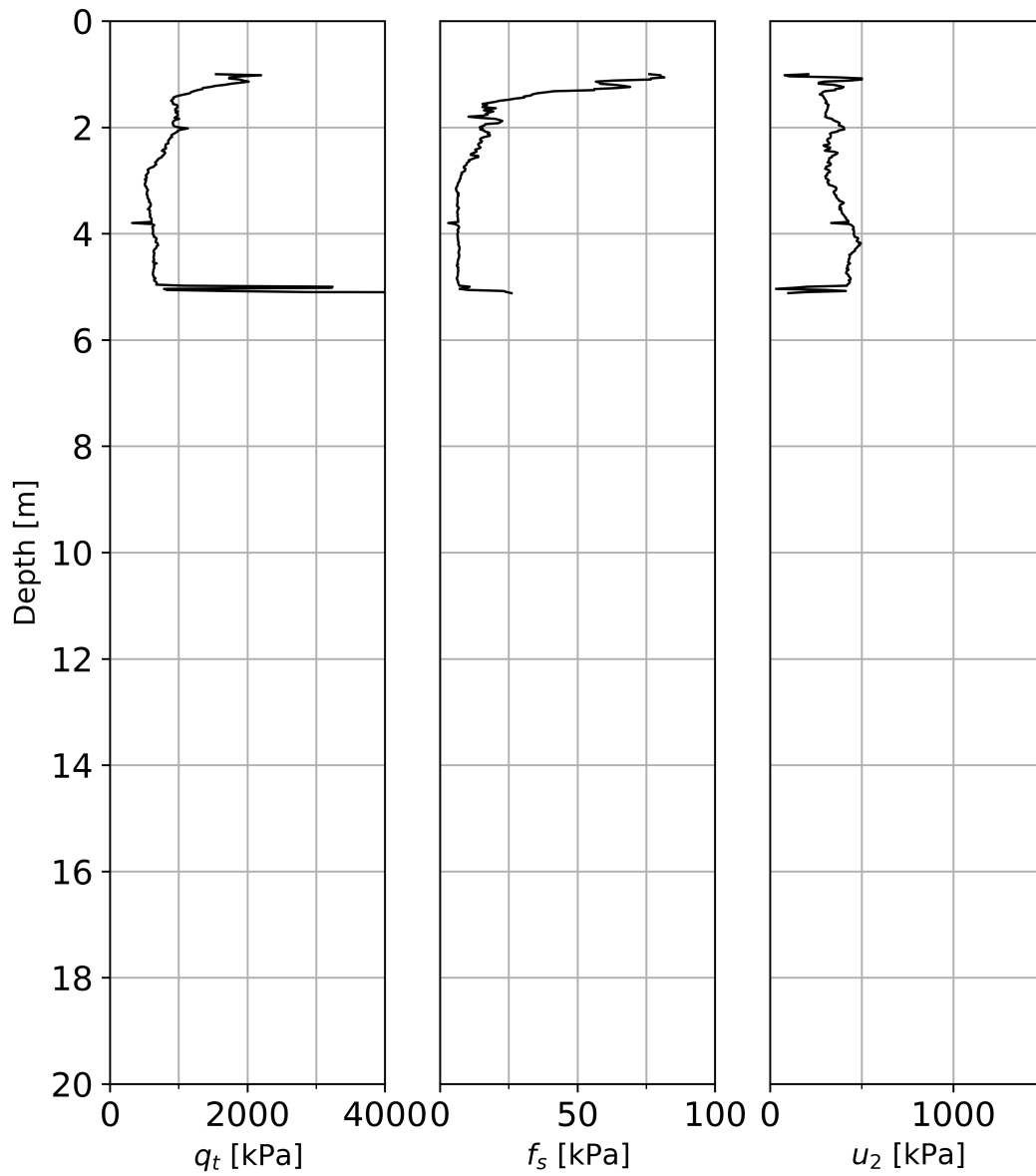


Figure A.45: Raw data graphs of Saksvik CPTu S3, displaying corrected cone resistance  $q_t$ , sleeve friction  $f_s$  and pore pressure  $u_2$ .

## **Appendix B**

### **Saksvik total soundings**

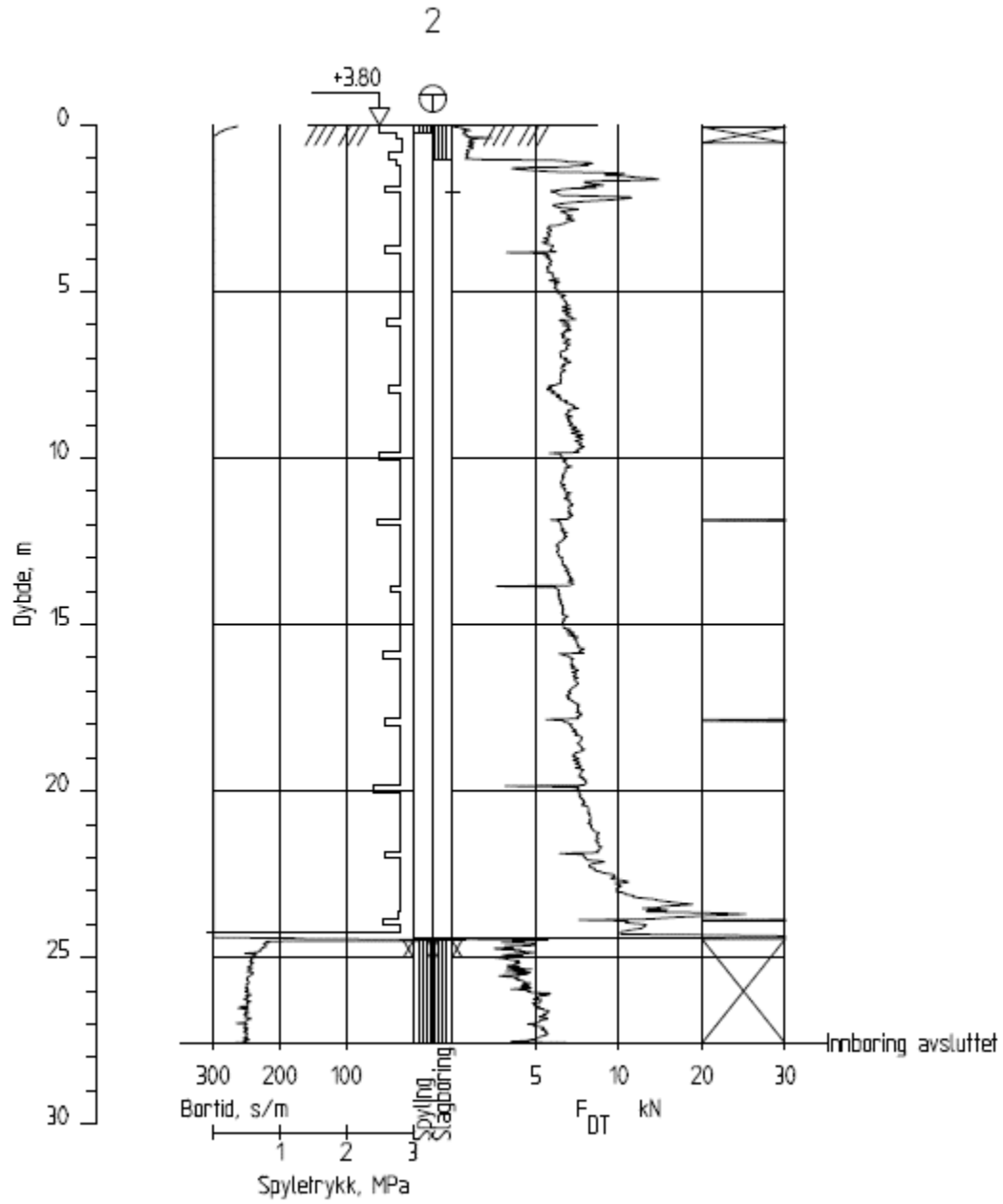


Figure B.1: Total sounding 2 at Skatval site, Malvik.

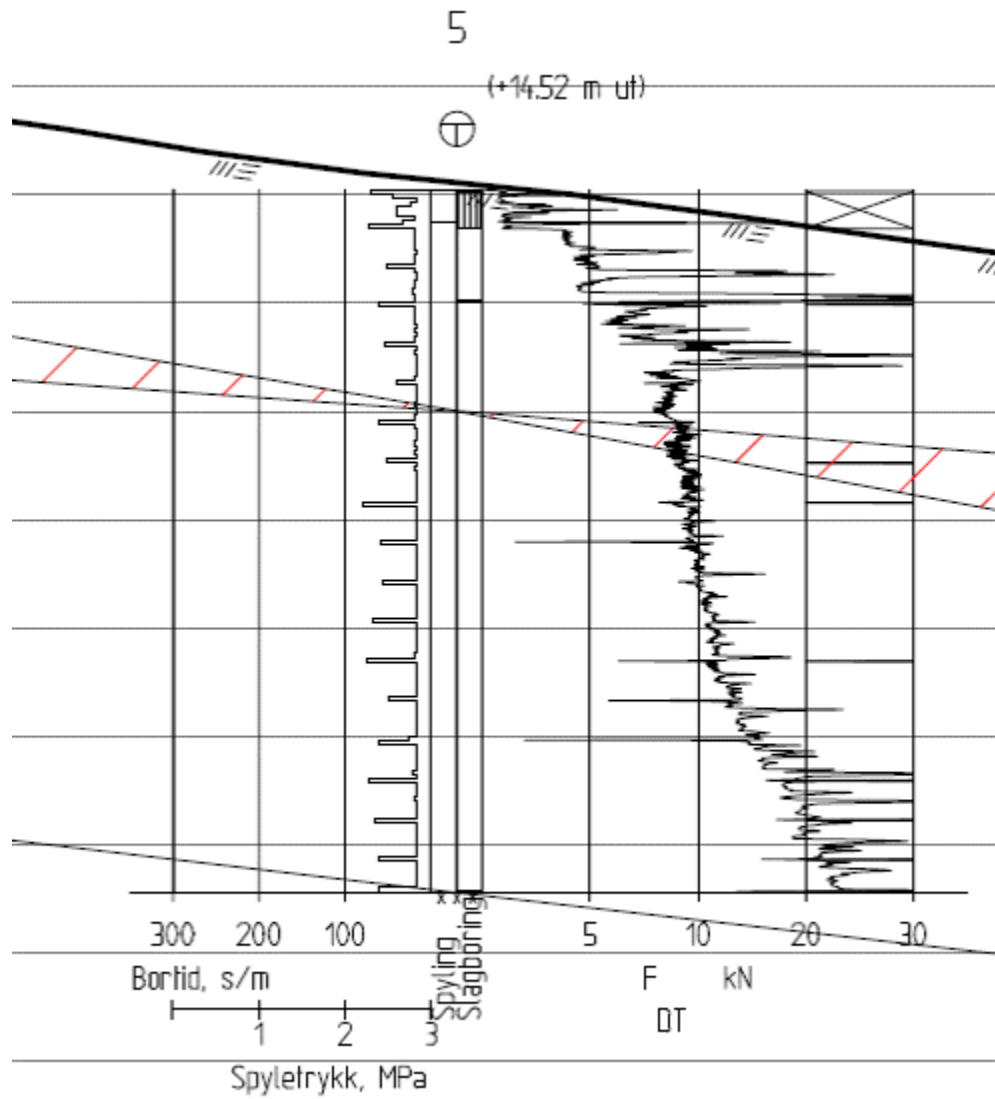


Figure B.2: Total sounding 5R at Skatval site, Malvik.

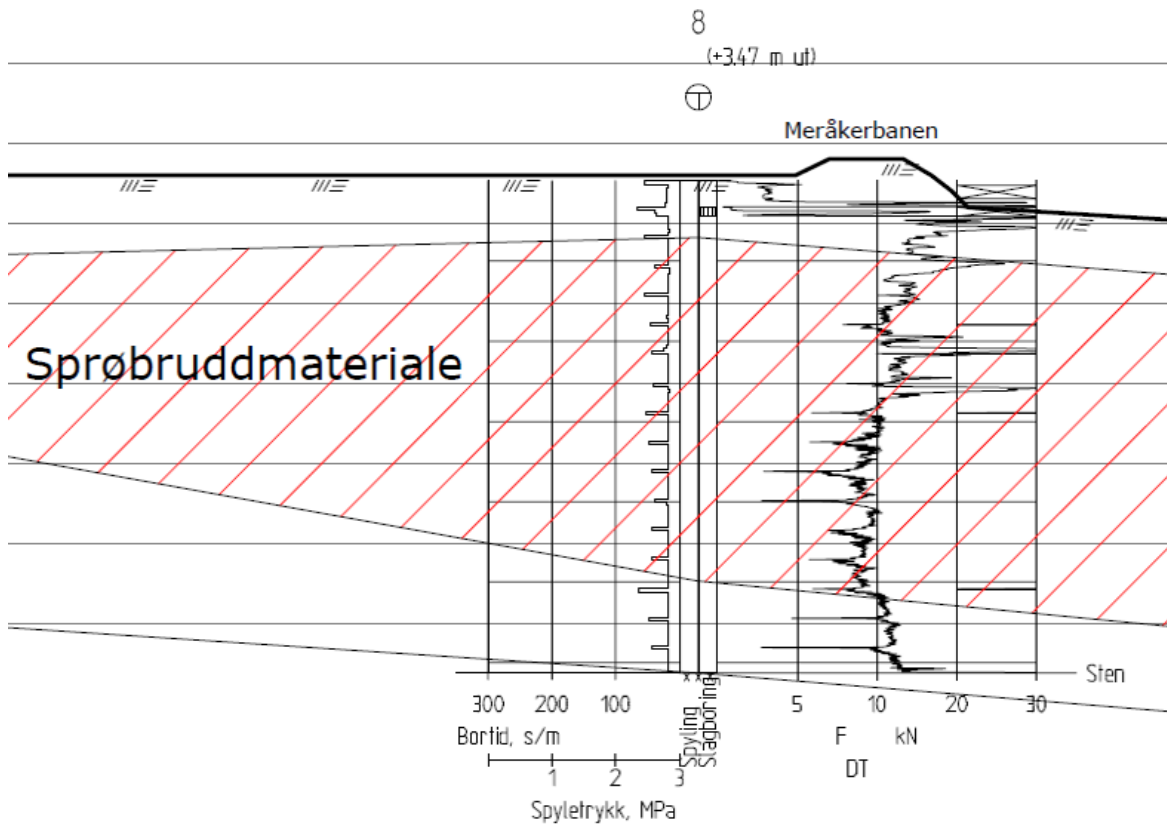


Figure B.3: Total sounding 8R at Skatval site, Malvik.

