Stefan Rauter, BSc

CPT-Data Interpretation Using Machine Learning

MASTER THESIS

to achieve the university degree of

Diplom-Ingenieur

submitted to

Graz University of Technology

Supervisor
Assoc.Prof. Dipl.-Ing. Dr.techn., Franz Tschuchnigg

Institute of Soil Mechanics, Foundation Engineering and Computational Geotechnics

Graz, March 2021

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

…… Datum

…… Unterschrift

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Franz Tschuchnigg. Thank you for providing the topic of my master thesis and your outstanding guidance and commitment during this project.

Additionally, I want to thank Simon Oberhollenzer, who has supported me regarding the dataset this thesis is based on, and also Christian Knoll from the Institute of Signal Processing and Speech Communication for reviewing my methods.

Finally, my thanks go to my family. I want to express my deepest love and appreciation to my parents, Manuela and Helmut and in particular to my girlfriend Lisa. Thank you for your unconditional support!

Graz, March 2021 Stefan Rauter

Kurzfassung

CPT Daten Interpretation mit Machine Learning

Die Klassifizierung von Böden in Kategorien ähnlicher Eigenschaften ist einer der grundlegenden Prozesse in der Geotechnik. Nach Stand der Technik wird diese Klassifizierung auf Basis von verschiedenen zeit- und kostenintensiven Labor- und/oder Feldversuchen durchgeführt. Diese Untersuchungen sind essentiell für jedes Bauvorhaben und dienen als Grundlage für die Planung und Bemessung. Machine Learning könnte eine Schlüsselrolle in der Kosten- und Zeitreduktion für geotechnische Erkundungsmaßnahmen einnehmen, deshalb wird die grundlegende Fähigkeit von Machine Learning Algorithmen zur Bodenklassifizierung von CPT Daten untersucht. Um ein passendes Klassifizierungstool zu finden werden insgesamt 24 verschiedene Machine Learning Modelle basierend auf 3 verschiedenen Algorithmen (Artificial Neural Network, Support Vector Machine und Random Forest) mithilfe einer Datenbank von 1339 CPTs trainiert und getestet. Als Input „Features“ dienen verschiedene Kombinationen von direkten Messdaten (Spitzendruck q_c, Mantelreibung f_s, Reibungsverhältnis R_f, Tiefe d) mit „definierten“ Daten (nicht direkt gemessene Daten wie totale und effektive Vertikalspannungen σ_v und σ'_v sowie der hydrostatische Porenwasserdruck u_0). Als „Targets“ dienen Bodenklassen, basierend auf Korngrößenverteilung sowie Bodenverhaltensklassen nach Robertson (1990, 2009, 2016). Die verschiedenen Modelle werden auf Basis ihrer Genauigkeit und ihrer Trainingszeit miteinander verglichen. Die besten Resultate wurden unter Verwendung eines Random Forest Algorithmus erzielt. Für die Bodenklassen auf Basis von Korngrößenverteilung konnte eine Genauigkeit von 75 % und für die Bodenverhaltensklassen nach Robertson eine Genauigkeit von 97-99 % erzielt werden. Zusätzlich wurden die Random Forest Modelle verwendet, um Bodenschichten und -klassen von neuen CPT Daten aus Österreich und den Niederlanden zu klassifizieren. Die Ergebnisse sind durchwegs akkurat und zeigen, dass Machine Learning ein nützliches Tool sein kann, um den Prozess der Bodenklassifizierung zu verbessern.

Abstract

CPT-Data Interpretation Using Machine Learning

The classification of soils into categories with a similar range of properties is one of the fundamental procedures in geotechnical engineering. Currently, this classification is based on various types of cost and time intensive laboratory and/or in-situ tests. These soil investigations are essential for each individual construction site and have to be performed prior to the design of a project. Since Machine Learning could play a key role in reducing the costs and time needed for a suitable site investigation program, the basic ability of Machine Learning models to classify soils from Cone Penetration Tests (CPT) is evaluated. To find an appropriate classification model, 24 different Machine Learning models based on three different algorithms, namely Support Vector Machine, Artificial Neural Network and Random Forest are built and trained on a dataset consisting of 1339 CPTs. As input features, different combinations of direct cone penetration test data (tip resistance q_c, sleeve friction f_s, friction ratio R_f, depth d) combined with "defined", thus not directly measured data (total vertical stresses σ_v, effective vertical stresses σ'_v and hydrostatic pore pressure u_0) are used. As targets, standard soil classes based on grain size distributions and soil classes based on Soil Behaviour Types acc. to Robertson (1990, 2016 and 2009) are applied. The different models are compared with respect to their prediction performance and the required learning time. The best results for all targets are obtained with models using a Random Forest classifier. For the soil classes based on grain size distribution an accuracy of about 75 % and for soil classes according to Robertson an accuracy of about 97-99 % is reached. In addition, the models are used to predict soil classes and strata from unseen CPT tests from sites in Austria and the Netherlands. The predictions yield accurate results and show that Machine Learning can improve the process of soil classification.

Table of contents

List of symbols and abbreviations

Small letters

q_c	[MPa]	tip resistance
q_t	[MPa]	corrected tip resistance
f_s	[MPa]	sleeve friction
u_0	[MPa]	hydrostatic pore pressure

Capital letters

ANN	[-]	Artificial Neural Network
RF	[-]	Random Forest
SVM	[-]	Support Vector Machine
CPT	[-]	Cone Penetration Test
CPTu	[-]	Cone Penetration Test with pore pressure
SCPT	[-]	Seismic Cone Penetration Test
SCPTu	[-]	Seismic Cone Penetration Test with pore pressure
Q_t	[-]	normalized cone resistance
Q_{tn}	[-]	updated normalized cone resistance
I_c	[-]	soil behaviour type index
I_B	[-]	modified soil behaviour type index
B_q	[-]	pore pressure parameter ratio
R_f	[-]	friction ratio
F_r	[-]	normalized friction ratio

Small Greek letters

σ_v	[MPa]	total vertical stresses

σ'_v	[MPa]	effective vertical stresses

1 Introduction

Artificial Intelligence and its subfield Machine Learning has become a constant companion in our daily life. Whether it's the ability of an email program to identify spam or a streaming portal recommending new movies one might like, many processes would not be possible without the field of Artificial Intelligence. But besides that, Machine learning has also become a useful tool to interpret large datasets through regression and classification in various fields of research.

One of the fundamental procedures in geotechnical engineering is to classify soils into categories with similar physical properties. This classification is usually done through costly and time intensive laboratory and/or field tests during the preliminary stages of a construction project. This often leads to high costs before one even knows if the planned project is feasible.

The Cone Penetration Test (CPT) is a cost-effective field test and over the last years, various Soil Behaviour Type Charts were developed to classify and identify soil strata from the measured test data. In 2021, Oberhollenzer et. al. published a dataset consisting of measured and interpreted data from 1339 CPT. Additionally, soil classes based on grain size distribution from adjacent boreholes were added to 490 of them.

The aim of this master thesis is to evaluate the ability of a Machine Learning model to classify soils from Cone Penetration Test data. Therefore, different models based on different learning algorithms are built, tested and compared in order to find the best suitable model for this task.

1.1 Objectives

After a literature study to identify the state of the art of Machine Learning in geotechnical engineering, the objectives of this thesis are to evaluate the basic ability of a Machine Learning Model to classify soils from CPT data, once targeting Soil Behaviour Types according to Robertson (1990, 2009, 2016) and once targeting soil classes based on grain size distribution according to Oberhollenzer et. al. (2021). Therefore, different models based on learning algorithms like Support Vector Machine, Artificial Neural Network and Random Forest are built and compared with respect to their prediction accuracy and training time. The models which yield the best performance are then used to predict soil classes and generate a soil model from unseen CPT data from sites in Austria and the Netherlands.

1.2 Approach

Based on three different algorithms, namely the Support Vector Machine, the Artificial Neural Network and the Random Forest, 24 different learning models are built using 2 different sets of input features and 4 different target classes. As target classes the soil classes after Oberhollenzer et. al. (2021) and the 3 different Soil Behaviour Types according to Robertson (1990, 2009, 2016) are used. The comparison and ranking of the models are predominantly based on the prediction accuracy and secondarily on the necessary training time.

All models are built on a MacBook Pro 13” 2018 (CPU: INTEL core i5 2,3 GHz quad core, RAM: 8 GB, GPU: Intel Iris Plus Graphics 655 1536 MB) using the *SPYDER* python environment. The applied Machine Learning algorithms are part of the opensource software library of *scikit-learn*.

2 Literature Study

2.1 Application of Machine Learning in soil sciences

With an increasing amount of high-quality data sets available in various fields of research, the application of Machine Learning for data science increased quickly, especially over the last 10 years. Machine learning and its subsets (e.g., Deep Learning) are used in several fields of research in soil sciences reaching from crop prediction in agricultural sciences to determine physical soil properties like bulk density in geotechnical engineering. With a raising number of available open-source algorithms (e.g., scikit-learn and TensorFlow in Python) the research output increased rapidly. Figure 1 shows the number of publications in this field over the last 20 years.

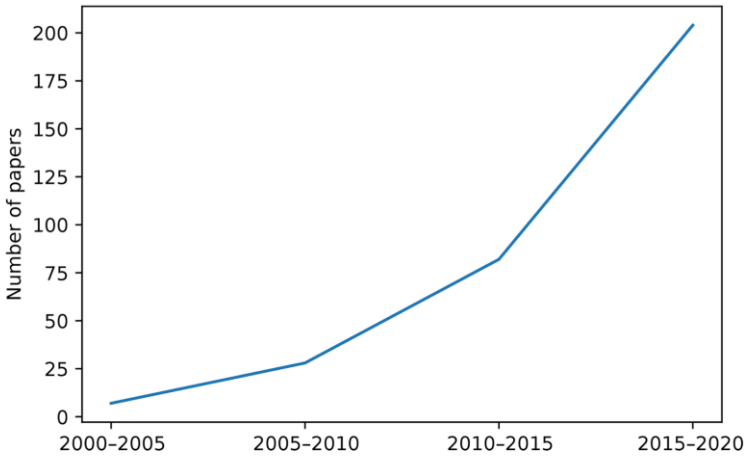

Figure 1: Number of publications regarding Machine Learning in soil sciences. (Padarian, Minasny, & Mcbratney, 2020)

The most common algorithms used in current research regarding Machine Learning in geotechnics are Artificial Neural Networks (ANN), Linear and Polynomial Regression (LR & PR), Support Vector Machines (SVM), Decision Trees and Random Forest (RF). The aforementioned techniques are mainly developed in Matlab (e.g. Wang, et al., 2019) and Python (e.g. Tsiaousi, et al., 2018) environment. The operation principle and application of these approaches on different problems is described below.

2.2 Frequently used algorithms

The description of the different algorithms in this chapter is based on the book “Python Machine Learning” by Raschka and Mirjalili (2019).

2.2.1 Artificial Neural Networks – ANN

The principle of Artificial Neural Networks is based on the mechanism of the neural network of a human brain. A flow chart of the algorithm is visualized in Figure 2. A neural network consists basically of three different layer types. The first one is the input layer where pre-processed input data is surrendered to the algorithm. The second layer type is called hidden layer, here the input values are combined with the weights and the third layer type represents the output. A Network with more than one hidden layer is called Deep Artificial Neural Network. In the forward propagation each layer hands the data to the next one until the output is computed. During training of the algorithm, the output values are compared to the real ones of the training set and the resulting error is calculated. With this information the weights of the hidden layer are adjusted to get a proper result, this is called backpropagation.

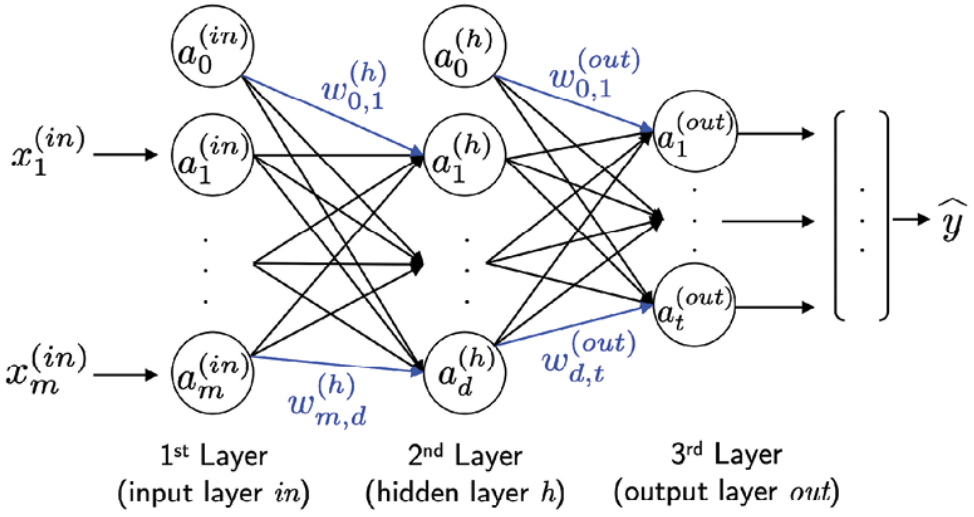

Figure 2: Pattern of an ANN algorithm (Raschka & Mirjalili, 2019)

Artificial Neural Networks have been used to classify soils from sounding-data (Reale, Gavin, Librić, & Jurić-Kaćunić, 2018), to predict bearing capacity of piles (Samui, 2008), the cone resistance of CPT tests (Erzin & Ecemis, 2017) or identify soil parameter (Puri, Prasad, & Jain, 2018). Recent research shows that with enough data available, neural networks outperform other Machine Learning approaches (e.g. Regression) in accuracy, but with less quality data available, other algorithms (e.g. SVM) might lead to better results.

2.2.2 Regression and Classification

Regression and Classification are tools of supervised learning. Regression is used to find a relationship between input (explanatory) and output (outcome) variables to predict an outcome. Regression is defined as linear, if the relationship can be described with a straight line or polynomial, if higher order functions are needed.

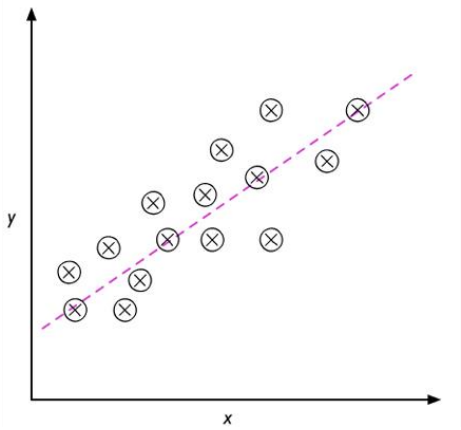

Figure 3: Linear regression (Raschka & Mirjalili, 2019)

With Classification the examined variables can be divided into classes based on past observations (e.g. classification of spam mails). Similar to regression the order of the classification rule depends on the relationship between input and output data.

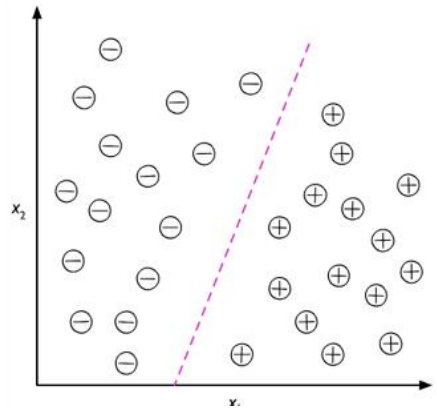

Figure 4: Classification of data in two classes with a linear function (Raschka & Mirjalili, 2019)

Regression analysis has been used recently for the prediction of soil parameter (Puri, Prasad, & Jain, 2018), estimating optimal additive content for soil stabilization (Gajurel, Mukherjee, & Chittoori, 2019) and in more advanced models (e.g. evolutional polynomial regression EPR) for an artificial intelligence based finite element method (Javadi, Mehravar, Faramarzi, & Ahangar-Asr, 2009), for the settlement and bearing capacity analysis of shallow foundations (Shahin, 2015) and for analysis of the correlation of soil properties (Jin & Yin, 2020).

2.2.3 Support Vector Machine – SVM

A Support Vector Machine (SVM) can be described as an advancement of the classification or regression. The optimization objective of a SVM is to maximize the margin, which is defined as the maximum distance between the variables of the training-set and the separating hyperplane. The training examples on the edge,

which define the margin are called the support vectors. The advantage of this approach is that decision boundaries with large margins tend to have lower generalization errors.

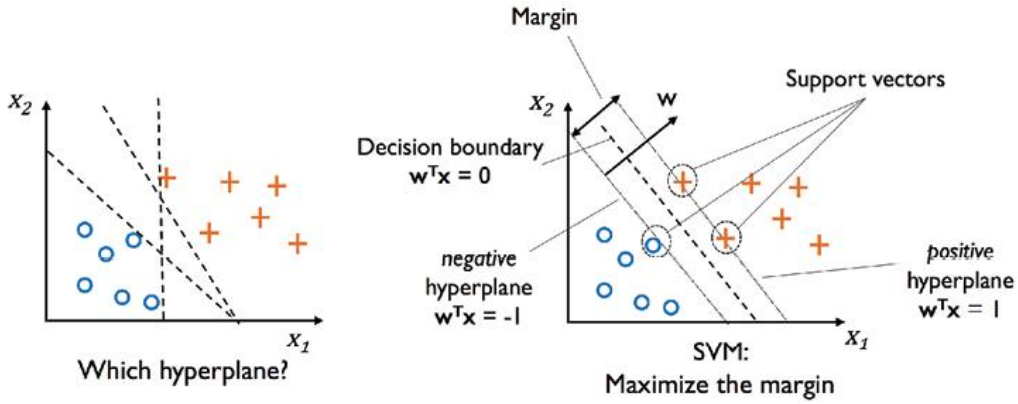

Figure 5: Principle of an SVM algorithm (Raschka & Mirjalili, 2019)

Support Vector Machines have been used to predict the bearing capacity of bored piles from CPT data (Alkroosh, Bahadori, Nikraz, & Bahadori, 2015), soil compressibility (Kirts, Panagopoulos, Xanthopoulos, & Nam, 2018) or soil classification (Harlianto, Adji, & Setiawan, 2017). In general, SVM perform with higher accuracy than ANNs if less data is available.

2.2.4 Decision Trees and Random Forest

A Decision Tree is a non-parametric supervised learning method that can summarise decision rules from a series of data with features and labels and use the structure of the tree to present these rules to solve classification and regression problems (Zhang , Wang, & Wu, 2019). The advantage of Decision Trees is that the prediction is comprehensible, and the influence of the deciding parameter can be identified immediately. A schema of a decision tree is shown in Figure 6.

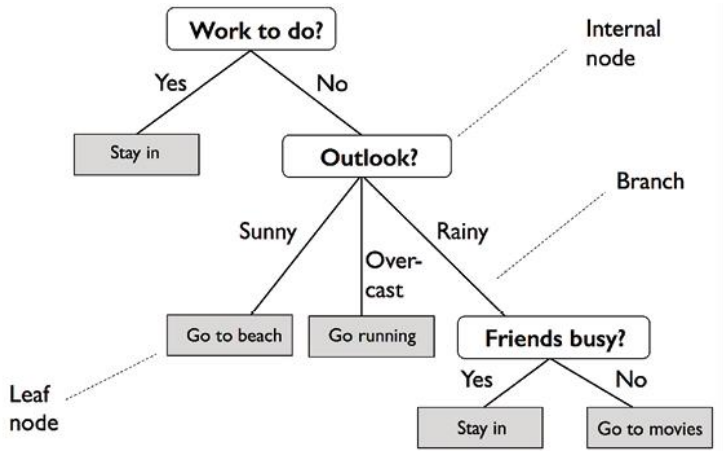

Figure 6: Simple example of a decision tree (Raschka & Mirjalili, 2019)

A combination of various Decision Trees is called Random Forest (RF). With RF, the results of different Decision Trees for the same investigated problem are

averaged. This leads to a more robust model which is less vulnerable to overfitting. Overfitting means that the model fits better to training data than to unseen data, for more information see chapter 4.1.

In recent research, Random Forest and Decision Trees have been used to predict geotechnical parameters (Puri, Prasad, & Jain, 2018), the undrained shear strength (Wu, Zhong, Zhang, Li, & Wang, 2020) and the pile drivability (Zhang , Wang, & Wu, 2019).

2.3 Recent publications regarding Machine Learning for soil classification

One of the fundamental criteria for proper geotechnical design of any under- or on the ground structure is comprehensive knowledge of the type, composition and mechanical properties of the subsurface. To gather enough information about the subsurface conditions, an appropriate and usually cost-intensive testing program has to be planned and executed for each construction project. Based on the laboratory test results various soil classification systems using particle size distribution and Atterberg limits, e.g., the North American Unified Soil Classification System USCS (ASTM D2487-17e1, 2017) or the European Soil Classification System ESCS (EN ISO 14688-2, 2019) have been established. To avoid costly laboratory tests and to gain information about undisturbed in-situ conditions for the classification of soil types various classification systems based on in-situ tests i.e., Cone Penetration Test (CPT) were introduced in the past, e.g. soil behaviour types according to Robertson (1990). The main disadvantage of soil classification based on in-situ tests is that the subsoil conditions aren't visible for the engineering judgement and therefore misclassification of soils which behave nearly similar while testing, but are actually different, is possible. The state of the art and current research results in the interpretation of test results and soil classification with the help of soft computing methods is described in the following chapters.

2.3.1 Artificial Neural Network – ANN

The application of Artificial Neural Networks for soil classification has been used recently to interpret soil stratigraphy and determine shear wave velocity (Tsiaousi, Travasarou, Drosos, Ugalde, & Chacko, 2018) and to classify soils according to USCS from CPT data (Reale, Gavin, Librić, & Jurić-Kaćunić, 2018).

Tsiaousi (2018) used an Artificial Neural Network based on multilayer perceptron (MLP) to interpret soil stratigraphy and estimate shear wave velocity based on CPT test data. A multilayer perceptron is able to distinguish between data which is not

linearly separable (more information about the function principle of a MLP is provided in chapter 4.2.1.). For soil classification, a supervised ANN using classification based on backpropagation was applied. Test data of 15 CPTs, where seven soil strata were identified manually, were used for training and validation of the algorithm. 12 for training and 3 for evaluation of accuracy. For the estimation of shear wave velocity, 11 seismic CPTs were used, 9 for training and 2 for validation. The input values which leaded to the highest accuracy were the cone tip resistance q_c, the sleeve friction f_s, the friction ratio R_f, and the elevation. The input features were standardized and normalized because the range of values of the input features varied significantly. This resulted in a more robust prediction. The effect of imbalanced data sets was considered by applying a sensitivity analysis which implements under- and oversampling techniques. Under- and oversampling are part of resampling techniques to avoid under- or overrepresented classes in a dataset, more information about class balance is provided in chapter 4.1.6. To avoid many interchanges in classified soil layers a minimum thickness of each layer was determined. The algorithm classified soil stratigraphy with an accuracy of about 90% and determined the shear wave velocity with a r^2-score of 0.92 (best is 1.0).

Reale (2018) used an ANN backpropagation algorithm for soil classification according to USCS based on CPT test data. Two feed forward neural networks trained with Levenberg-Marquardt backpropagation algorithm were developed to estimate on one hand the fines content and on the other hand the liquid limit and plasticity index. Both consist of three hidden layers and two input nodes for the sleeve friction and tip resistance. With these outputs, the soil is classified according to USCS or ESCS. Overall, 216 pairs of CPT and laboratory test data sets were available, all of them applied for the estimation of the fines content and 176 were used for the liquid limit and plasticity index. The classification with the neural network resulted in an r^2-score of 0.95 for the fines content, 0.85 for the liquid limit and 0.78 for the plasticity index (best is 1.0). Almost 90% of the soils were classified correctly and all in all the ANN classified the samples with a better performance compared to other published correlations.