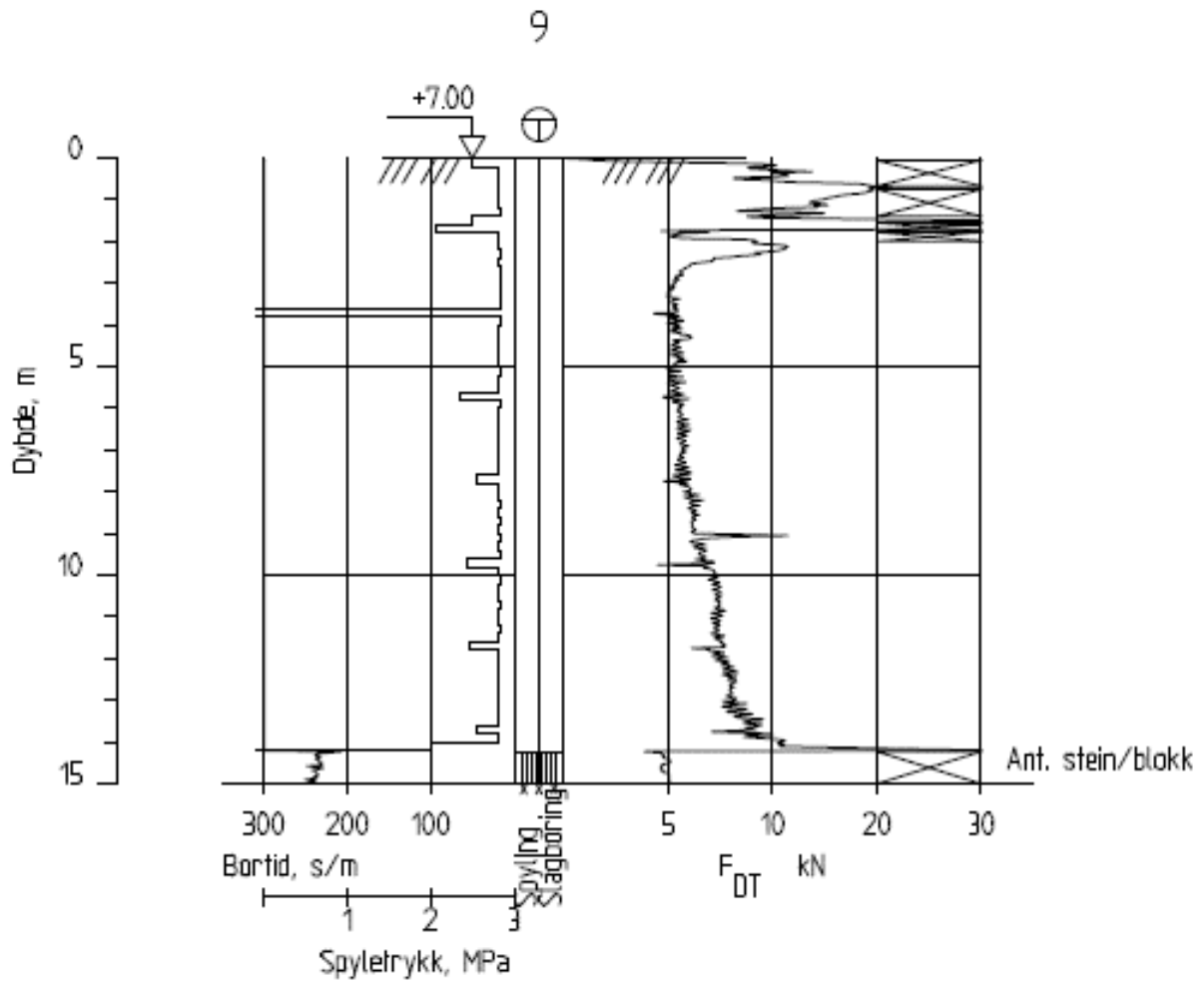


Figure B.4: Total sounding 9 at Skatval site, Malvik.

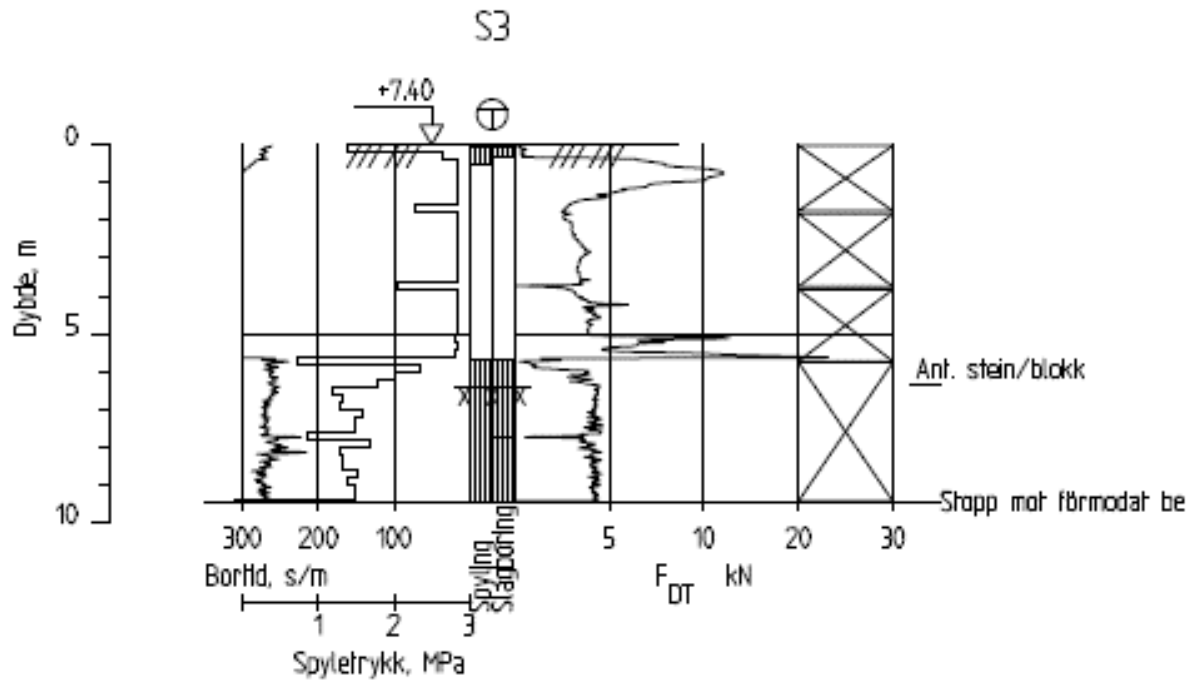


Figure B.5: Total sounding S3 at Skatval site, Malvik.

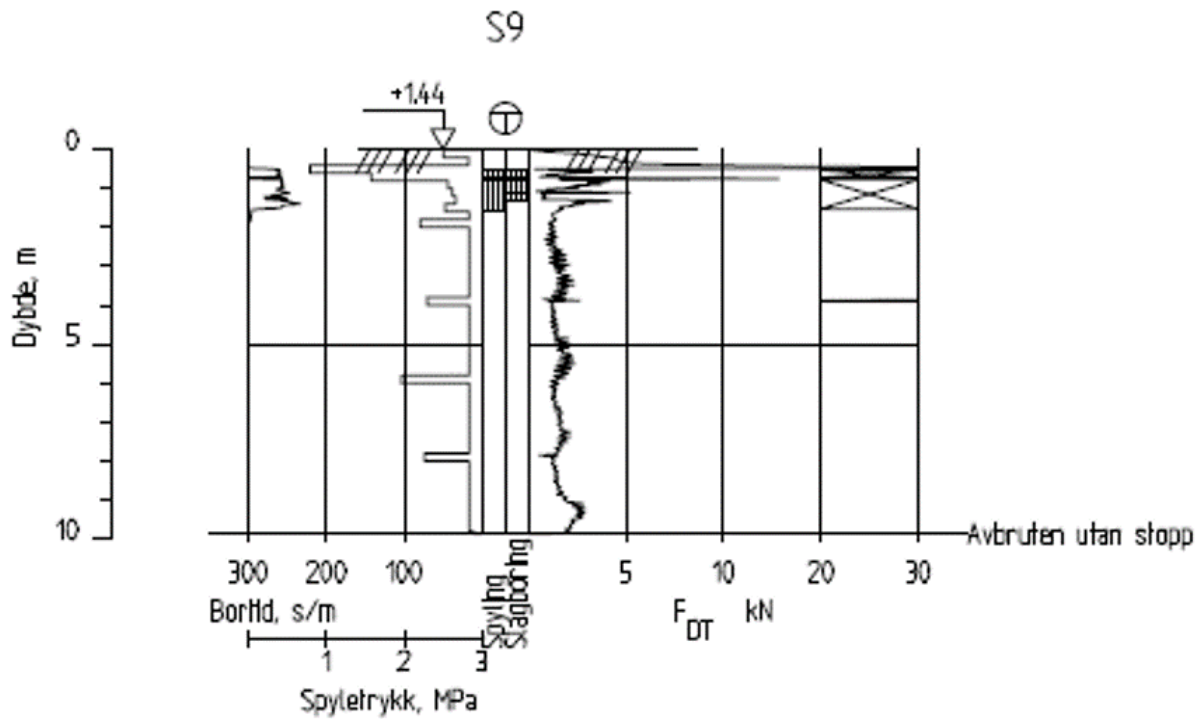


Figure B.6: Total sounding S9 at Skatval site, Malvik.

## **Appendix C**

### **Index testing profiles**



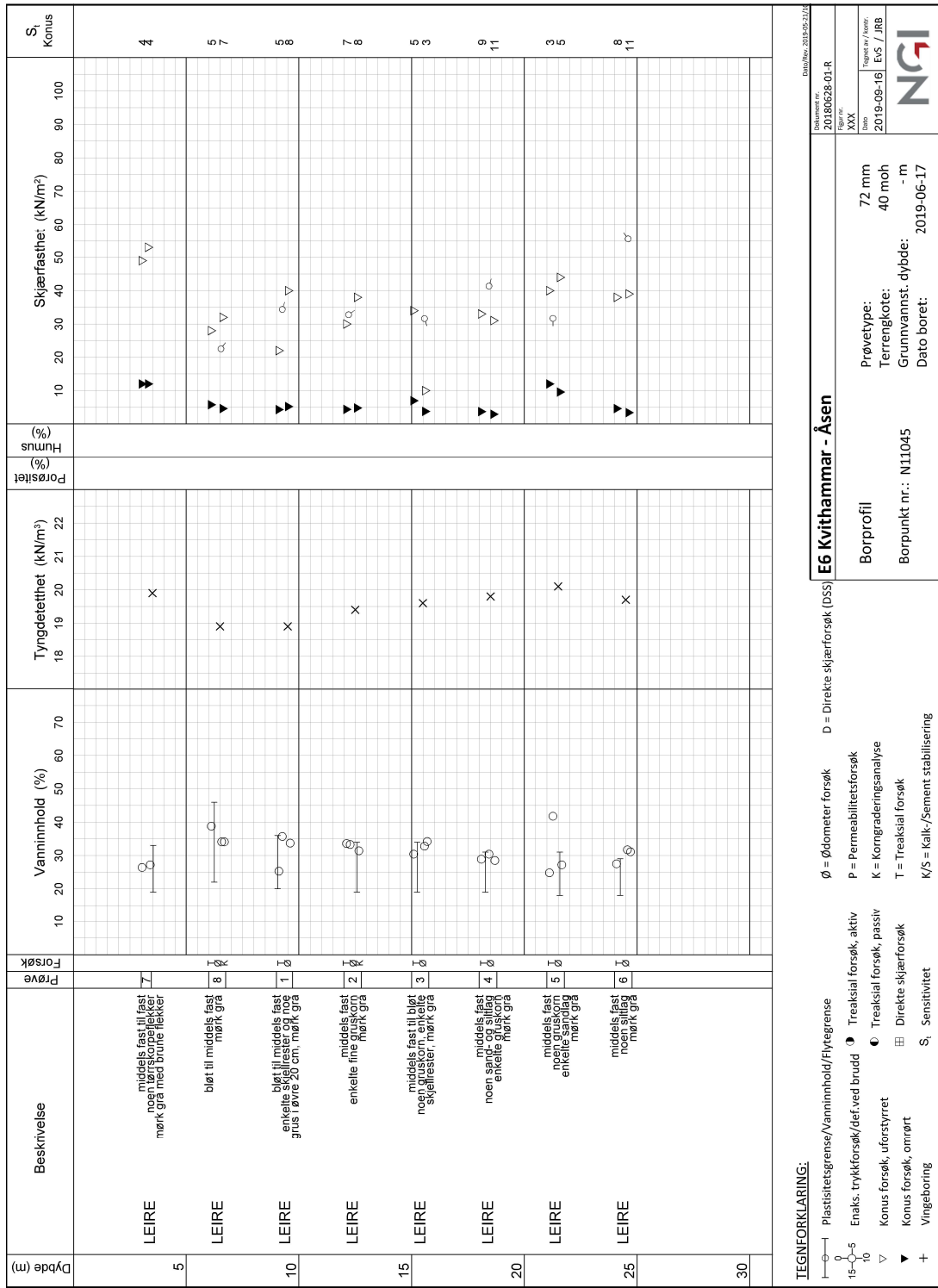


Figure C.1: Index testing from E6 Kvithammer-Åsen.

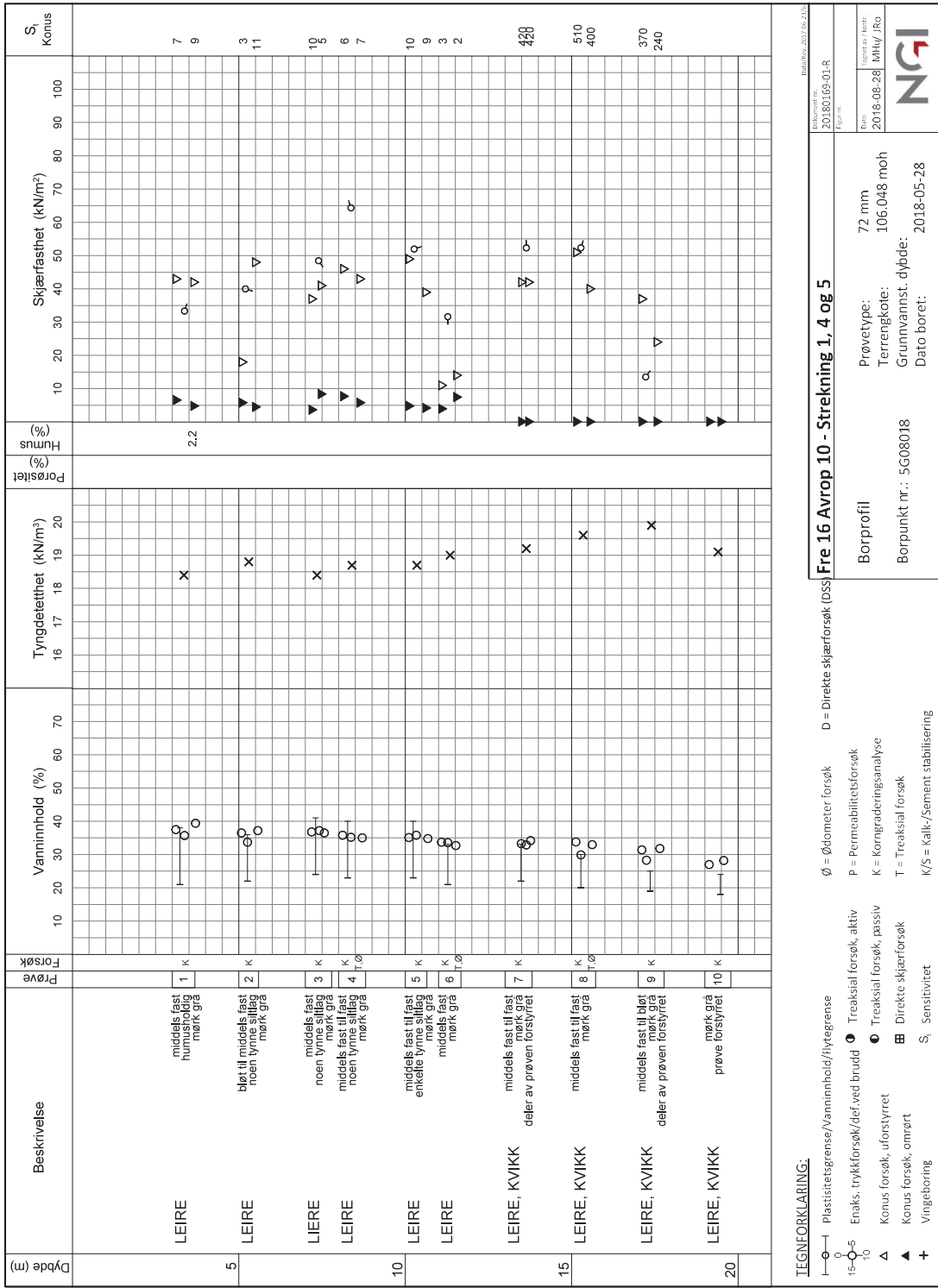


Figure C.2: Index testing from Fre16 (Ringeriksbanen and E16 - the joint railway & road project).



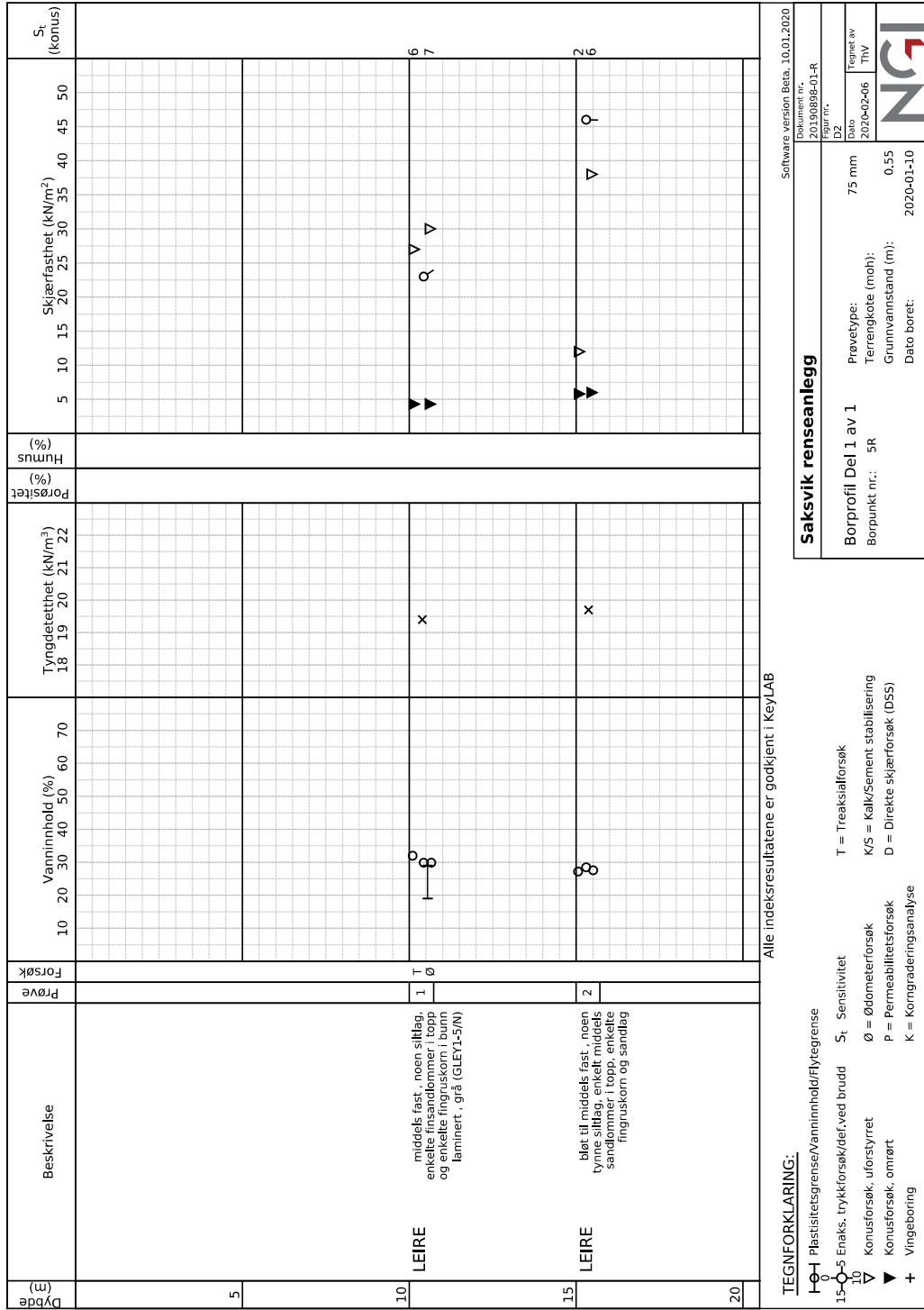


Figure C.4: Index testing from Saksvik 5R.







## **Appendix D**

### **Saksvik piezometer**



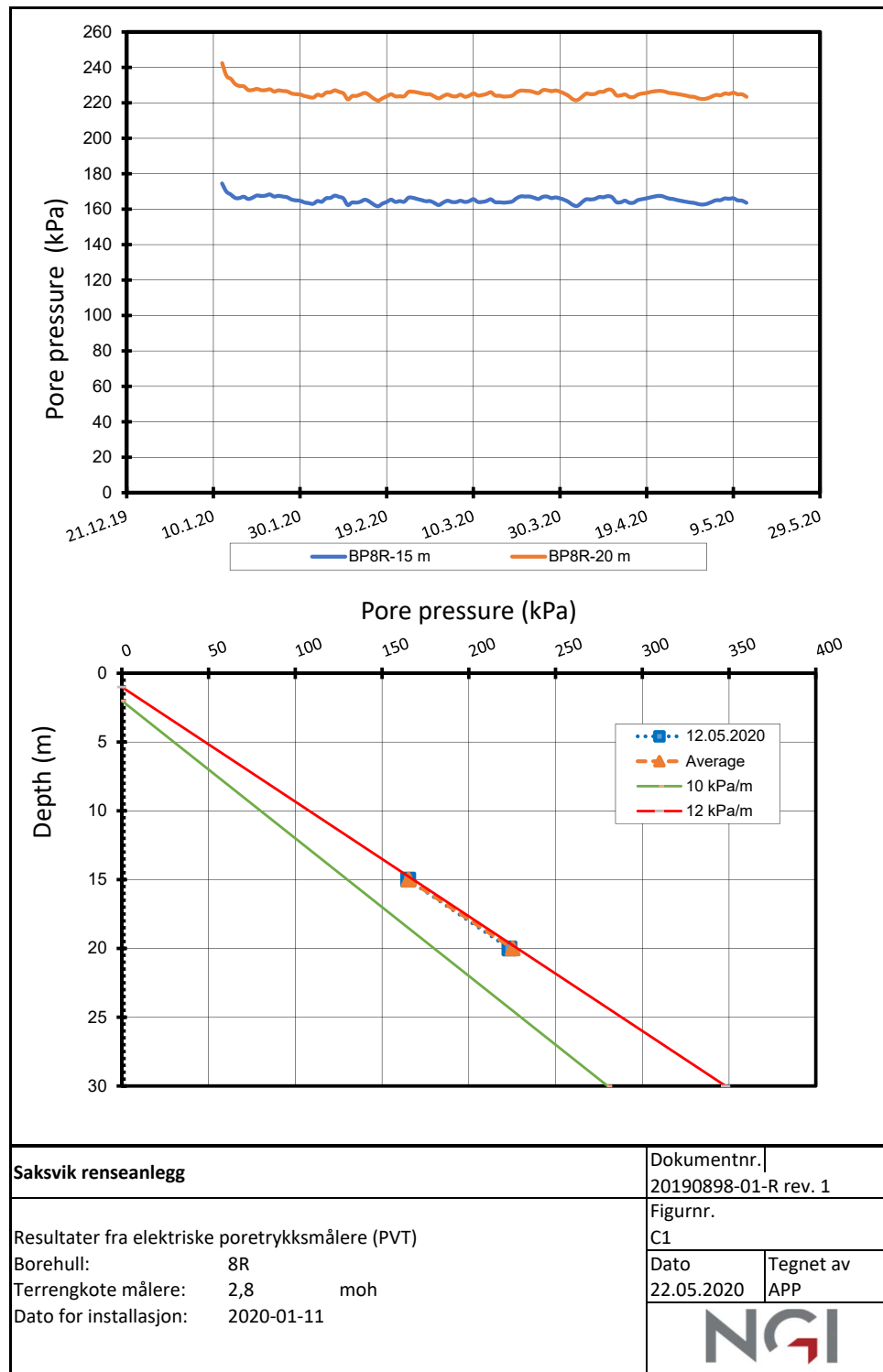


Figure D.1: Pore pressure at 15 and 20 m depth, borehole 8R Saksvik.

## **Appendix E**

**2D plots from NGTS dataset not included in  
the text**

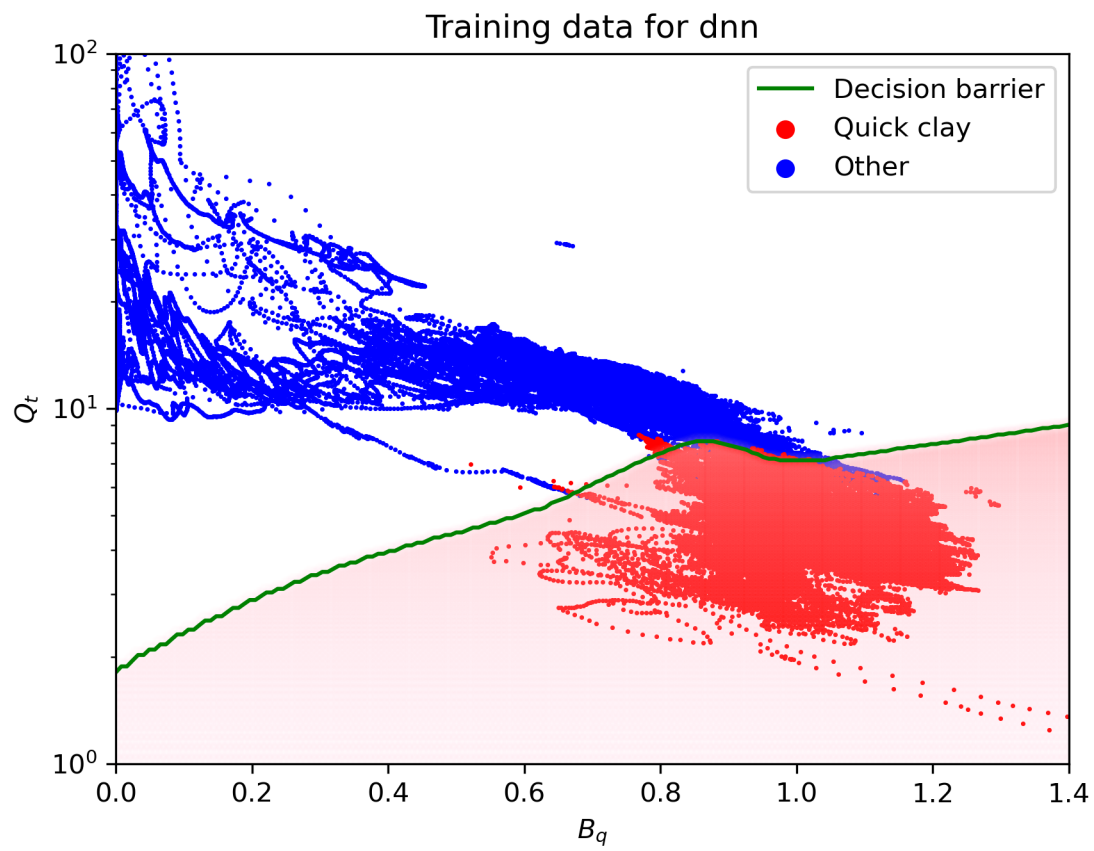


Figure E.1:  $Q_t$  vs  $B_q$  plot of a DNN model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