2.3.2 General Regression Neural Network – GRNN

The General Regression Neural Network is a multi-layer feed forward neural network that performs general regression analysis directly from sample data. It computes the Euclidean distance (shortest distance between two points, e.g. in 2D space it is the length of a straight line between two points) separating the input values of a given testing case x_i from those of each training case x_{ij} and then finds the weighted mean of the target values of the training cases y_i closest to the testing case to predict the network output $\hat{y}$. A draw-back of this algorithm is that it is not capable of extrapolation which means that a point of a certain testing set must be in the range of the training data. (Kurup & Griffin, 2006)

Kurup & Griffin (2006) used the GRNN to classify soils from CPT test data. For training and testing the algorithm 142 data sets were available. 100 were used for training and 42 for testing, 10% of the training sets were used for fine tuning of the network. In data pre-processing the results of the CPT tests were scaled between -1 and 1. The applied network predicts the probability of a soil type in a specific layer i.e., for a layer between -1.2m and -2.5m the result yields to 70% sand, 10% silt and 20% gravel, therefore the estimated soil type for this layer is sand. The GRNN predicted about 86% correct with respect to the sample size.

Goh (1999) used the GRNN to predict the coefficient of consolidation c_v and to classify soils from particle size distributions. 60 patterns of laboratory particle size distribution were used for training and 10 were used for testing, additionally, 23 patterns were used for the prediction of c_v. The samples for training consisted of particle size distribution and the determined soil type. The algorithm yielded c_v values with good agreement with the validation data and all soil types were classified correctly.

2.3.3 Bayesian Statistical Models

In Bayesian Statistics, the likelihood of a resulting parameter is defined based on prior knowledge, i.e., the estimation of a parameter y (posterior) based on a set of parameter x is done by the combination of a maximum likelihood function with an already known prior (e.g., a friction angle of the soil in the prospected area must be between 30-35 degrees). The outcoming posterior can be set as a new prior for the combination with new input values. Bayesian statistics were used for several statistical soil classification approaches based on in-situ tests, e.g., (Wang, Huang, & Zijun, 2013).

Wang (2019) developed an unsupervised learning approach based on a Bayesian inferential framework (contrary to common statistics, Bayesian statistics introduce a prior belief or knowledge to the calculation) which used CPT data to find soil strata and classify the different layers according to Robertson's soil behaviour type chart (1990). The approach is divided into two major parts, first a pattern detection approach infers a spatial pattern in physical space and a statistical pattern in feature space from the input dataset, while in the second part the patterns get transformed into sets of multiple soil layers with different soil behaviour types. The advantage of this approach is the combination of probabilistic soil classification combined with spatial consistency and statistical similarity. The approach has been validated by two numerical studies and indicated accurate results. This method is more complex compared to other studies because additionally to properties of the feature space (q_c, Q_t, et.) it takes properties of physical space, e.g., depth, to account for the classification. The code is available for download under the following link: *https://github.com/hwang051785/pyCPT*

3 Dataset

3.1 Cone Penetration Test – CPT

The content provided in this chapter is based on the book “Cone Penetration Testing” by Lunne, et al. (1997)

In a cone penetration test (CPT), a cone is pushed vertically into the subsoil under a constant rate (usually 2 cm/s) and the resisting forces generated by the penetration of the cone are measured permanently in terms of tip resistance q_c and sleeve friction f_s. In a “piezocone” test (CPTu), pore water pressures (u_1, u_2, u_3) are measured to gain information about groundwater conditions and permeability (different pore pressures at u_1, u_2 and u_3) of the subsoil. In a seismic cone penetration test (SCPT or SCPTu), shear wave velocities V_s are measured at different depth (usually every 50 cm or 100 cm) additionally.

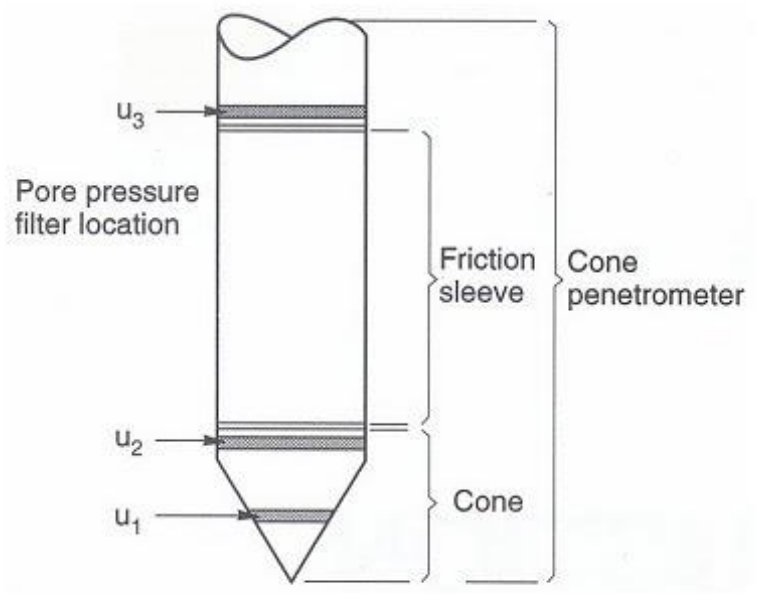

Figure 7: Schematic sketch of the cone (Lunne, Robertson, & Powell, 1997)

Cone penetration tests are mainly applied for the determination of soil strata and identification of soil types, the estimation of geotechnical parameter, thus, to provide inputs for the geotechnical design. The main advantages are:

- continuous generation of data,
- repeatable and reliable penetration data,
- cost efficient sampling
- test data of undisturbed in situ soil conditions

Table 1: Test types and measured values

Test type	Measurements
CPT	q_c, f_s
CPTu	q_c, f_s, u_2
SCPT	q_c, f_s, V_s
SCPTu	q_c, f_s, u_2, V_s

The main disadvantage of CPT is that the composition of the subsoil is not visible for the engineer, therefore, a subsurface exploration program with cone penetration tests is usually complemented with core drillings, sampling and laboratory tests to verify correlations (e.g. for geotechnical parameter) or to evaluate the soil behaviour under proposed site-specific future loading conditions.

3.1.1 Testing and data acquisition

The recommended rate of penetration is 20 mm/s. Due to dissipating excess pore water pressures the soundings should be as continuous as possible. The interval of readings is usually around 10-50 mm to obtain sufficient detailed data. Figure 8 provides a comparison of the tip resistance q_c, the sleeve friction f_s and the porewater pressure u_2 with the soil classification based on grain size distribution from adjacent core drillings. In this picture, the influence of changes in subsoil strata are visible in the data from the CPT measurements. (Oberhollenzer , et al., 2021)

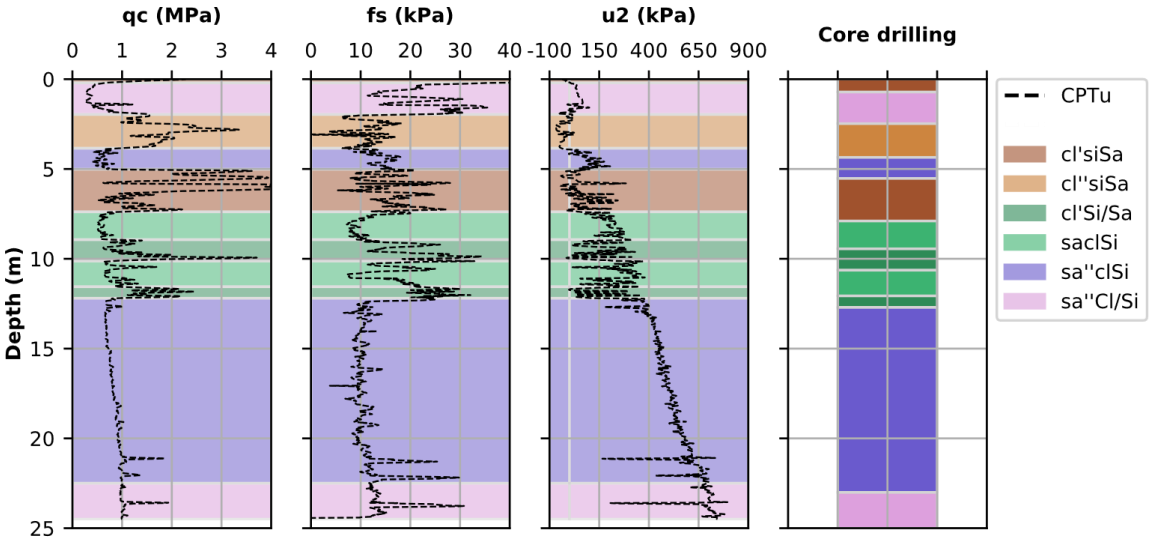

Figure 8: Comparison of CPT measurements with soil classification based on grain size distribution (Oberhollenzer , et al., 2021)

3.2 Database “CPT_PremstallerGeotechnik”

3.2.1 Source

The dataset “Cone Penetration Test Dataset Premstaller Geotechnik” was published in 2021 (Oberhollenzer , et al., 2021) by the Institute of Soil Mechanics, Foundation Engineering and Computational Geotechnics and the Institute of Rock Mechanics and Tunnelling at Graz University of Technology in cooperation with the company Premstaller Geotechnik ZT GmbH. It is available online and can be downloaded from the following address:

https://www.tugraz.at/en/institutes/ibg/research/computational-geotechnics-group/database/

The dataset consists of test data from 1339 cone penetration tests (CPT, CPTu, SCPT, SCPTu) from various sites in Austria and the south of Germany. All of the test data is classified in soil behaviour type charts according to Robertson (1991, 2009, 2016). Additionally, 490 of them were classified via grain size distribution from adjacent core drillings. More information related to the dataset is given in Oberhollenzer, et.al., (2021).

3.2.2 Features

The database consists of 28 feature columns and 2.516.978 rows with test and interpreted data. Table 2 provides an overview of the features, their meaning, type and amount of available data. The features are distinguished in three different types, where “Defined” means that the value, or class of this feature is defined by an engineer, “Test data” means that it is measured raw data from in situ tests without any human interpretation and “Empirical correlation” means that these values are determined after empirical relationships according to published literature (e.g. Robertson 1986) from the defined and measured data. The column “Amount of data” describes the quantity of non-null values for each feature but does not represent the amount of suitable data, e.g. in column “Oberhollenzer_classes” 1.030.569 rows have an entry and therefore 1.486.510 are empty. To eliminate measurement errors threshold values of -100 and +10.000 were defined and measured points which are exceeding these numbers are left blank.