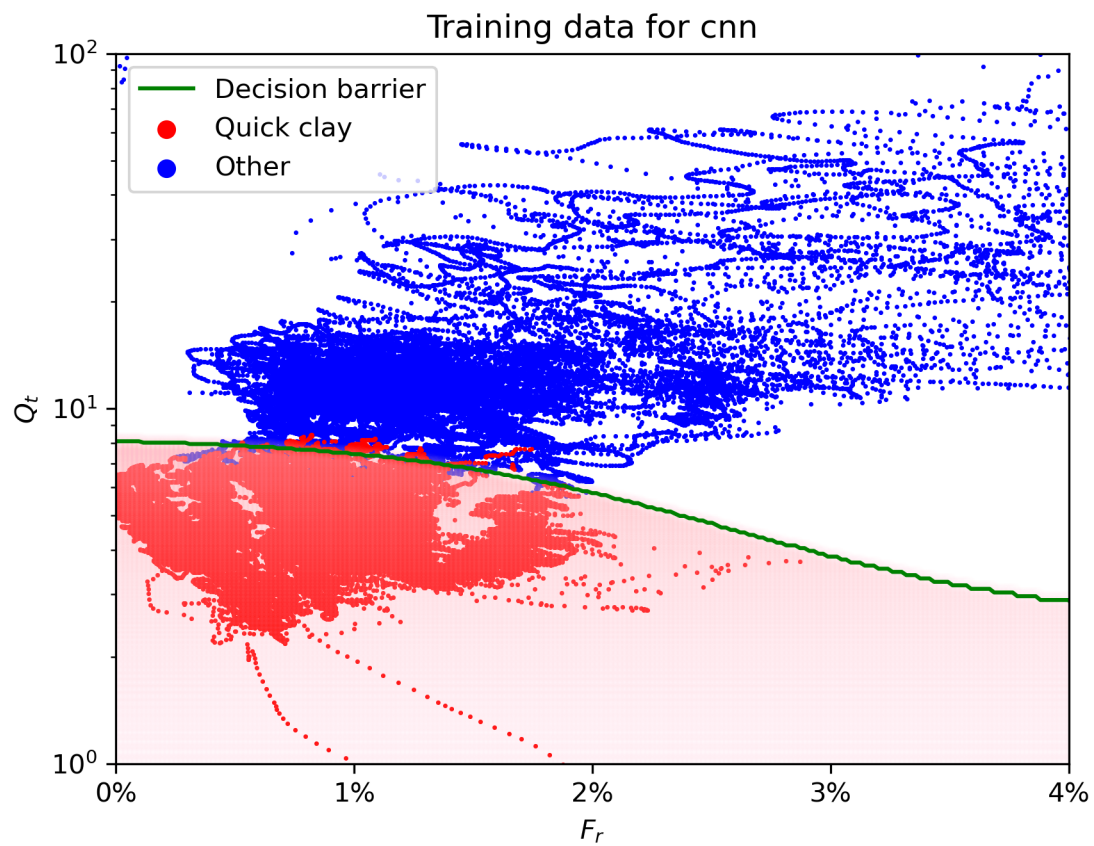


Figure E.2:  $Q_t$  vs  $F_r$  plot of a CNN model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

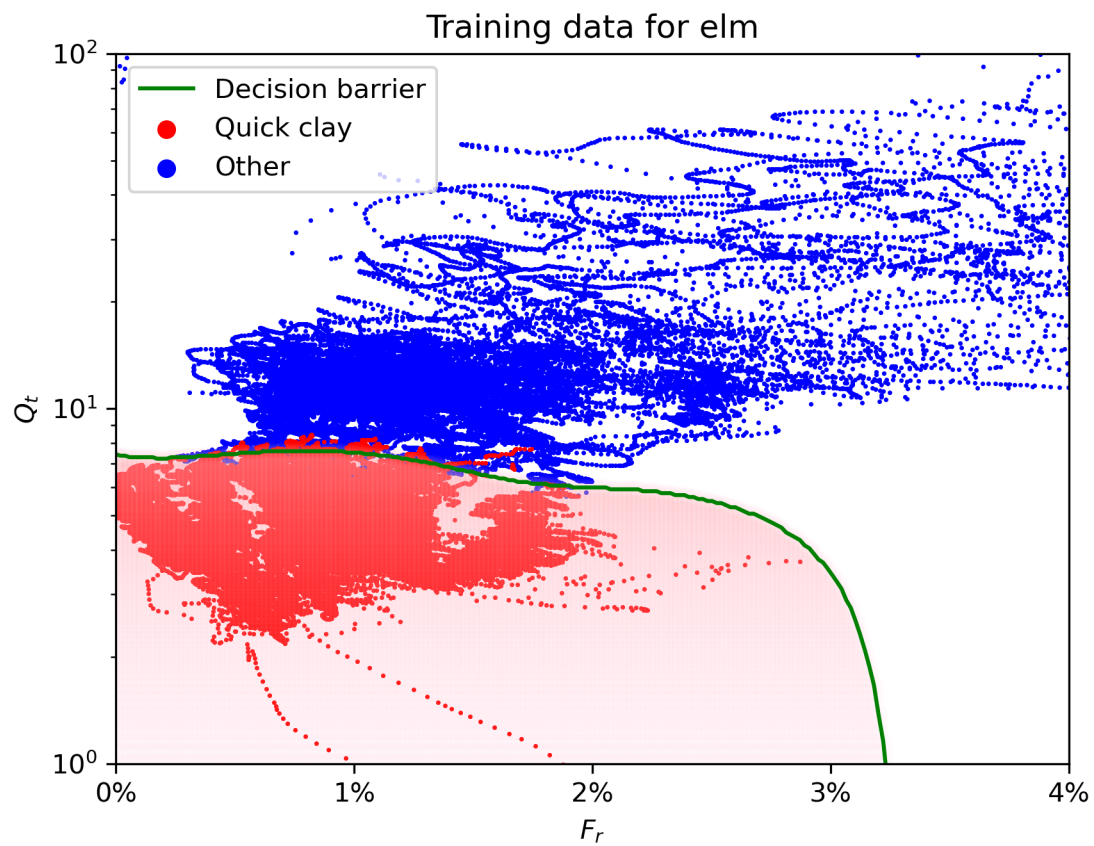


Figure E.3:  $Q_t$  vs  $F_r$  plot of a ELM model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

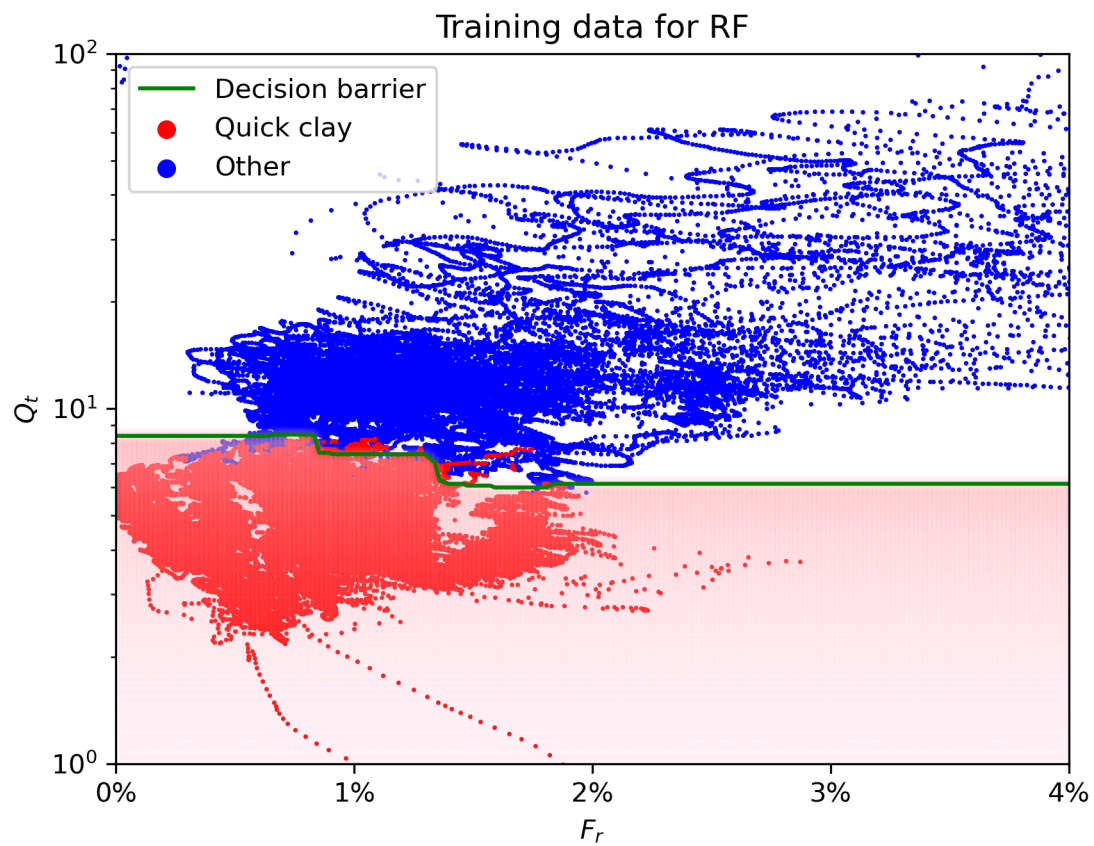


Figure E.4:  $Q_t$  vs  $F_r$  plot of a RF model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

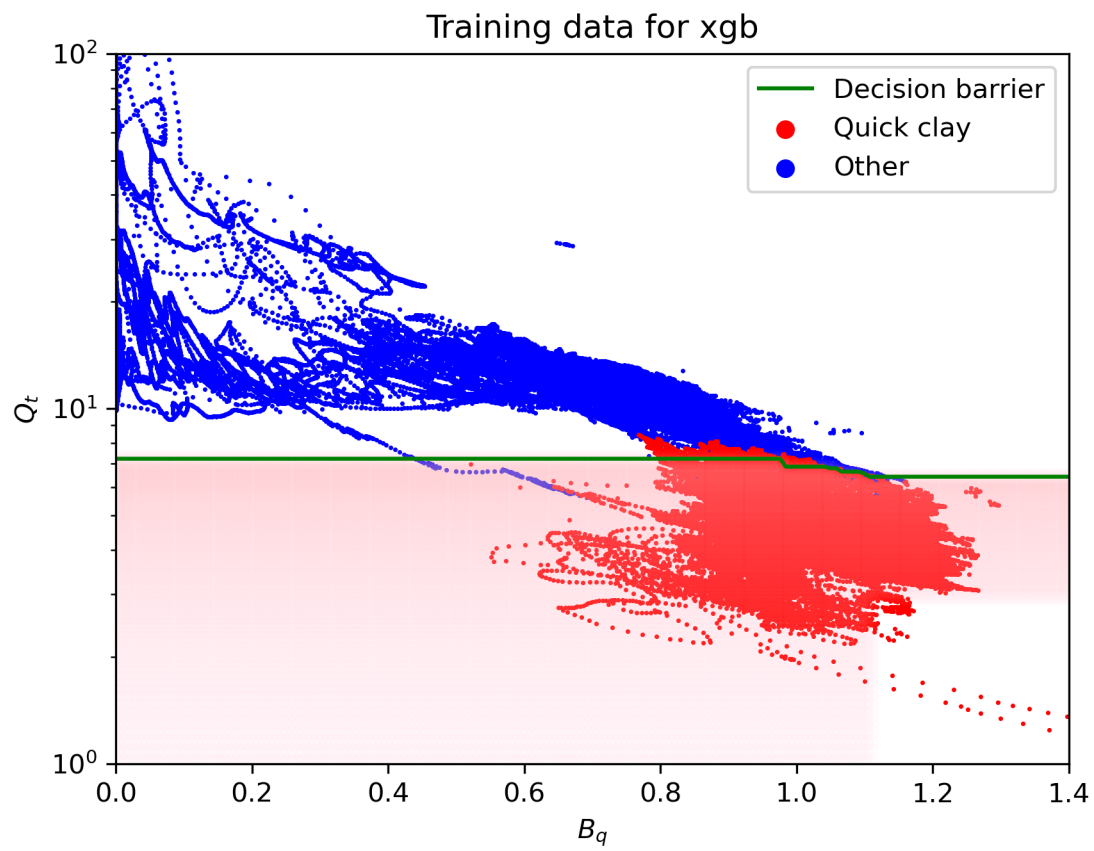


Figure E.5:  $Q_t$  vs  $B_q$  plot of a XGB model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

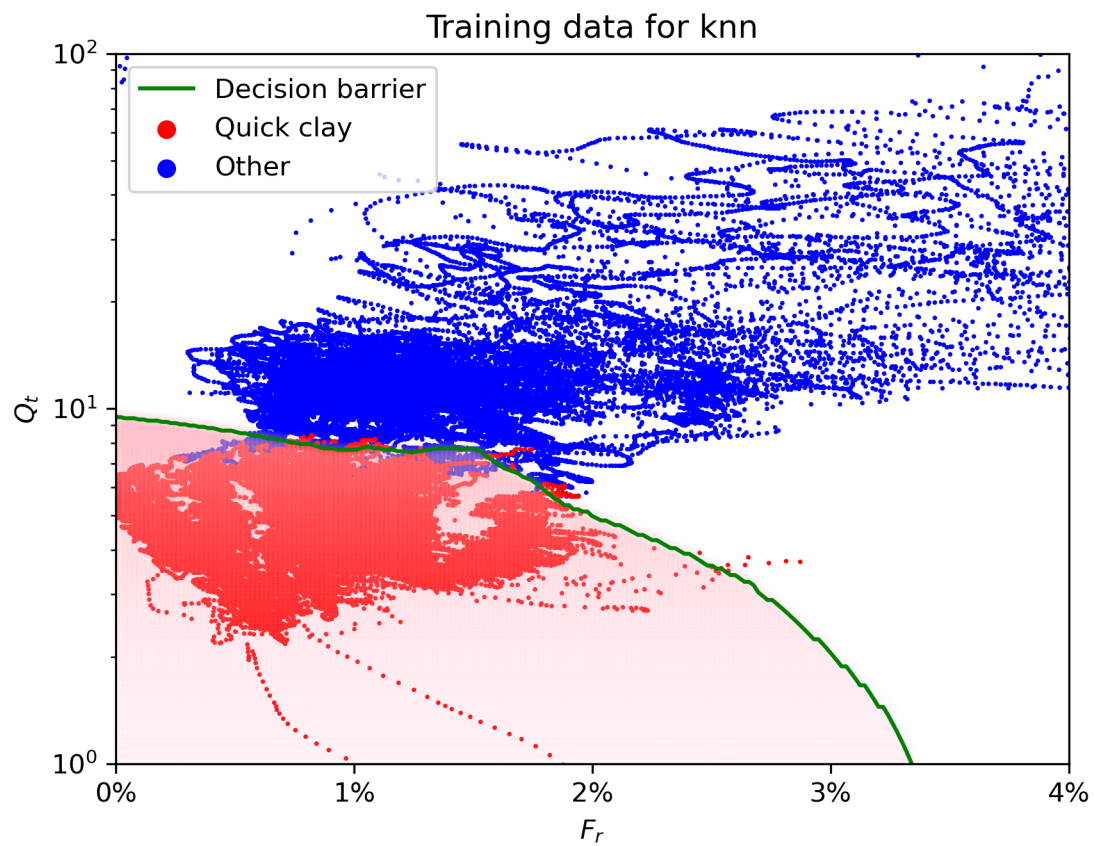


Figure E.6:  $Q_t$  vs  $F_r$  plot of a KNN model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.



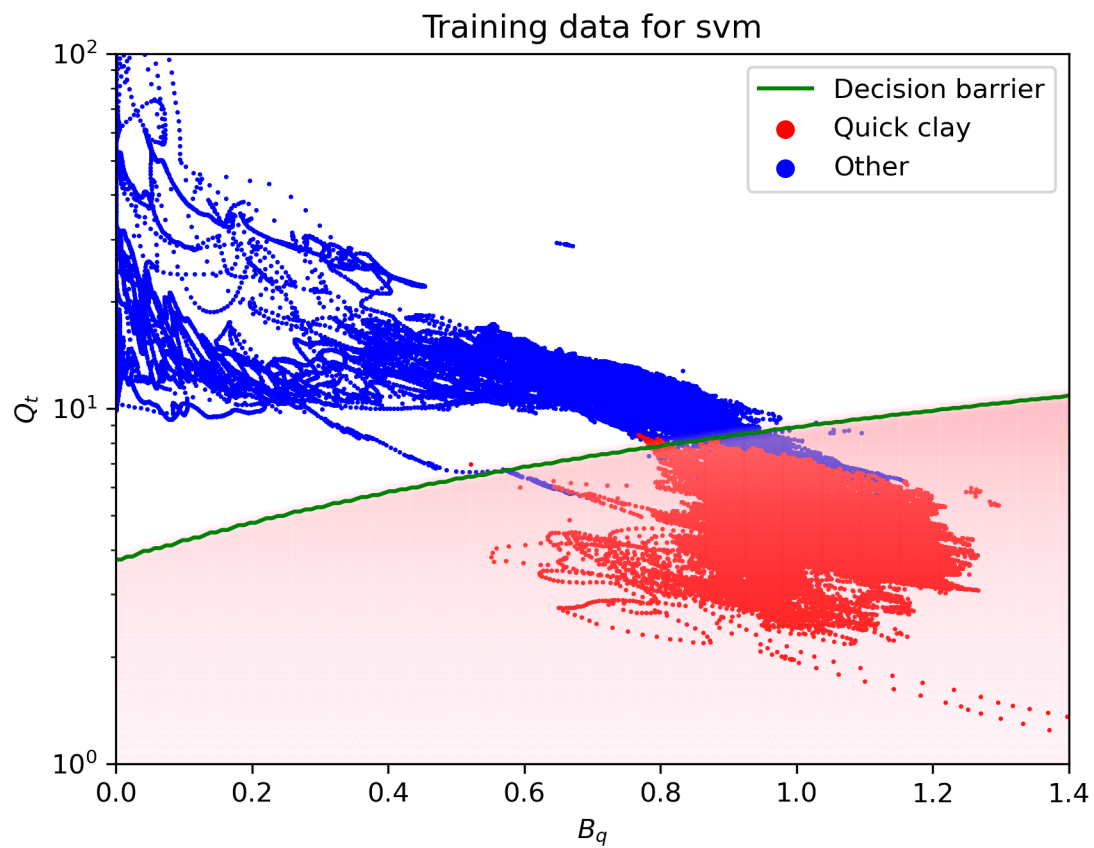


Figure E.7:  $Q_t$  vs  $B_q$  plot of a SVM model trained on 5 CPTus from NGTS Tiller-Flotten. The model classifies all points in the shaded pink as quick clay. The edge of the quick clay zone is marked in green.

## **Appendix F**

**3D plots from dataset II not included in the text**

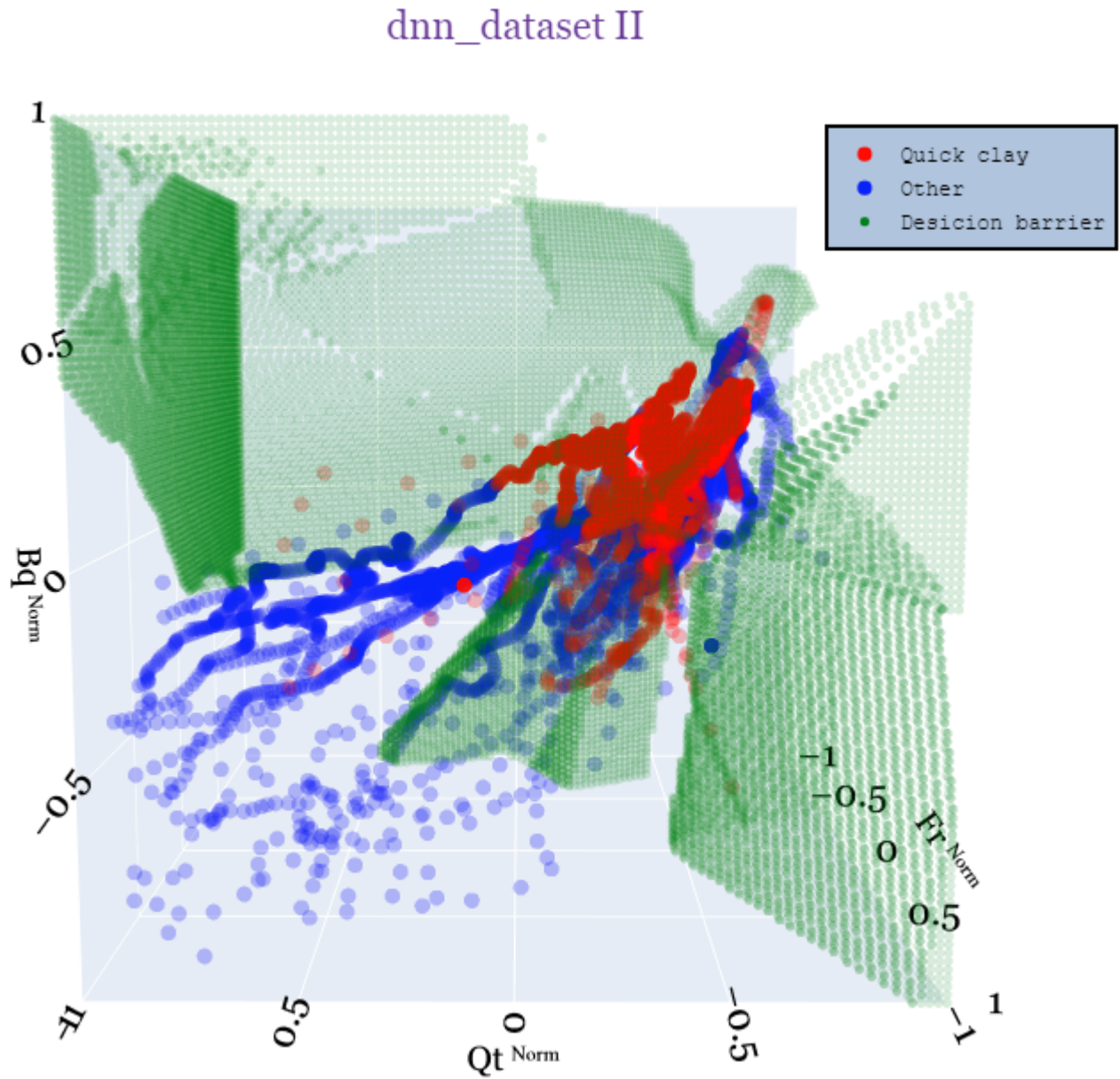


Figure F.1: 3D plot of DNN trained on dataset II (full). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

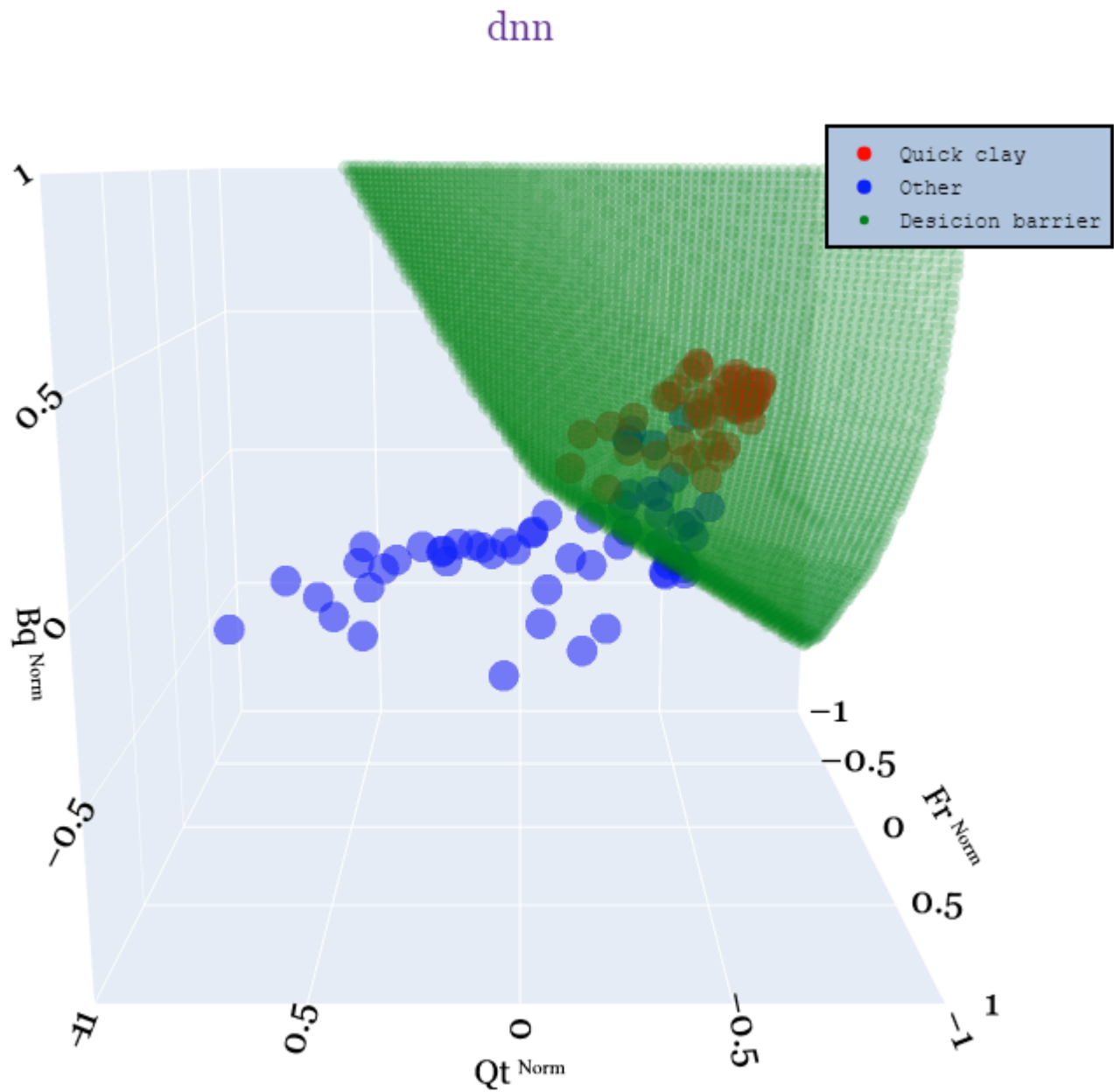


Figure E2: 3D plot of DNN trained on dataset II (reduced). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

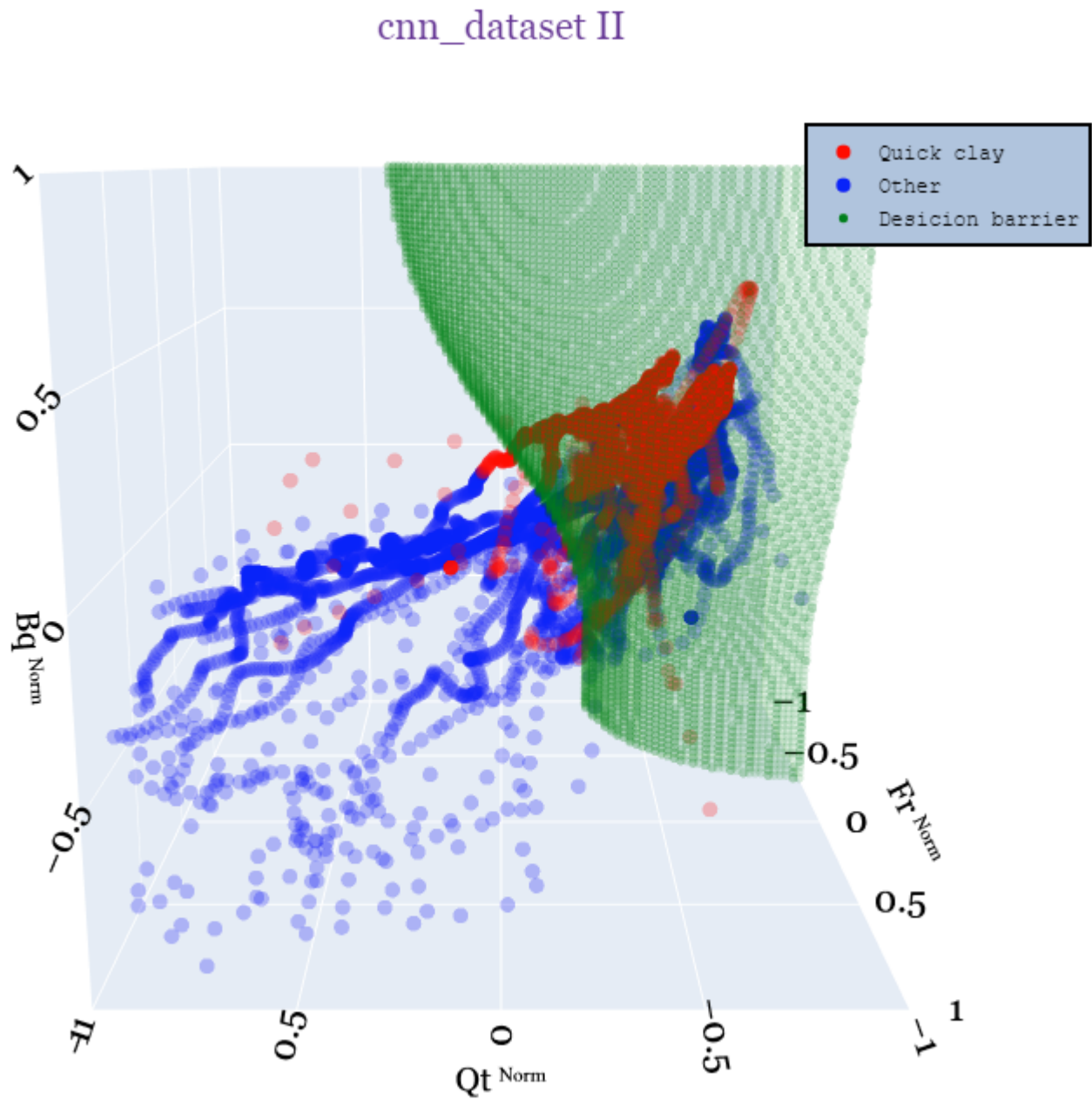


Figure E3: 3D plot of CNN trained on dataset II (full). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