Table 2: Features of dataset

Abbreviation	Name	Type	Amount of data
ID	ID of Test	Defined	2516979
test_type	CPT, CPTu, SCPT or SCPTu	Defined	2516979
basin_valley	Location of test site	Test data	2516979
Depth (m)	Depth of measured data	Test data	2516979
qc (MPa)	Tip resistance	Test data	2516162
fs (kPa)	Sleeve friction	Test data	2515446
u2 (kPa)	Generated porewater pressure	Test data	798379
Vs (m/s)	Shear wave velocity	Test data	3981
qt (MPa)	Cone resistance corrected for water effects	Empirical correlation	2516853
Rf (%)	Friction ratio	Empirical correlation	2515663
γ (KN/m3)	Specific weight	Defined	2516852
σ,v (kPa)	Total vertical stress	Empirical correlation	2516852
u0 (kPa)	Hydrostatical stress due to groundwater horizon	Defined	2516852
σ',v (kPa)	Buoyant vertical stress	Empirical correlation	2516852
Qt (-)	Normalized cone resistance	Empirical correlation	2516315
Qtn (-)	Updated normalized cone resistance	Empirical correlation	2516765
Fr (%)	Normalized friction ratio	Empirical correlation	2514275
Bq (-)	Pore pressure parameter	Empirical correlation	798424
U2 (-)	Normalized excess pore pressure	Empirical correlation	798429
SBT (-)	Soil behaviour type (non-normalized)	Empirical correlation	2516853
SBTn (-)	Soil behaviour type (normalized)	Empirical correlation	2516851
Mod. SBTn (-)	Modified soil behaviour type (normalized)	Empirical correlation	2516851
n	Stress exponent	Empirical correlation	2516851
Ic (-)	Soil behaviour type index	Empirical correlation	2516851
Ic-SBT (-)	Soil behaviour type index	Empirical correlation	2516852
Ib (-)	Modified soil behaviour type index	Empirical correlation	2514622
EN_ISO_14688_classes	Soil classification with grain size distribution acc. to EN ISO 14688	Defined	880960
Oberhollenzer_classes	Soil classification with grain size distribution acc. Oberhollenzer 2020	Defined	1030469

Machine learning algorithms are usually not able to handle missing data (or null values) therefore, the dataset has to be adjusted for the proposed application, e.g. rows with missing data must be deleted or filled with synthetic data, this is part of the pre-processing and is discussed in chapter 4.1.

3.3 Classification of CPT data

3.3.1 Classification based on grain size distribution

The content provided in this chapter is mainly based on chapter 2 of the book “Handbuch Geotechnik” by Boley (Eigenschaften und Klassifikation von Böden, 2019)

The size of the soil grains is described by their equivalent grain diameter d in mm. The distribution of the different grain diameters is usually divided in 6 groups, provided in Table 3.

Table 3: Grain size groups according to “Handbuch Geotechnik” (Zou & Boley, 2019)

Name	Grain size range [mm]
Clay	< 0.002
Silt	0.002 - 0.063
Sand	0.063 - 2.0
Gravel	2.0 - 63.0
Stones	63.0 - 200
Blocks	> 200

The distribution of different grain groups is determined with laboratory tests. Samples with grain size > 0.063 mm are determined with a sieve and that with grain size < 0.063 mm by sedimentation. The resulting grain size distribution is usually displayed in grain size charts. In Figure 9 the grain size distribution for a soil sample based on sedimentation (B), sieve (A) and the combination of both (C) is displayed.

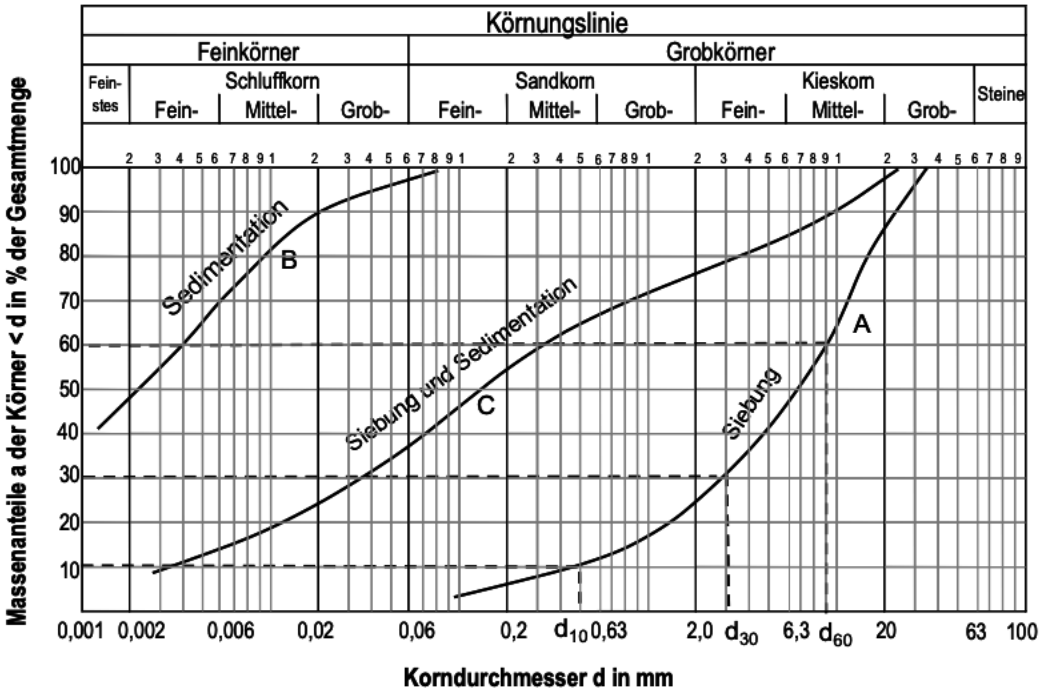

Figure 9: Grain size distribution chart according to "Handbook Geotechnik" (Zou & Boley, 2019)

Based on the grain size distribution, the soil is classified according to a standardized classification system, e.g. Unified Soil Classification System (ASTM D2487-17e1, 2017) or European Soil Classification System (EN ISO 14688-2). According to EN ISO 14688-2 samples where more than 50 % of mass fraction of particles are greater than 0.063 mm are described as coarse and their classification is predominantly determined based on the grain size distribution. For samples where more than 50 % is smaller than 0.063 mm, called fine grained soil, the determination is based on the plasticity (liquid limit w_L and plastic limit w_P),

compressibility, stiffness, etc. Figure 10 provides an overview of the soil classification process based on laboratory tests according to the European Soil Classification System. (EN ISO 14688-2, 2019)

Soil group	Quantification	Denomination into soil groups			Further subdivision as appropriate by
		Primary fraction (symbol)	Composite fractions		
very coarse	>50 % of particles by mass ≥200 mm	Boulders (Bo)	BOULDERS BOULDERS with cobbles	BOULDERS with finer soils	Requires special consideration
	>50 % of particles by mass <200 mm and ≥63 mm	Cobbles (Co)	COBBLES COBBLES with boulders	COBBLES with finer soils	
coarse	>50 % of particles by mass <63 mm and ≥2 mm	Gravel (Gr)	GRAVEL with cobbles GRAVEL sandy GRAVEL with cobbles	Sandy GRAVEL GRAVEL with clay and silt	Particle size distribution, Shape of grading curve, Relative density/ density index, Permeability
	>50 % of particles by mass <2 mm and ≥0,063 mm	Sand (Sa)	Gravelly SAND SAND	SAND with clay or silt	Mineralogy, Particle shape
fine	low plasticity or non-plastic	Silt (Si)	sandy SILT	sandy gravelly SILT sandy clayey SILT	Plasticity, Water content, Strength, Sensitivity, Compressibility, Stiff-ness, Clay mineralogy
			clayey SILT, silty CLAY		
	plastic	Clay (Cl)	Sandy gravelly CLAY Organic SILT Organic CLAY		
organic		PEAT (Pt) GYTTJA (Gy) DY (Dy) HUMUS (Hu)	Sandy PEAT sandy clayey GYTTJA		Requires special consideration
Anthropogenic soil		Made Ground	Placed without control	Synthetic material or Reworked natural mate-rials (such as crushed, grad-ed or washed materials	Requires special consideration As for natural soils
		Fill	Placed with control		

Cases requiring special consideration should be classified in accordance with national or project requirements (see, for example, EN 16907 standards).

Figure 10: Soil classification according to (EN ISO 14688-2, 2019)

3.3.1.1 Classification according to Oberhollenzer, et al. (2021)

Oberhollenzer, et al. (2021) defined 7 soil classes containing a specific range of grain sizes, reaching from gravely to clayey material. (displayed in Table 4). The classification is based on grain size distribution from adjacent core drillings. The assigned core drillings have a maximum distance of about 50 m to the in-situ tests.

The variation of height and thickness of soil strata due to the distance between borehole and CPT test is considered in this classification by manually adjusting the depth of the strata boundaries. Soil types which could not be assigned to the 7 classes are described as “ignored” and labelled as “0” in the dataset.

Table 4: Groups of Oberhollenzer_classes

Name	Grain size range	Mainly contents	Label
Group 1	Gr,sa,si’ → Gr,co	gravel	1
Group 2	Or,cl → Or,sa’	fine grained organic soils	2
Group 3	Or,sa → Or/Sa	coarse grained organic soils	3
Group 4	Sa,gr,si → Gr,sa,si	sand to gravel	4
Group 5	Sa,si → Sa,gr,si’	sand	5
Group 6	Si,sa,cl’ → Si,sa,gr	silt to fine sand	6
Group 7	Cl/Si,sa’ → Si,cl,sa	clay to silt	7
Ignored Group	--		0

3.3.2 Soil Behaviour Types – SBT according to Robertson

For the classification of soils from cone penetration test data, various soil behaviour charts were developed. Some of the most prevalent charts were published by Robertson (1990, 2009, 2016). In these charts the soil behaviour type is determined using non-normalized (q_c, q_t, f_s, R_f) and normalized parameters (Q_t, Q_{tn}, F_r, B_q) from CPT tests. In normalized parameters, the influence of the groundwater conditions on the tip resistance and friction ratio is taken to account, therefore piezocone tests (CPTu or SCPTu) are required. In the used dataset, u_0 was determined from adjacent bore holes or if no information was available the groundwater table was assumed to be at the top of ground level. Additionally, u_2 was assumed to be 0 for the calculation of normalized values for all tests were no porewater pressures were measured (CPT, SCPT).

The first soil behaviour type chart by Robertson was published 1986 and consisted of 12 behaviour types based on the friction ratio

$$R_f = f_s/q_c \tag{1}$$

and the corrected tip resistance

$$q_t = q_c + u_2 \times (1 - \alpha) \tag{2}$$

where q_t is the corrected tip resistance, u_2 is the generated porewater pressure above the cone and α is the cone area ratio. Instead of q_t, q_c can be used either for the tip resistance, but this will lead to slightly different results. (Oberhollenzer , et al., 2021)

In 1990, a soil behaviour type chart, using normalized parameters was published by Robertson which is shown in Figure 11.