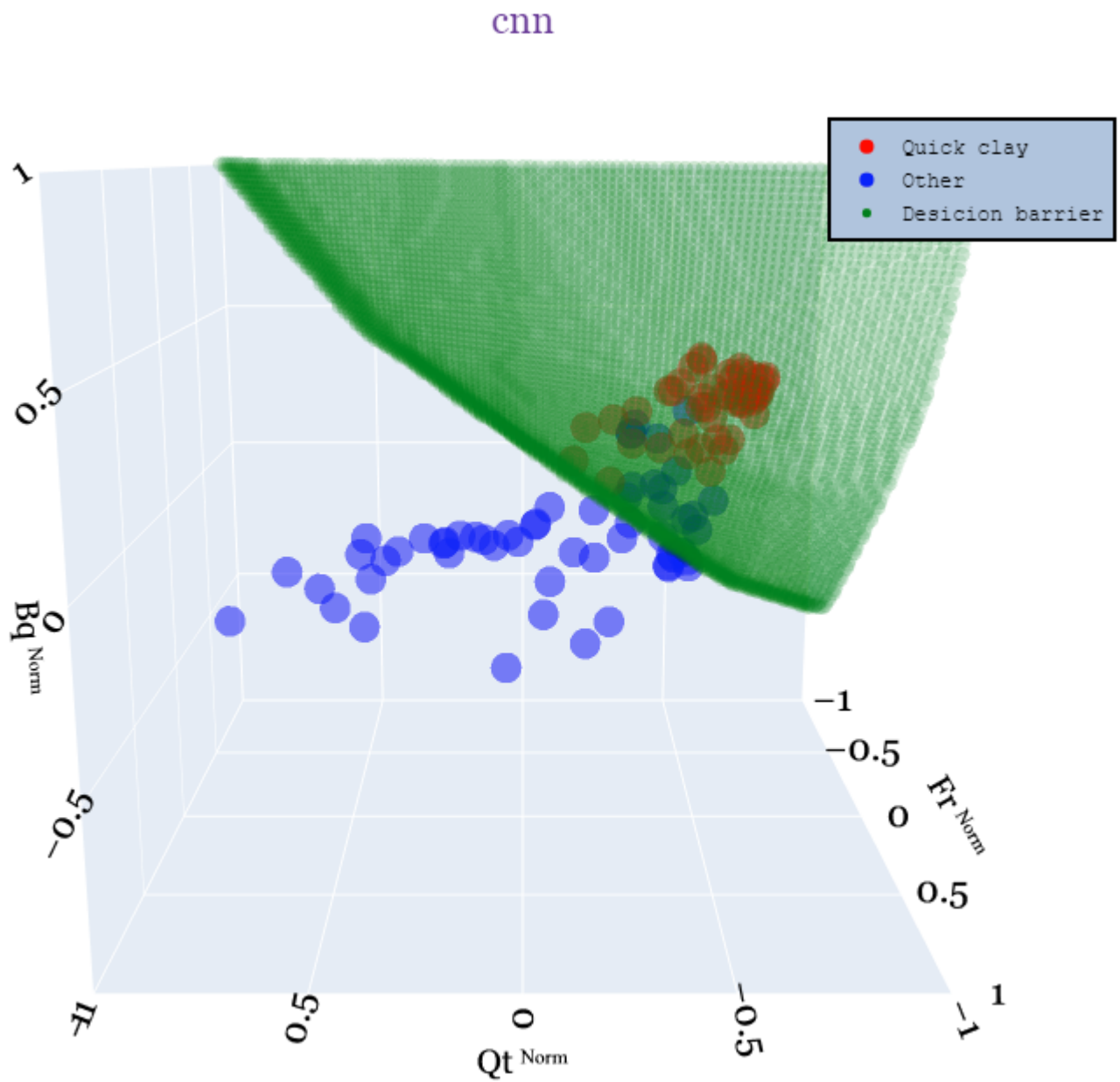


Figure E4: 3D plot of CNN trained on dataset II (reduced). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

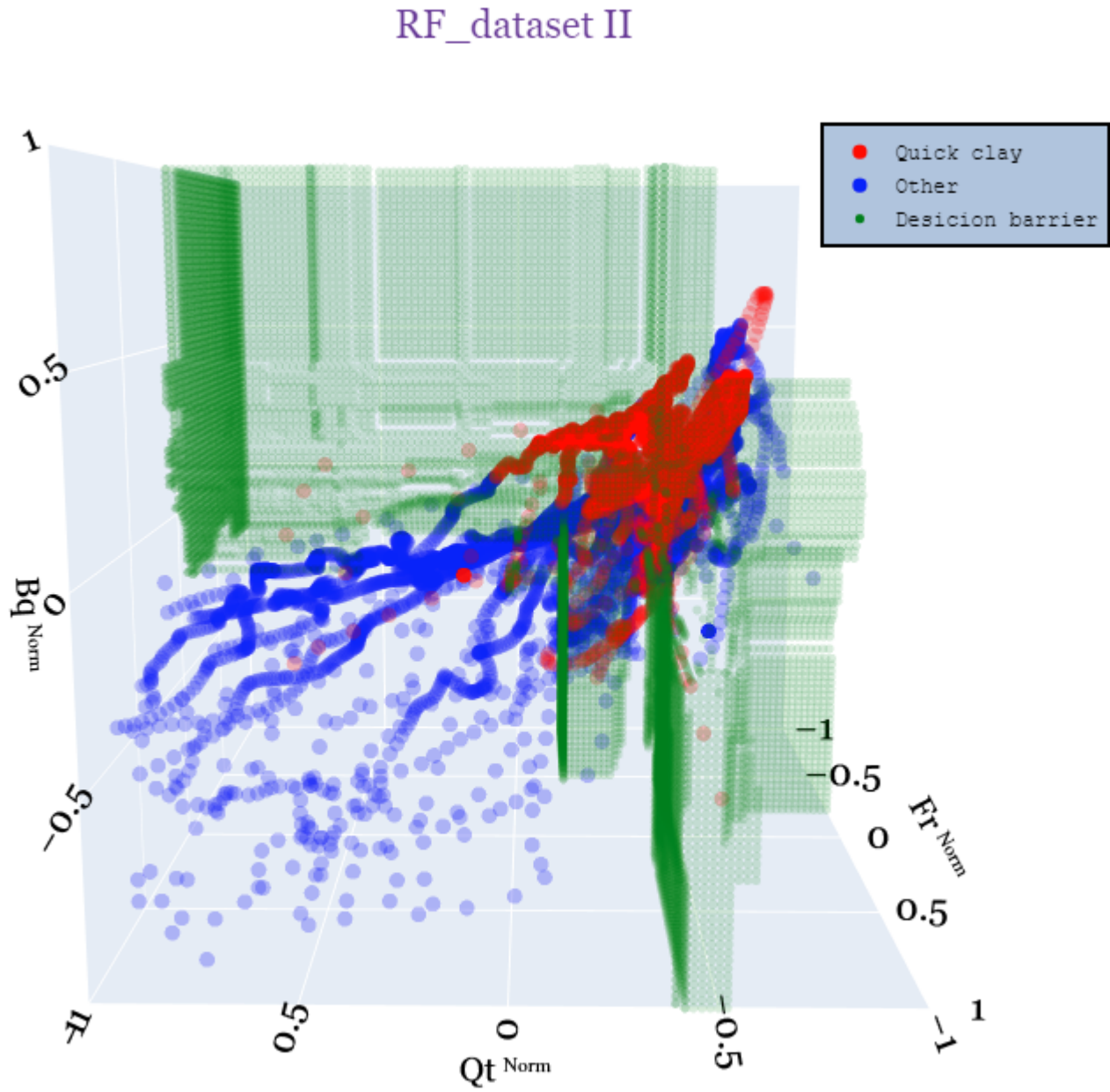


Figure E5: 3D plot of RF trained on dataset II (full). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

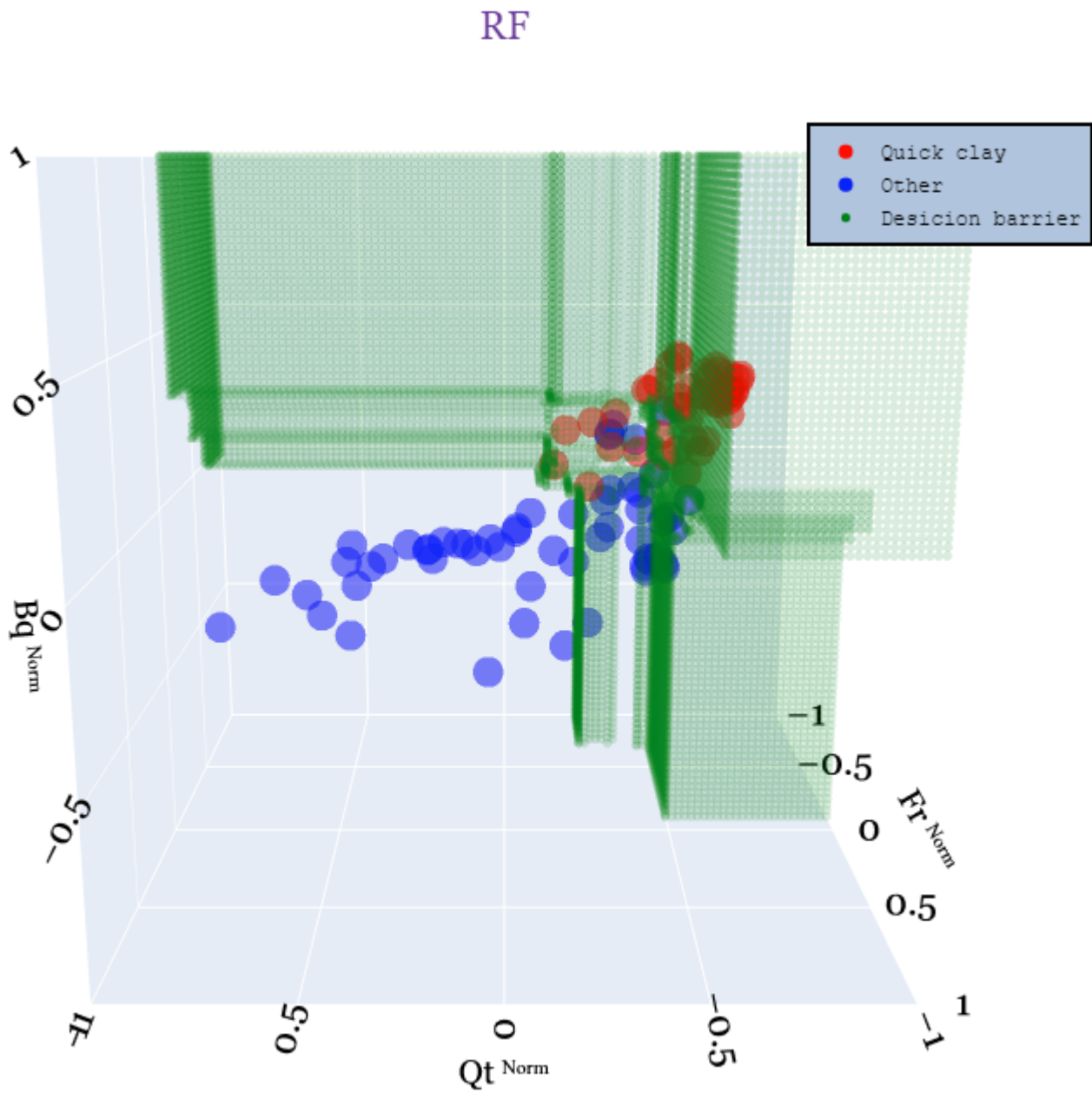


Figure E6: 3D plot of RF trained on dataset II (reduced). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.



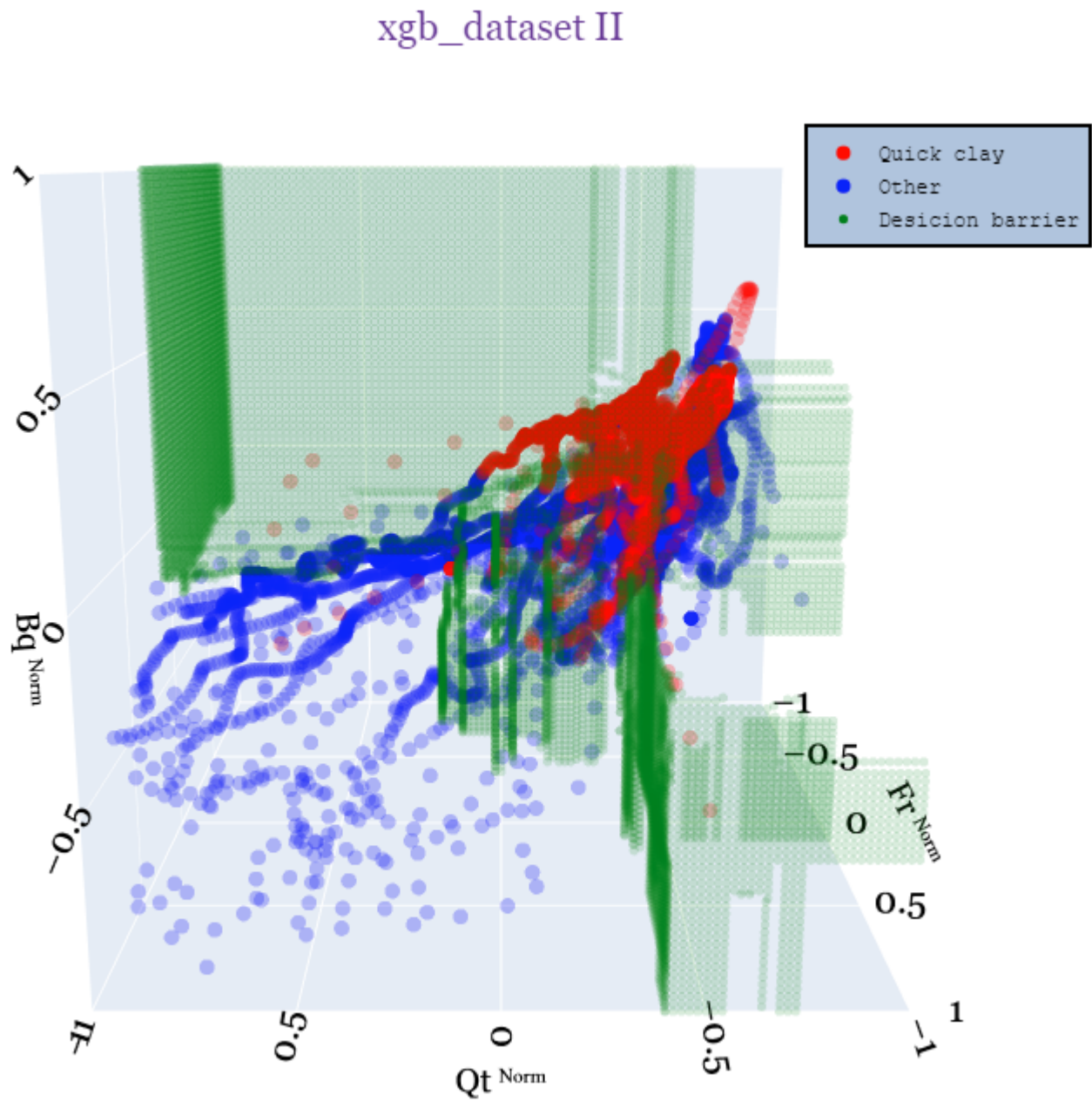


Figure E.7: 3D plot of XGB trained on dataset II (full). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

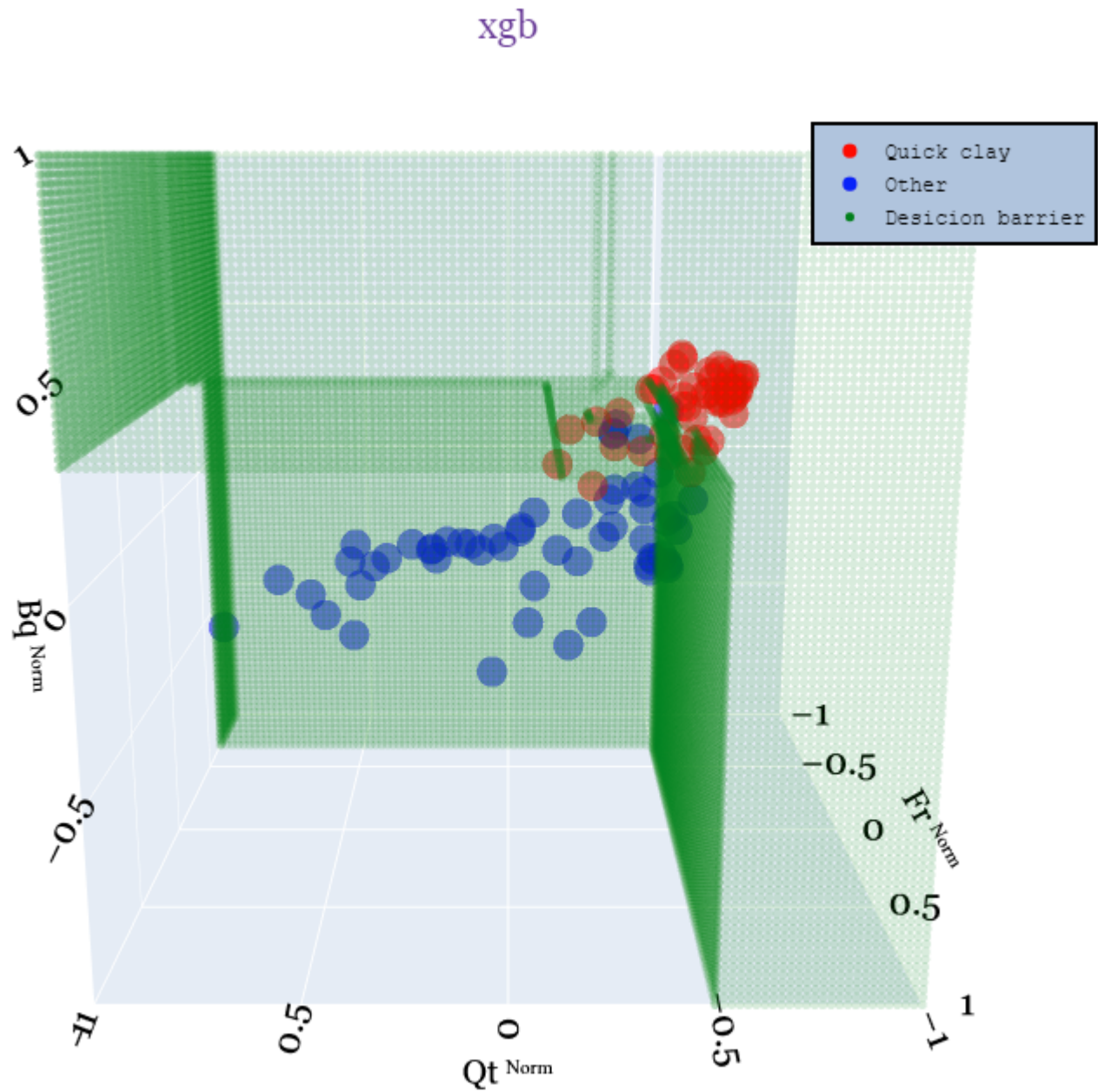


Figure E8: 3D plot of XGB trained on dataset II (reduced). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

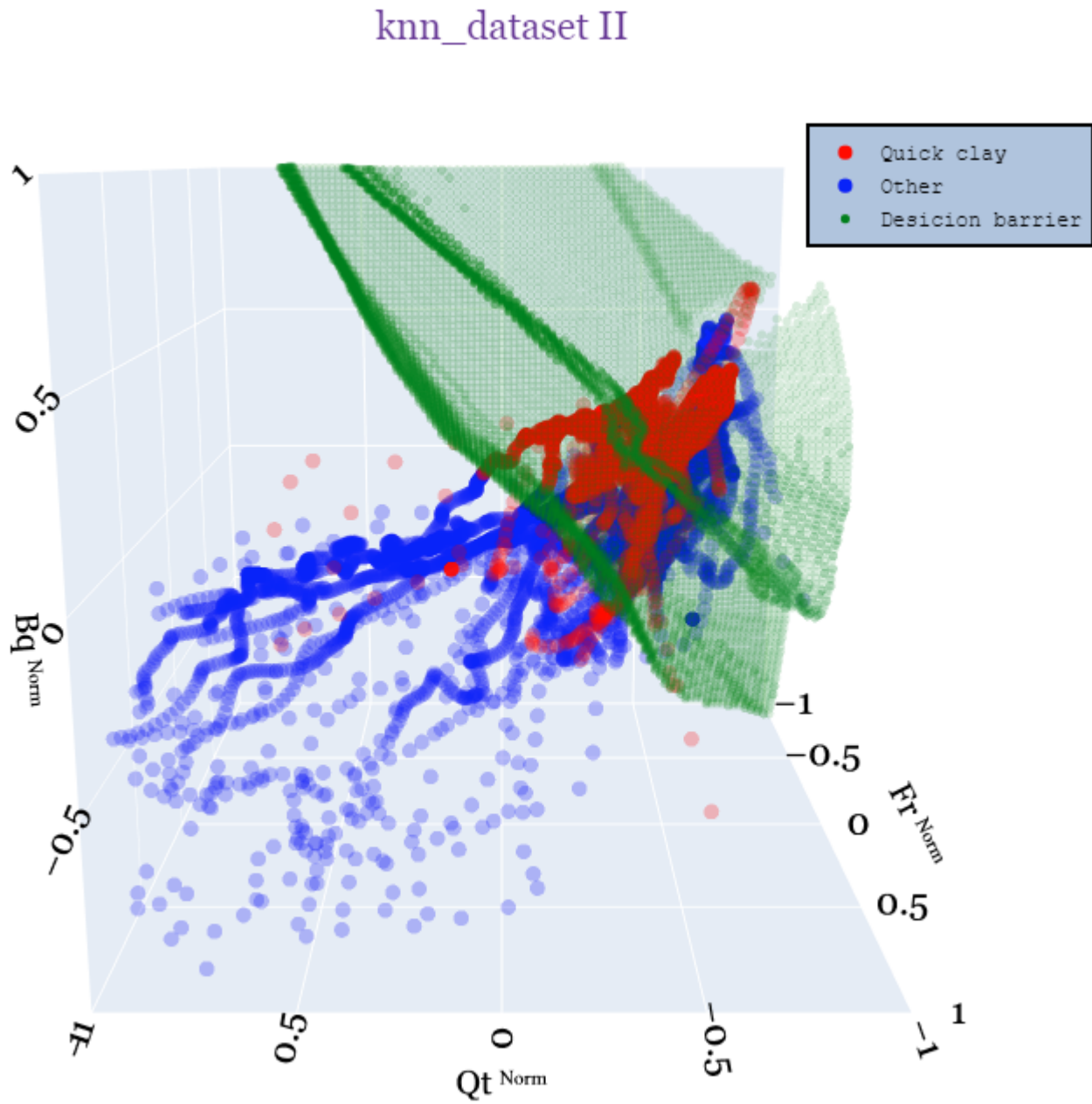


Figure E9: 3D plot of KNN trained on dataset II (full). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

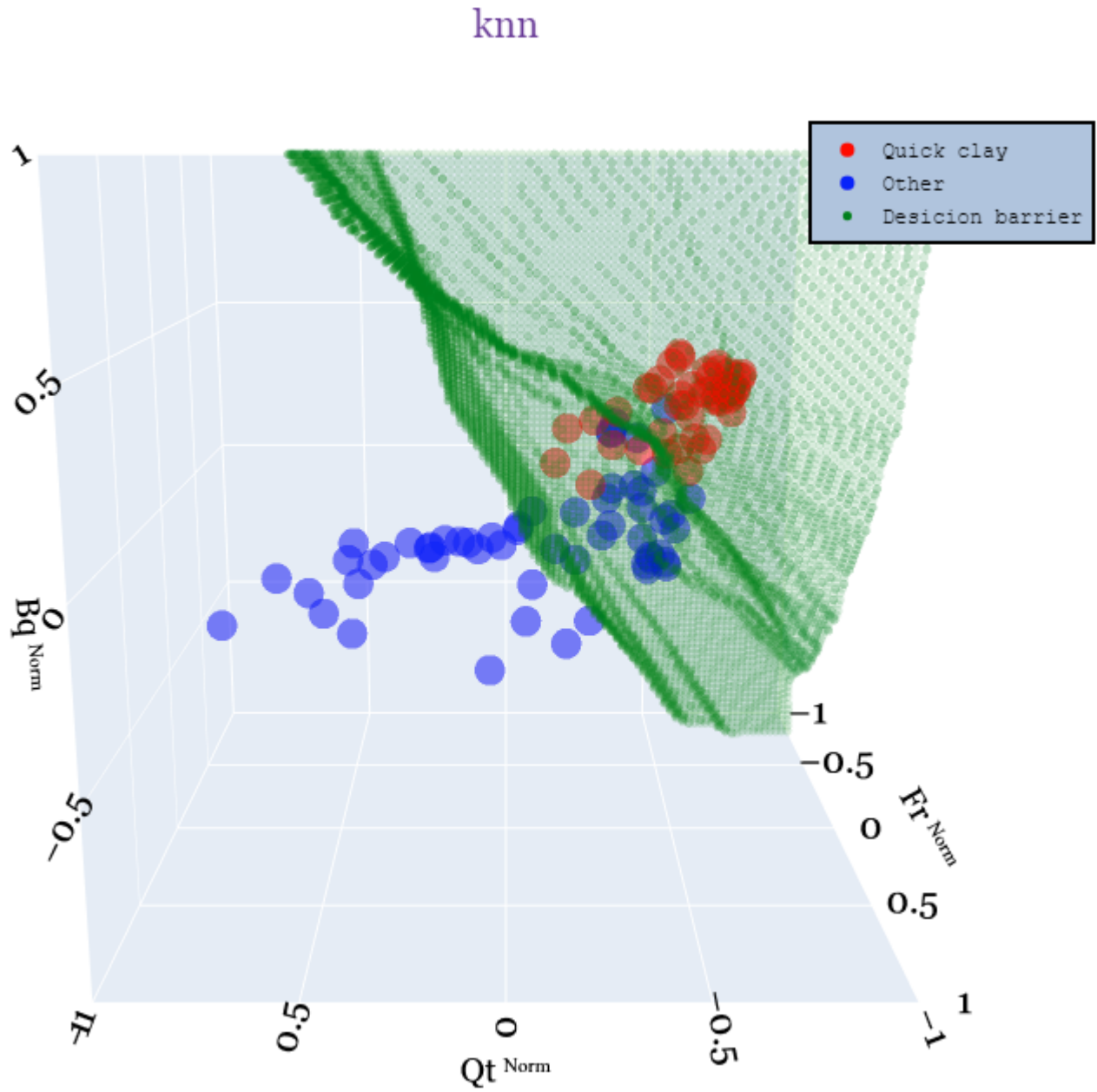


Figure F.10: 3D plot of KNN trained on dataset II (reduced). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