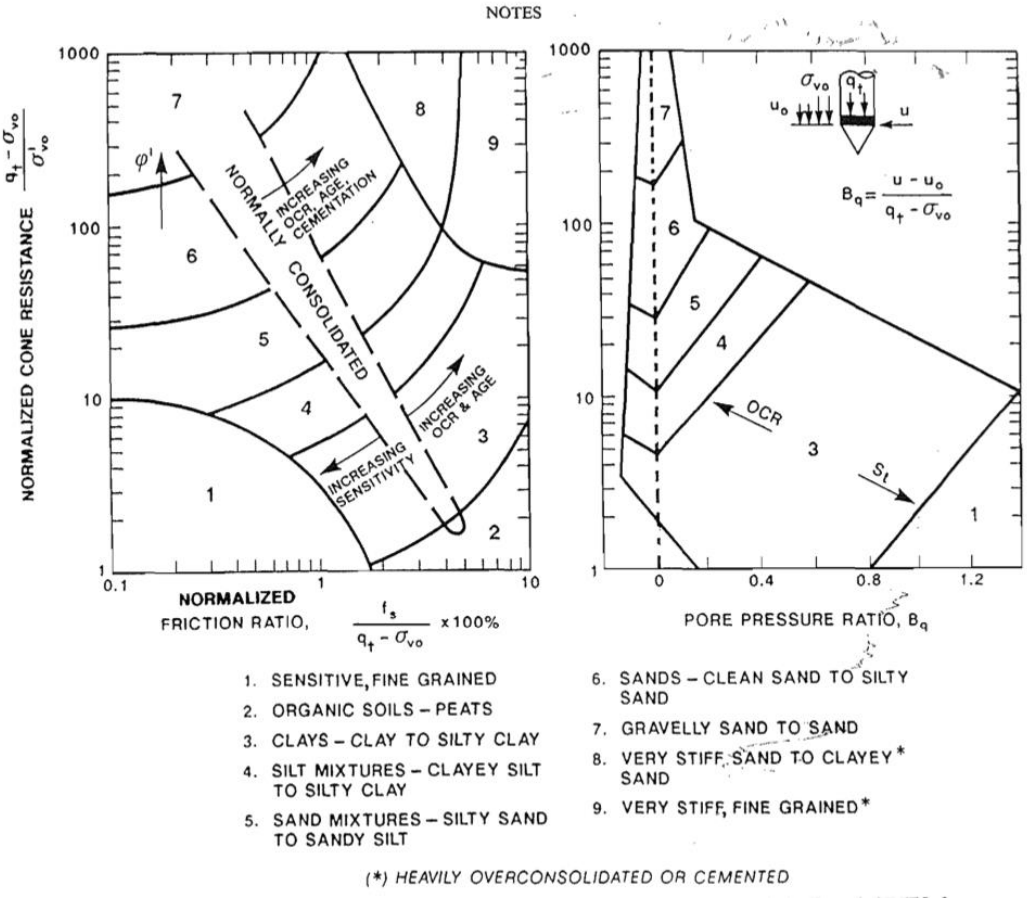

Figure 11: Soil behaviour type chart with normalized parameters according to Robertson (1990)

This chart distinguishes between 9 soil behaviour types using the normalized cone resistance:

$$Q_t = \frac{q_t - \sigma_{vo}}{\sigma'_{vo}} \tag{3}$$

The normalized friction ratio:

$$F_R = \frac{f_s}{q_t - \sigma_{vo}} \times 100\% \tag{4}$$

The pore pressure ratio:

$$B_q = \frac{u_2 - u_0}{q_t - \sigma_{vo}} = \frac{\Delta u}{q_t - \sigma_{vo}} \quad (5)$$

where u_0 is the in-situ pore water pressure, σ_{vo} is the in situ total vertical stress and σ'_{vo} is the in situ effective vertical stress. The main advantage of this chart is that the effects of water pressures and cone design are considered in the normalized values. One disadvantage is that measurements of the porewater pressure is necessary and therefore, CPTu or SCPTu tests are required. Soil behaviour type classifications based on this chart are labelled as "SBTn" in this thesis.

In 2009, Robertson published an updated version of his soil behaviour chart from 1986, where the 12 initial behaviour types are reduced and adjusted to the 9 types from the SBTn chart from 1990 (Robertson P. , 2009).

The updated chart is based on the dimensionless cone resistance:

$$\frac{q_c}{p_a} \quad (6)$$

Where p_a is the atmospheric pressure (1 bar = 100 kPa = 0.1 MPa) and with the friction ratio:

$$R_f = f_s/q_c \quad (7)$$

Both values are shown on log scales to provide the area where R_f < 1% too. The chart is displayed in Figure 12 and classifications based on this chart are labelled as "SBT" in this thesis.

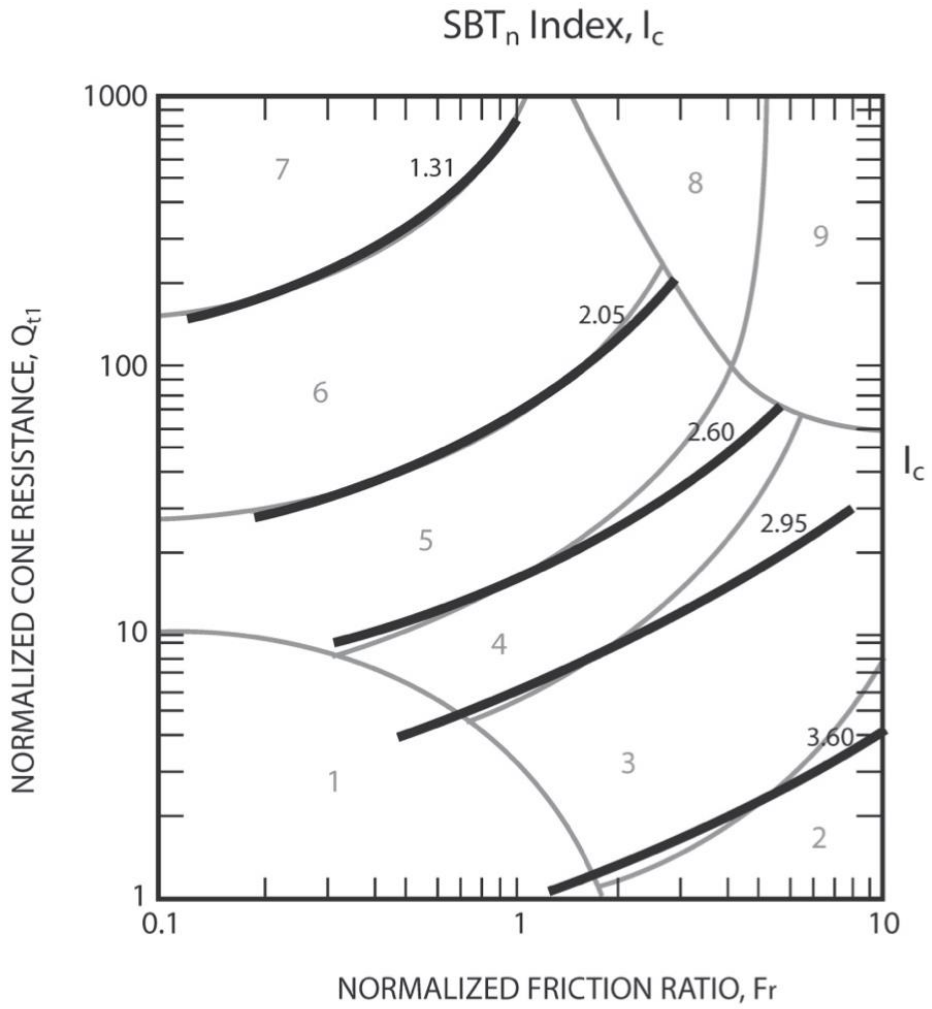

Figure 12: Updated soil behaviour type chart according to Robertson (2009)

Most of the published soil behaviour type classification systems classify soils based on textural descriptions like silt, clay, sand or gravel. In 2016, Robertson published a modified chart for the determination of the soil behaviour type which focuses more on the behaviour, e.g., dilative or contractive, of a certain soil than on the name (Robertson P. , 2016). The chart is based on the updated normalized cone resistance:

$$Q_{tn} = \left(\frac{q_t - \sigma_v}{p_a}\right)\left(\frac{p_a}{\sigma'_{vo}}\right)^n \tag{8}$$

where n is the stress exponent that varies with normalized Soil Behaviour Type, which is defined by

$$n = 0.381(I_c) + 0.05\left(\frac{\sigma'_{vo}}{p_a}\right) - 0.15 \tag{9}$$

where $n \leq 1$ and I_c is the soil behaviour type index:

$$I_c = \sqrt{(3.47 - logQ_t)^2 + (logF_r + 1.22)^2} \quad (10)$$

Figure 13 shows the modified behaviour type chart based on a modified behaviour type index:

$$I_B = \frac{100(Q_{tn} + 10)}{Q_{tn}F_r + 70} \quad (11)$$

The proposed behaviour types decreased to 7 and distinguish mainly between sand or clay like and contractive or dilative behaviour. This chart is labelled as "Mod.SBTn" in this thesis.

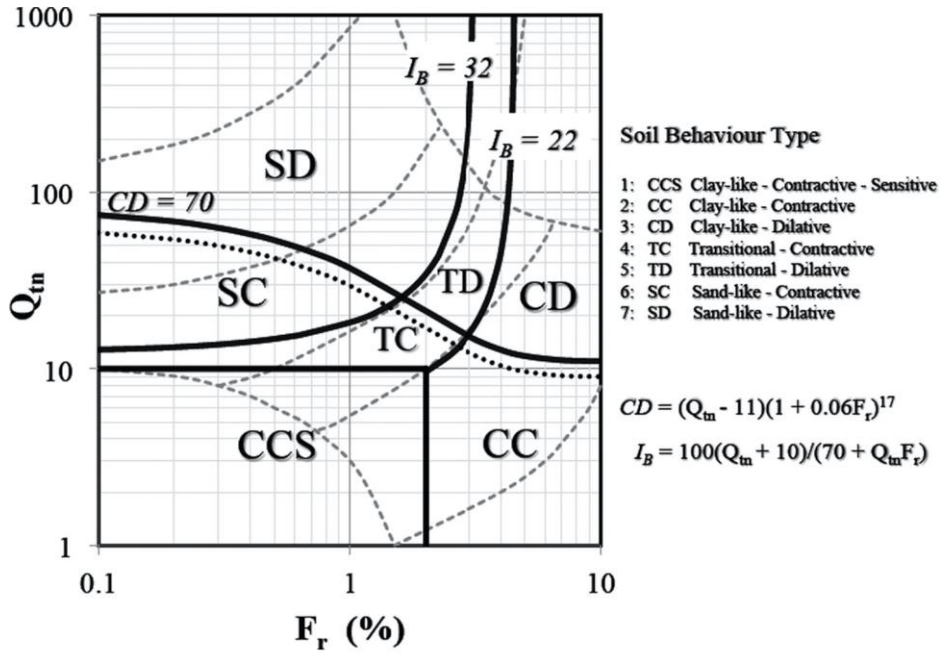

Figure 13: Modified soil behaviour type according to Robertson (2016)

A summary of the three different soil behaviour types and their respective inputs provided by Robertson is given in Table 5.

Table 5: Overview of soil behaviour types provided by Robertson

Label	Reference	Number of SBT	Parameters
SBT	Robertson, 1990	9	R_f, q_t
SBTn	Robertson, 2009	9	Q_t, B_q
Mod. SBTn	Robertson, 2016	7	Q_{tn}, F_r

4 Building a Machine Learning Model

The content provided in this chapter is based mainly on the book “Python Machine Learning” by Raschka & Mirjalili (2019) as well as on the websites of Python (Python Software Foundation, 2020), Pandas (The pandas development team, 2020), Matplotlib (The Matplotlib development team, 2020), Seaborn (Waskom, 2020) and Scikit-learn (scikit-learn developers, 2020) with the related article “Scikit-Learn: Machine Learning in Python” (Pedregosa, et al., 2011)

The process of building a machine learning model can be divided into 4 major steps. First, the preparation of the dataset which is called pre-processing. Second, the training of the dataset where the model is learning from a part of the data to find the best relation between input and desired output. Third, the validation of the dataset, where the performance of the model is evaluated and improved by tuning hyperparameters, and fourth, the testing of the model on unseen data.

4.1 Pre-processing

One of the most important steps in data science using machine learning is the pre-processing of the dataset, which means the quantity and quality of data is evaluated and adjusted to satisfy requirements of the proposed learning technique. Robust predictions are generally obtained when the following conditions are fulfilled.

- **Complete data:**

  Generally, machine learning algorithms are not able to deal with empty inputs, so called null values or NANs. There are two possibilities to deal with empty inputs, first, if the number of null values compared to the whole dataset is small, relevant rows can be deleted, second, if there is not much data available, null values can be filled with synthetic data. Mostly the average or median value of all other inputs of one feature is taken to fill these empty cells. There is no general rule, which procedure should be used as it depends largely on the quality and quantity of the available data and the projected learning approach.

- **Feature selection:**

  In this thesis, two types of feature selection methods are distinguished. First, feature selection considering geotechnical parameters, which means the features are selected with respect to their origin, e.g., solely measured test data or additional interpreted features. Second, features which are selected through machine learning techniques, e.g., sequential feature selection algorithms, to optimize the selection of features with respect to the output of the learning algorithm but these are not used in this thesis. With automatic

feature selection, a subset of features, most relevant to the problem is determined to improve computational efficiency or to reduce generalization error. Generalization errors are often related to overfitting (and underfitting), which generally describes the performance ratio of an algorithm on the test set compared to the training set. Overfitting, or high variance can be caused by having too many features, leading to a too complex model. Underfitting or high bias is often caused by applying a model which is not complex enough. The consequence of overfitting is that the model fits the training data better than the test data. The consequence of underfitting is that the model cannot learn from the data and therefore also not predict sufficiently from test data. A visualization of the aforementioned generalization errors is provided in Figure 14.