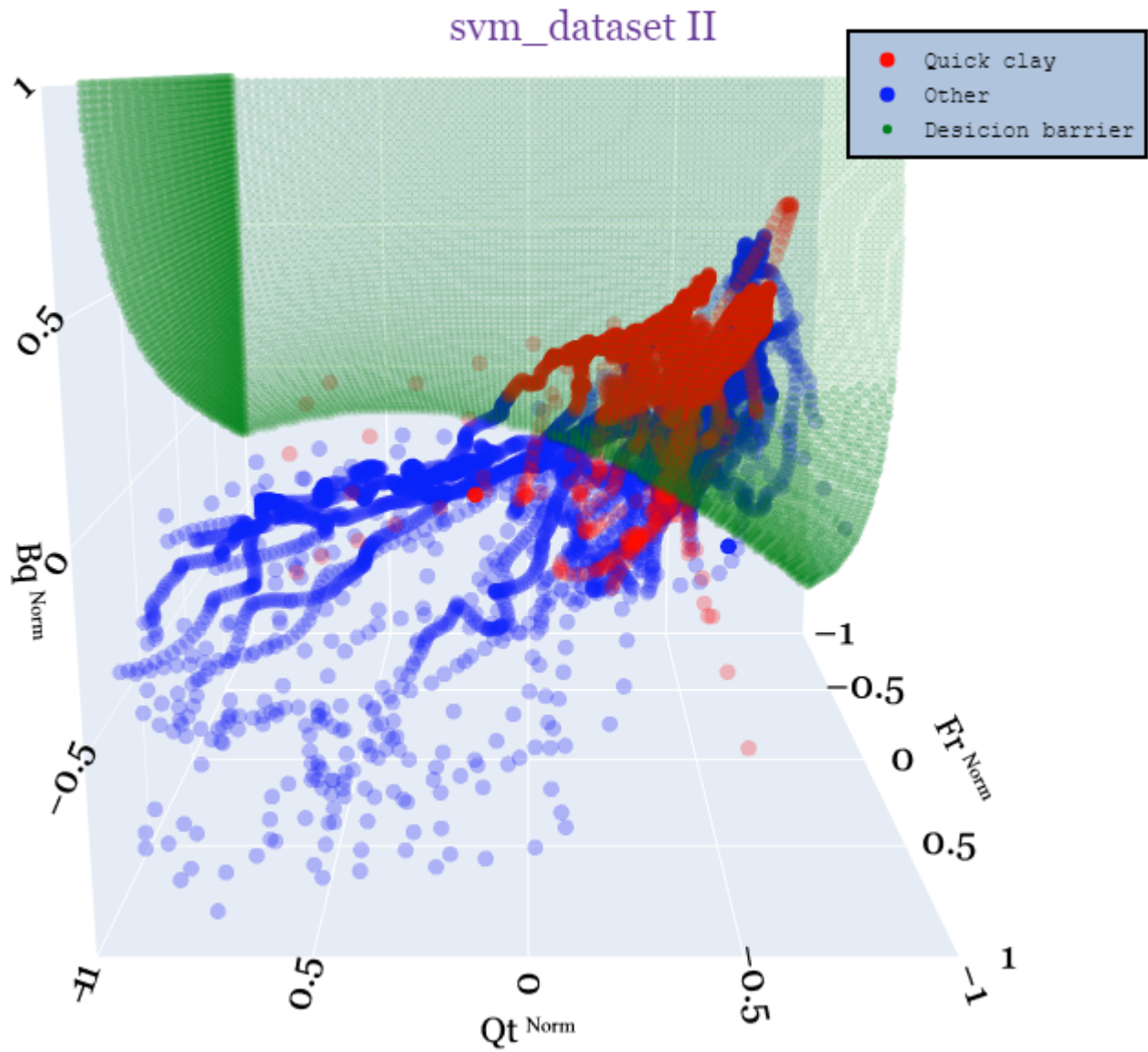


Figure F.11: 3D plot of SVM trained on dataset II (full). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

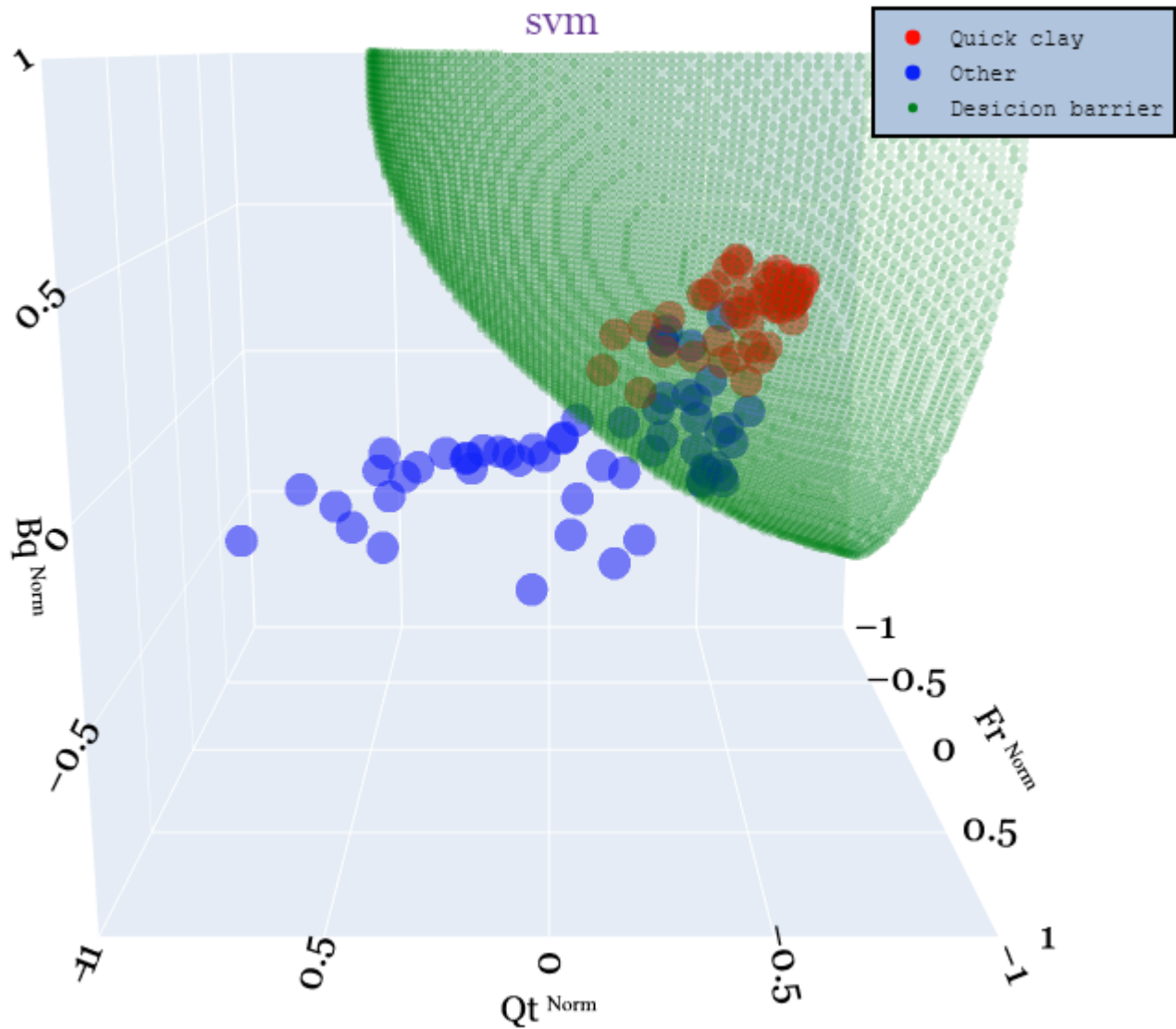


Figure F.12: 3D plot of SVM trained on dataset II (reduced). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

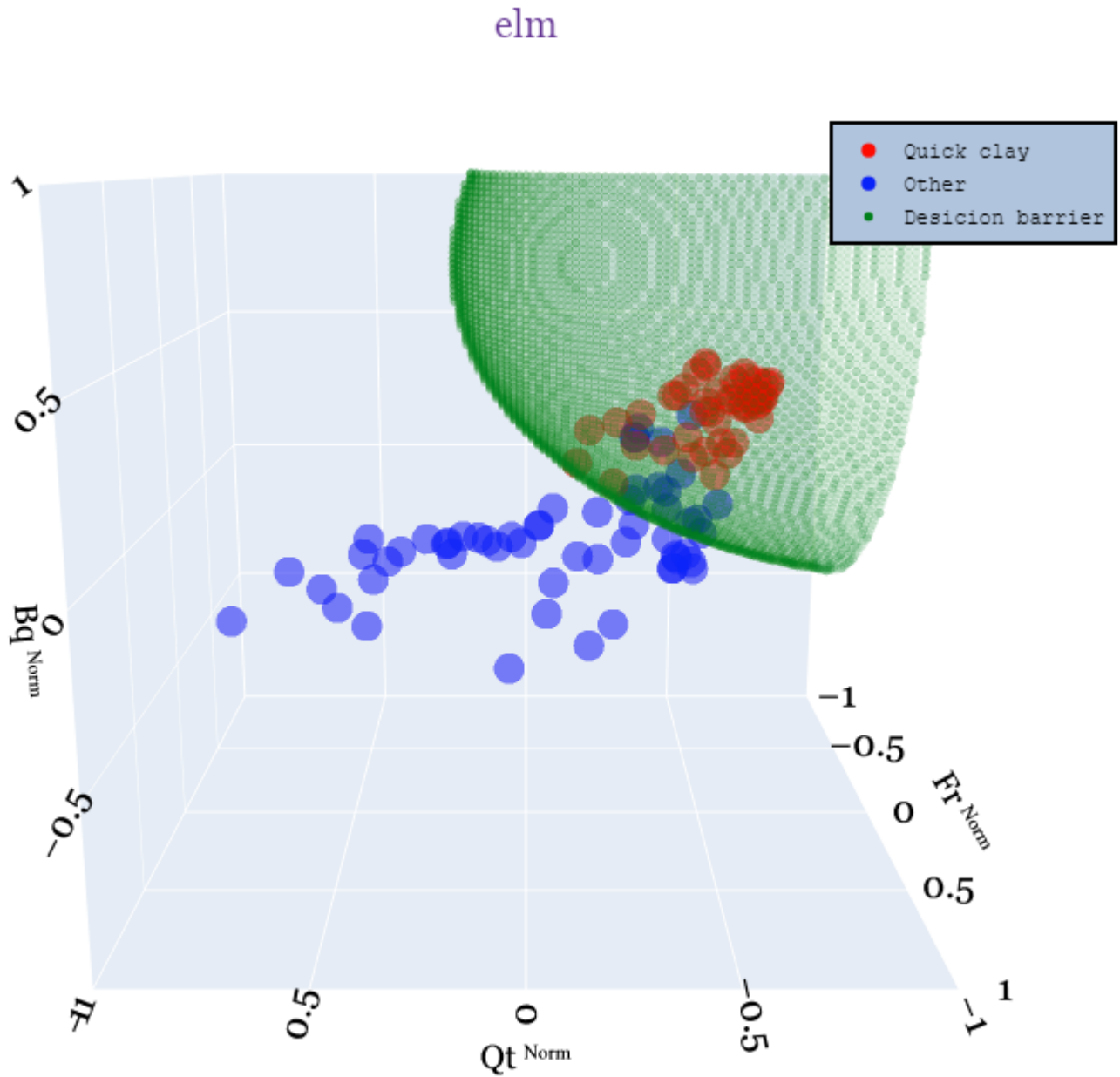


Figure F.13: 3D plot of ELM trained on dataset II (reduced). Blue points are training data labeled as non-quick clay, red points are training data labeled as quick clay and the green points are the decision barrier which marks where the algorithm starts classifying quick clay.

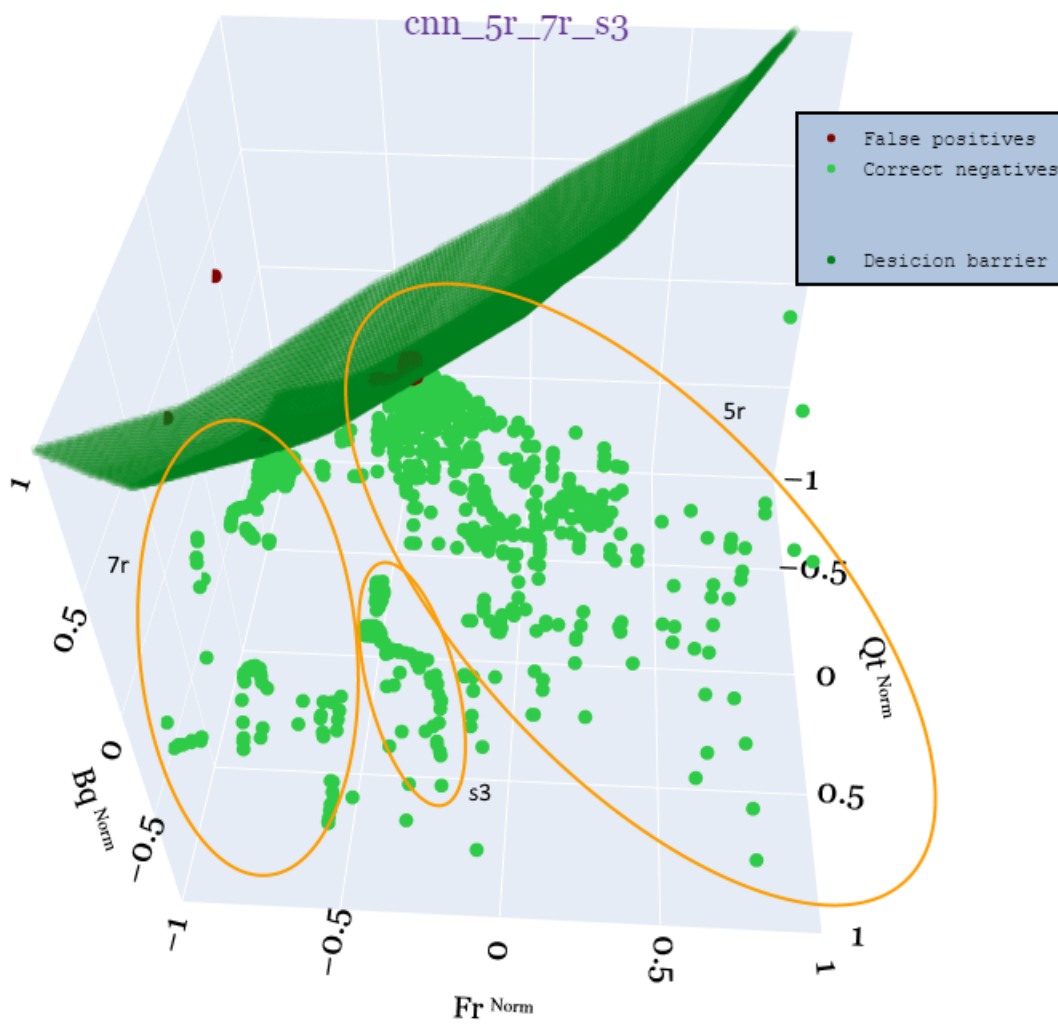


Figure F14: 3D plot where CPTu 5r, 7r and s3 from Saksvik are scattered. The CNN model is trained on dataset II (reduced)