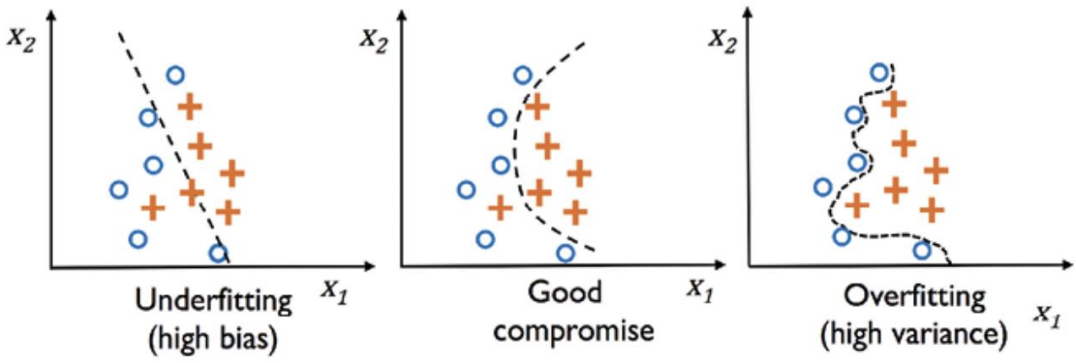

Figure 14: Visualisation of generalization errors (Raschka & Mirjalili, 2019)

- **Split of test and training data:**

  To evaluate the prediction performance of a learning model it has to be tested on unseen data. Since training and testing data is usually gained from the same dataset it must be split in a way that it has no influence on the statistical parameters of the data (e.g., mean and median) and that data is also selected randomly to avoid that parts of the dataset are under- or overrepresented in testing or training data. The most common algorithm for validation in scikit-learn is “train-test-split”. It separates training and test data randomly based on a user defined ratio, where usually the data for training is of about 60-90 % of the dataset and the rest is used for testing. After evaluation, the train and test data are often reunited, and the model is retrained on the entire dataset.

- **Balanced dataset**

  The balance of a dataset can be described as the ratio between the available target values. A balanced dataset consists of a nearly uniform amount of target values, whereas in an unbalanced dataset parts of the target values are dominating. There are two common ways to deal with unbalanced datasets. First, if there is much data available, balance between the targets can be

reached by deleting rows with overrepresented targets. Second, if there is only a small amount of data available, data for underrepresented targes is increased using synthetic data. A combination of both is also possible and it has to be ensured that the distribution of the data values remain the same after deleting data or adding synthetic data. The consequences of an unbalanced dataset can be described with the following example: Assuming a dataset with 1000 rows whereas 995 of them are related to target A and only 5 are related to target B. If this model gets trained and tested, it might result in an accuracy score for the prediction of 95% without predicting target B once. If now the goal of the model is to find target B it might be useless. An example for an unbalanced dataset is provided in Figure 15, class 3.0 is clearly underrepresented and if the goal of the model is to find class 3.0 in new unseen data it might result in very bad predictions.

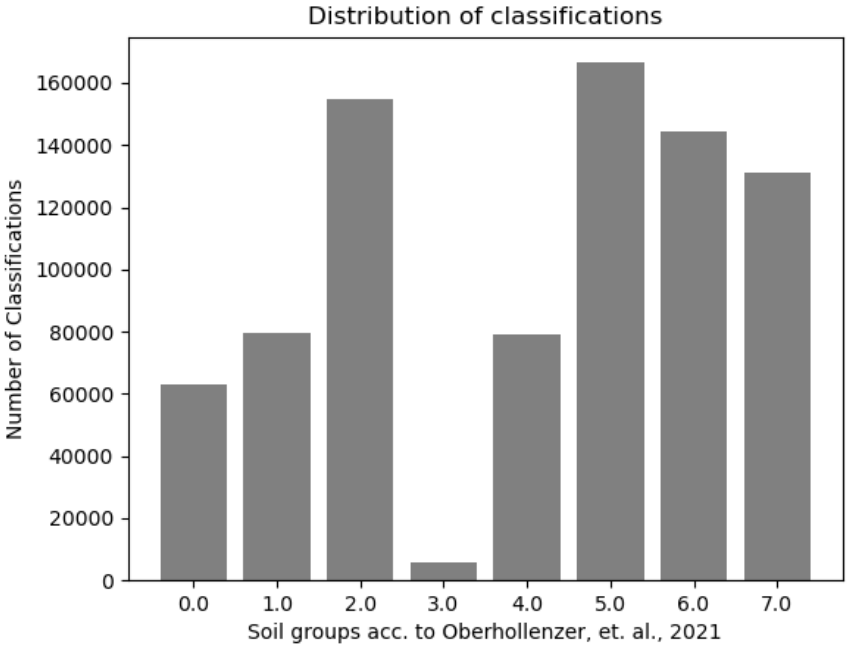

Figure 15: Example of an unbalanced dataset. Data taken from Oberhollenzer, et. al., (2021)

- **Scaling of features**

  Besides the balance of features in the dataset, the model quality of many algorithms, except decision trees and random forest, depends on the scale of each feature of the dataset too. For example, if values of features “A” and “B” are in a range between 1 and 10 and values of feature “C” are in a range between 1 and 1000, the prediction of the learning algorithm will most probably strongly depend on feature “C”, while features “A” and “B” will play a minor role due to their small size compared to “C”. Two common ways to bring features onto the same scale are standardization and normalization.

Normalization is often described as rescaling features to a range of 0 and 1, while in standardization the feature columns are centred at mean 0 with standard deviation 1 that the feature columns have equal parameters compared to a standard normal distribution, additionally, standardization is less sensitive against outliers and also keeps useful information about them. Table 6 provides a comparison between the aforementioned scaling techniques.

Table 6: Comparison of feature scaling techniques (Raschka & Mirjalili, 2019)

Input	Standardized	Min-max normalized
0,0	-1,46385	0,0
1,0	-0,87831	0,2
2,0	-0,29277	0,4
3,0	0,29277	0,6
4,0	0,87831	0,8
5,0	1,46385	1,0

The proposed procedure of pre-processing the data for this thesis is provided in Figure 16.

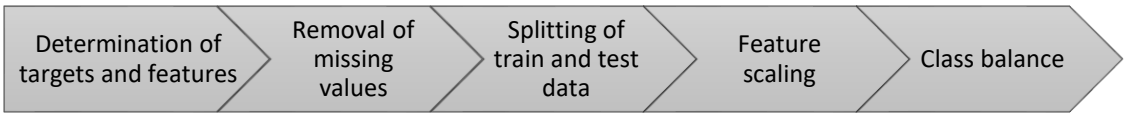

Figure 16: Pre-processing procedure

As a first step, the dataset is checked with respect to the types of entries (integers, strings, float, etc.) and to the number of null values. The missing entries can either be filled with synthetic data or deleted from the data set. After that the respective target for each model is determined (Note: In this thesis, features are selected only by considering geotechnical properties, therefore automatic feature selection algorithms, e.g., sequential feature selection, are not applied). The next steps are splitting the dataset into a training and testing dataset to measure the prediction performance of each model. After that, the features are scaled to ensure that each feature has the same influence on the training process. The last step of the pre-processing procedure in this thesis is the evaluation of the class balance of the dataset.

4.1.1 Evaluation of dataset

The first step of data pre-processing is the evaluation of the raw data. The dataset is checked for completeness and the type of data, e.g. integers or strings as well as missing values are identified. Table 7 provides an overview of the database “CPT_PremstallerGeotechnik”.

Except the columns “test_type”, “basin_valey” and “EN_ISO_14688_classes”, which will not be used for the model anyway, all columns are of type integer or float, therefore they don’t have to be converted for the machine learning algorithm. The column “NAN” displays the sum of the missing values for each feature column of the dataset. The processing of these values is described in chapter 4.1.3.

Table 7: Data types and null values (NAN) of database

column	Dtype	NAN
ID	int64	0
test_type	object	0
basin_valley	object	0
Depth (m)	float64	0
qc (MPa)	float64	0
fs (kPa)	float64	0
u2 (kPa)	float64	706354
Vs (m/s)	float64	1027077
qt (MPa)	float64	0
Rf (%)	float64	0
γ (kN/m³)	float64	0
σ,v (kPa)	float64	0
u0 (kPa)	float64	0
σ',v (kPa)	float64	0
Qt (-)	float64	235
Qtn (-)	float64	18
Fr (%)	float64	778
Bq (-)	float64	706355
U2 (-)	float64	706354
SBT (-)	float64	0
SBTn (-)	float64	0
Mod. SBTn (-)	float64	0
n	float64	0
Ic (-)	float64	0
Ic SBT (-)	float64	0
Ib (-)	float64	687
EN_ISO_14688_classes	object	149402
Oberhollenzer_classes	float64	0

4.1.2 Determination of targets and features

In this thesis, machine learning models based on an Artificial Neural Network, a Support Vector Machine or a Random Forest for four different targets (Oberhollenzer_classes, SBT, SBTn, Mod.SBTn) are built. For each target, two models, one where only raw measuring data from CPT tests and another one, where additional features, defined by an engineer, e.g. groundwater level and unit weight of soil, are used. This leads to two different approaches for each algorithm

and furthermore to overall 6 models for each target classification. Table 8 summarizes the targets with their respective models and features.

For each target, three algorithms with two versions of inputs, respectively are used. As mentioned above, one version considers only measured data from CPT tests and the other version considers additional interpreted or defined data (based on. groundwater level and unit weight of soil) as features. A sequential feature selection is not applied in this thesis.

To evaluate the ability of a model to detect soil strata and determine the respective soil types on unseen CPT data, two arbitrary CPTu tests are selected from the dataset before the data is used for training and validation. The selected test datasets are from piezocone tests to cover all possible input and output values of the learning models and have the IDs 934 and 1317.

Table 8: Targets with the respective models and features

Target	Model ID	Classifier	Features
Oberhollenzer_classes	OC1_SVM	SVM	Depth, qc, fs, Rf
	OC2_SVM	SVM	Depth, qc, fs, σv, u0, σ'v, Rf
	OC1_ANN	ANN	Depth, qc, fs, Rf
	OC2_ANN	ANN	Depth, qc, fs, σv, u0, σ'v, Rf
	OC1_RF	RF	Depth, qc, fs, Rf
	OC2_RF	RF	Depth, qc, fs, σv, u0, σ'v, Rf
SBT	SBT1_SVM	SVM	Depth, qc, fs, Rf
	SBT2_SVM	SVM	Depth, qc, fs, σv, u0, σ'v, Rf
	SBT1_ANN	ANN	Depth, qc, fs, Rf
	SBT2_ANN	ANN	Depth, qc, fs, σv, u0, σ'v, Rf
	SBT1_RF	RF	Depth, qc, fs, Rf
	SBT2_RF	RF	Depth, qc, fs, σv, u0, σ'v, Rf
SBTn	SBTn1_SVM	SVM	Depth, qc, fs, Rf
	SBTn2_SVM	SVM	Depth, qc, fs, σv, u0, σ'v, Rf
	SBTn1_ANN	ANN	Depth, qc, fs, Rf
	SBTn2_ANN	ANN	Depth, qc, fs, σv, u0, σ'v, Rf
	SBTn1_RF	RF	Depth, qc, fs, Rf
	SBTn2_RF	RF	Depth, qc, fs, σv, u0, σ'v, Rf
Mod.SBTn	MSBTn1_SVM	SVM	Depth, qc, fs, Rf
	MSBTn2_SVM	SVM	Depth, qc, fs, σv, u0, σ'v, Rf
	MSBTn1_ANN	ANN	Depth, qc, fs, Rf
	MSBTn2_ANN	ANN	Depth, qc, fs, σv, u0, σ'v, Rf
	MSBTn1_RF	RF	Depth, qc, fs, Rf
	MSBTn2_Rf	RF	Depth, qc, fs, σv, u0, σ'v, Rf
SVM…Support Vector Machine | ANN…Artificial Neural Network | RF…Random Forest			

4.1.3 Missing values

Most Machine Learning algorithms cannot handle missing values (often called null-values or NAN). Generally, there are two ways to deal with this circumstance, first is to fill empty cells with values based on the rest of the data (e.g. mean or median). Second is to delete entire rows with empty cells.
In this thesis, rows with empty cells are deleted and the data frame is updated gradually. The python code for this step is provided below:

```
df = df.dropna(subset = ['Oberhollenzer_classes'])
df = df.dropna(subset = ['Depth (m)'])
df = df.dropna(subset = ['qc (MPa)'])
df = df.dropna(subset = ['fs (kPa)'])
df = df.dropna(subset = ['σ,v (kPa)'])
df = df.dropna(subset = ['u0 (kPa)'])
df = df.dropna(subset = ["σ',v (kPa)"])
df = df.dropna(subset = ['Rf (%)']
```

Compared to the raw database (Table 7) the number of empty cells yields to zero for used features and targets. Table 9 provides the number of data available for training and validation of each model.

Table 9: Amount of data for training and testing

Target	Model ID	Amount of data
Oberhollenzer_classes	OC1_SVM	1025284
	OC2_SVM	1029283
	OC1_ANN	1025284
	OC2_ANN	1029283
	OC1_RF	1025284
	OC2_RF	1029283
SBT	SBT1_SVM	2514264
	SBT2_SVM	2514263
	SBT1_ANN	2514264
	SBT2_ANN	2514263
	SBT1_RF	2514264
	SBT2_RF	2514263
SBTn	SBTn1_SVM	2514262
	SBTn2_SVM	2514262
	SBTn1_ANN	2514262
	SBTn2_ANN	2514262
	SBTn1_RF	2514262
	SBTn2_RF	2514262
Mod.SBTn	MSBTn1_SVM	2514262
	MSBTn2_SVM	2514262
	MSBTn1_ANN	2514262
	MSBTn2_ANN	2514262
	MSBTn1_RF	2514262
	MSBTn2_Rf	2514262

4.1.4 Splitting of training and test data

To measure the performance of a learning model it is necessary to split the dataset into train and test data. The data used for training is usually about 70 – 90 percent of the dataset. The complement of the training size is used for testing.

Data	Dataset	
Train_test_split	Train	Test

Figure 17: Splitting the dataset into training and test data

In python, the split of the database into two subsets is performed with the `train_test_split` algorithm from the Scikit-learn model selection:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 42)
```

The size of the respective subset is defined with the arguments `test_size` or `train_size`. If no inputs are given, the train size is automatically set to 0.25 (= 25 %). The argument `random_state` is a parameter used to reproduce the split at a later state and is usually set to 0 or 42.

4.1.5 Feature scaling

Many machine learning algorithms, except Random Forests, are sensitive to the scale of their respective input features, which means if feature “A” is in a range between 1 and 10 and feature “B” ranges between 1 and 1000, feature “B” will most probably dominate the decision function of the model. To ensure equal influence of each input feature to the model, the features should be scaled.
Two of the most common ways of scaling features are the standardization (scaling from -1 to 1) and scaling features to a range (usually from 0 to 1). In this thesis, standardization is used for scaling the features. Therefore, `StandardScaler` from the scikit-learn library is implemented in the model code:

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

This module standardizes features by removing the mean and scaling to unit variance. The mean and standard deviation is stored to be later reapplied on a testing set. To visualize the influence of standardization a fraction of the input features before and after scaling is given in Table 10

Table 10: Cut-out of data before and after scaling with `StandardScaler()`

Data before scaling			
Depth (m)	q_c (MPa)	f_s (kPa)	R_f (%)
0,50	26,42	195,00	0,74
0,51	26,57	198,10	0,75
0,52	26,19	197,70	0,75
0,53	26,13	197,20	0,75
0,54	26,05	189,60	0,74
0,55	26,05	191,00	0,75
0,56	23,93	186,90	0,77
0,57	22,97	182,90	0,80
0,58	22,14	182,00	0,79
0,59	22,14	168,20	0,76
Data after scaling			
Depth (m)	q_c (MPa)	f_s (kPa)	R_f (%)
-1,566698904	0,888468556	0,691169776	-1,025978352
-1,218543592	0,973898225	1,040131714	-0,512989176
-0,870388280	0,757476397	0,995104367	-0,512989176
-0,522232968	0,723304529	0,938820184	-0,512989176
-0,174077656	0,683437351	0,083300592	-1,025978352
0,174077656	0,677742039	0,240896306	-0,512989176
0,522232968	-0,529663947	-0,220634000	0,512989176
0,870388280	-1,076413827	-0,670907469	2,051956704
1,218543592	-1,549124661	-0,772219000	1,538967528
1,566698904	-1,549124661	-2,325662469	0,000000000

4.1.6 Class balance

The class imbalance of a dataset can have significant influence on the ability of a model to predict underrepresented classes.

One popular way to deal with imbalanced classes during training, is to assign a larger penalty to wrong predictions on the minority class. This is implemented in various classifiers of scikit-learn, by setting the `class_weight` parameter to `class_weight='balanced'`.

Another technique to balance class proportions is to use sampling algorithms. By undersampling, the number of overrepresented classes is reduced to that of the less represented ones. By oversampling, the number of underrepresented classes is increased. A commonly used algorithm is *Synthetic Minority Oversampling Technique* - *SMOTE*. A usual practice is to combine these sampling methods. There are various sampling algorithms available. In this thesis *SMOTEENN* and *SMOTETomek* are used to evaluate the influence of class imbalance. These algorithms provide a combination of over- and undersampling techniques, where first new datapoints are generated for weaker classes (*SMOTE*) and after that noisy data is cleaned using *Tomek's link* (*Tomek*) or edited nearest-neighbours

(*ENN*).The influence of the sampling algorithms on the dataset are provided in Figure 19. (Lemaitre, Nogueira, Oliveira D., & Aridas , 2020)
Since the application and evaluation of synthetic sampling algorithms on the dataset is beyond the scope of this thesis, the algorithms are just used once to visualize the effect on the data.

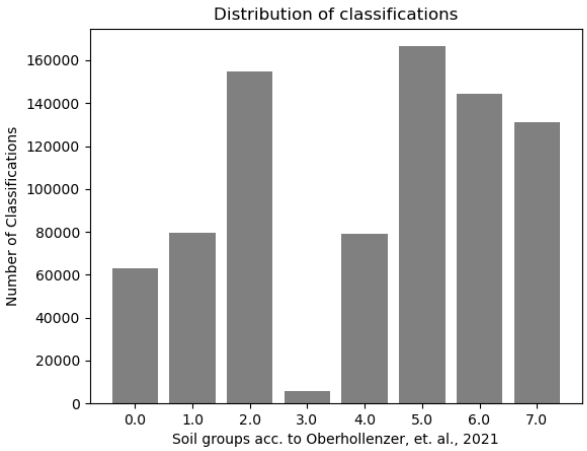

Figure 18: Distribution of classes in unsampled data set. Data taken from Oberhollenzer, et.al., (2021)

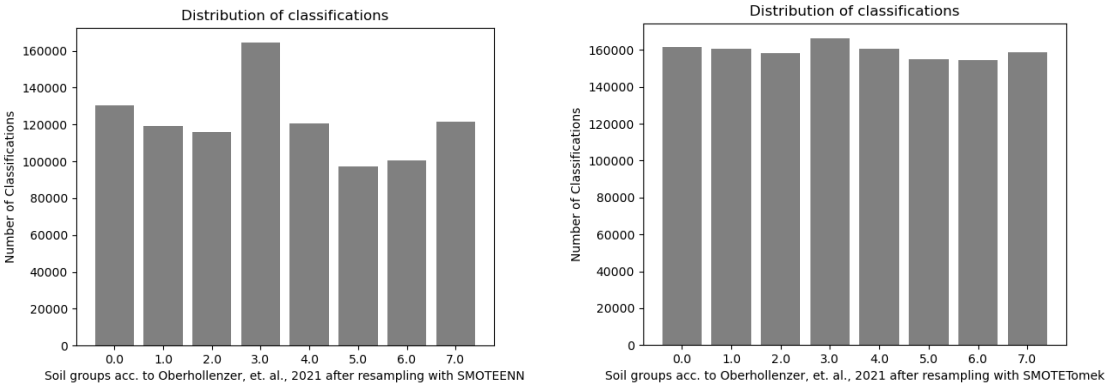

Figure 19: Distribution of classes after resampling with SMOTEENN (left) and SMOTETomek (right).

4.2 Training, validating and testing

4.2.1 MLP Classifier – ANN

Multi-layer Perceptron (MLP) is a supervised learning module of scikit-learn. It can learn a non-linear function estimator for classification and regression and basically consists of one input layer with features X_n, one output layer $f(X)$ and k hidden layers with neurons a_k in between.

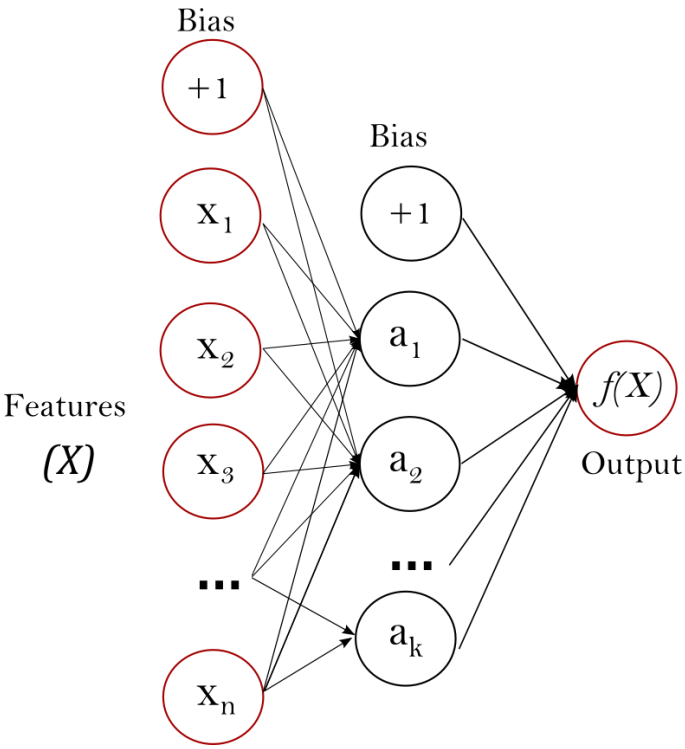

Figure 20: Multi-layer Perceptron with one hidden layer (scikit-learn developers, 2020)

At each neuron, the values from the previous layer get transformed with a weighted linear summation function $w_1x_1 + w_2x_2 + ... + w_kx_k$ and with a non-linear activation function. The output layer transforms the values of the last hidden layers into the target values. The advantages of a Multi-layer Perceptron are that it is able to learn non-linear models and also learn models in real-time, e.g., for online applications. The disadvantage is that it is sensitive to feature scaling, it requires determination of hyperparameters, and it has a non-convex loss function which means that there is more than one local minimum and therefore, different random weight initializations can lead to different validation scores.

The implementation of the class `MLPClassifier` in python is displayed below. With `clf.fit`, the model is trained with training samples `X_train` and associated target samples `y_train`. With `clf.predict`, the target values for the test samples `X_test` are predicted. These values are compared with the target values of the test samples `y_test` to compute the model performance and the classification error.

```
from sklearn.neural_network import MLPClassifier

clf = MLPClassifier()
clf.fit(X_train, y_train)
pred_clf = clf.predict(X_test)
```

The main hyperparameters which have to be defined and are evaluated in this thesis are provided on the next page.

- **hidden_layer_sizes**

  The parameter `hidden_layer_sizes` defines the number of hidden layers with their respective number of neurons. An example of how to determine these numbers is given below.

- **max_iter**

  The parameter `max_iter` defines the number of iterations. If the chosen number is too small, the learning process may not be finished, and the model will most probably underfit the data. If the number iterations is too big the model may overfit and not generalize very well.

All other parameters which can be adjusted are provided in the user guide of the scikit-learn website:

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Contrary to support vector machines and random forests, the MLPClassifier has no parameters to handle unbalanced data. Therefore, the class balance must be evaluated and if necessary improved before the model gets trained. Additionally, this classifier is sensitive to feature scaling, therefore, the input features must be scaled beforehand.

There are a few common rules of thumb to determine the number of hidden layers and neurons. In this thesis the recommendations of Heaton (2015) are applied. The influence of the number of hidden layers is provided below:

- **None**

  Only capable of representing linear separable functions and decisions

- **One**

  Can approximate any function that contains a continuous mapping from on finite space to another

- **Two**

  Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.

- **More than two**

  Additional layers can learn complex representations (sort of automatic feature engineering for layers).

Some rules of thumb for the number of hidden neurons:

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

4.2.2 Support Vector Classifier– SVM

A support vector machine is a supervised learning algorithm used for classification, regression and outlier detection. It is effective in high dimensional spaces and when the number of dimensions is greater than the number of samples, memory efficient and versatile, because kernel functions can be adjusted for the decision function. The disadvantages are that it does not provide probability estimates directly and training of large datasets can be time intensive as it is shown in this thesis.

The scikit-learn library provides multiple options for classification based on a support vector machine. The most common is the support vector classifier *SVC* (used in this thesis). Other modules are *LinearSVC* which works exclusively with a linear kernel and *SGDClassifier* which implements a SVM with stochastic gradient descent learning, but these are not applied in this thesis.

The most important parameter which has to be defined for the SVC is the kernel type. The kernel can be chosen between a linear, polynomial, sigmoid, precomputed and radial basis function which is also the default option. The influence of the different kernels on the separating hyperplanes in 2D space is displayed in Figure 21

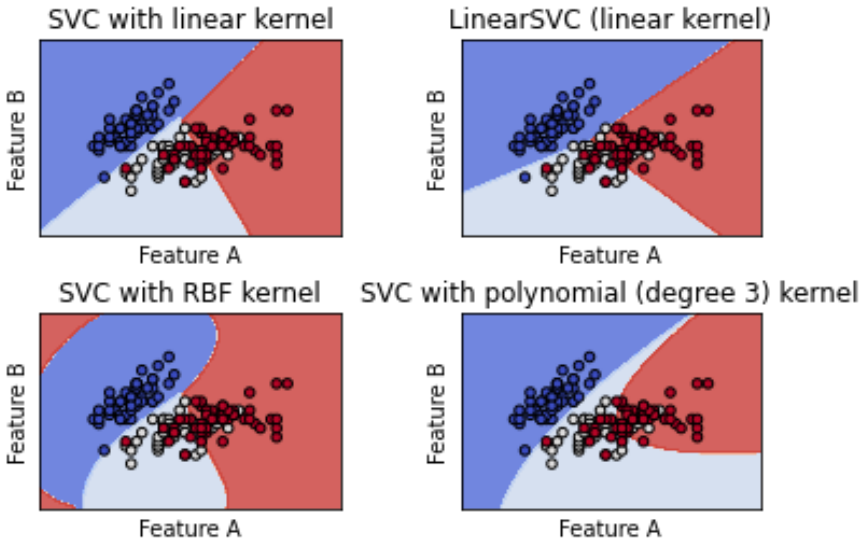

Figure 21: Different hyperplanes for different kernels of a SVC (scikit-learn developers, 2020)

A support vector machine is implemented in a python script as follows:

```
from sklearn import svm
from sklearn.svm import SVC

clf = svm.SVC()
clf.fit(X_train, y_train)
pred_clf = clf.predict(X_test)
```

The complete list of parameters is available on the website of scikit-learn:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

4.2.3 Random Forest Classifier – RF

The random forest classifier is a combination of decision tree classifiers. In scikit-learn, the random forest combines these classifiers by averaging their probabilistic prediction. Decision trees typically tend to overfit and have high variance. Due to decoupled prediction errors, and taking the average of those predictions, some errors cancel out. Therefore, random forests achieve a reduction in variance at the price of a slightly increased bias.

The main parameters which have to be focused for the hyperparameter tuning are the number of decision trees in the forest, called `n_estimators` , the size of random subsets of features to consider when splitting a node, called `max_features`, and the maximum size of each tree, called `max_depth`. The best combination of these parameters is evaluated in this thesis once for the Oberhollenzer_classes and once for all soil behaviour type classifications. More information is provided in chapter 7.

When adjusting the aforementioned parameters in a random forest model, the number of trees has usually no influence on susceptibility of a model to overfitting. In contrary, the depth of a tree is one main parameter to control the variance of the model. Figure 22 displays a decision tree with a `max_depth` of 2 and in Figure 23 a decision tree with `max_depth` of 3 is displayed (the parameter `max_depth` is usually set to a number greater than 15). Note: To display the trees on one A4 sheet, the input features for these decision trees are only the tip resistance q_c and the sleeve friction f_s and the targets are only class 5-7 from Oberhollenzer classes.

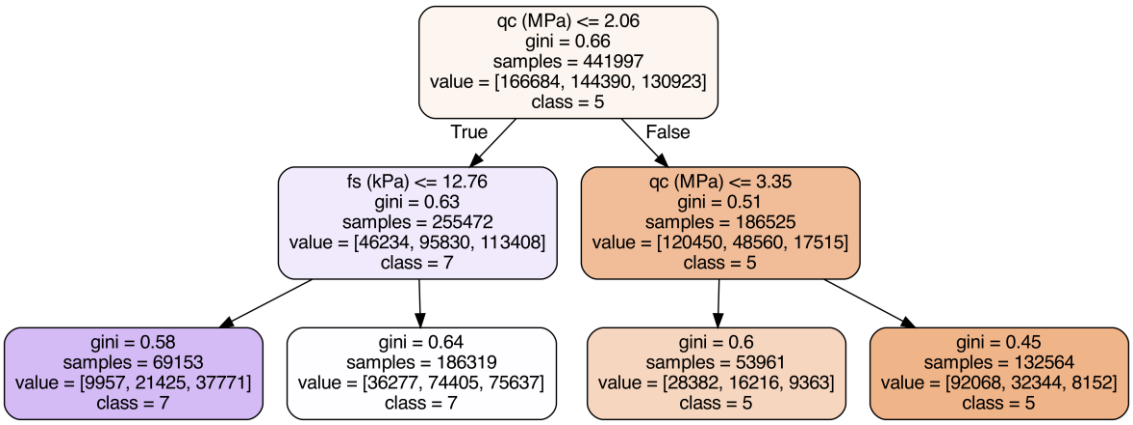

Figure 22: Decision tree with `max_depth` of 2

The first line of the node displays the decision function found by the algorithm. The decision function is defined for each node independently. The second line provides the decision criterion.

The default criterion for the random forest is the Gini impurity which represents the probability of a random datapoint to be classified wrong. The Gini Impurity indicates the quality of the split: A low impurity indicates a good split; a high impurity indicates a bad split. To calculate the impurity of a certain node, first the probability of each class in the node must be defined:

$$prob\ x\ = \frac{n_{samples}\ node}{n_{samples}\ class\ x} \tag{12}$$

After calculating the probability of each class in the node, the Gini Impurity is calculated as follows:

$$gini\ = 1 - (prob\ x1 * prob\ x1 + \cdots prob\ xi * prob\ xi) \tag{13}$$

As an example, the calculation for the Gini Impurity of the first node is provided in Table 11.

Table 11: Calculation of the Gini Impurity for one node

	n_samples	probability of class x	Gini Impurity
node	441997		
class 5	166684	0,377	0,663
class 6	144390	0,327	
class 7	130923	0,296	

The third line indicates the number of samples which are observed, and the fourth line indicates which samples fall in each of the categories (class). The last row displays the most common class for each node. If a sample satisfies the condition of the decision function it goes left, otherwise it goes right. The main factor which influences the complexity of the Random Forest model is the size of each decision tree. The greater the trees, the more complex the model, which also means that for complex data structures deeper trees must be chosen. In contrary, if the trees are too large, the model could tend to overfit the data. Hence, a good trade-off must be found through model validation. Note: in Figure 22 and 23, only split nodes and no leaf nodes are visualized, which means that the last row of nodes in these figures show only the final split and not the resulting class. The predicted classes of the final split are described in the line "values".

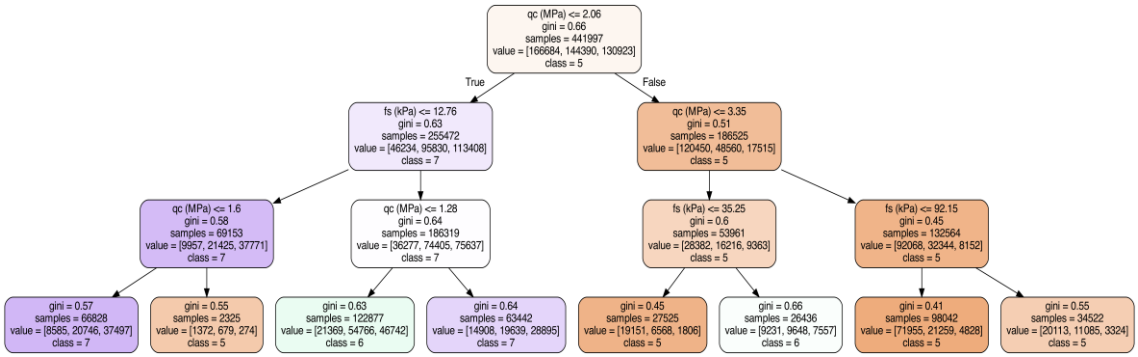

Figure 23: Decision tree with `max_depth` of 3

The random forest classifier is implemented in python as follows:

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators = 50,
max_depth=(15), max_features='auto')
clf.fit(X_train, y_train)
pred_rfc = clf.predict(X_test)
```

The complete list of parameters is available on the website of scikit-learn:

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#

4.2.4 Model evaluation and optimization

Before the developed learning model can be used on a test data set, the quality of the model has to be evaluated. This is done in the validation step. Besides the confusion matrix and the resulting scores of predictions (accuracy, precision, recall, f1-score), other characteristics like bias and variance are also important indicators for the model quality. In this thesis, the bias-variance trade-off is evaluated using learning and validation curves, the prediction performance is determined using the accuracy score in the validation step. Other scores like, precision, recall and f1-score (described in chapter 4.2.5) as well as the confusion matrix are only used for testing.

4.2.4.1 Validation

Before the model can be tested on the test dataset it must be validated. Therefore, the training set is split again into subsets for training and validation. The simplest form of validation is the *leave one out* method, where a predefined amount of data is used for training and the rest is used for validation. The disadvantage of this method is that the data is only split once. Important characteristics of the dataset may not be distributed sufficiently through train and validation set and therefore, errors due to over- and underfitting may be overseen. A severe improvement is reached using cross validation techniques. The data is split multiple times into subsets for training and validation. This leads to smoother results for the model performance. Additionally, bias and variance can be visualized with learning and validation curves.

A popular method for cross validation is *k-fold cross validation*. It basically splits the dataset k-times (user defined integer, a common value is between 5 and 10) into randomly defined test and training samples and performs the training and testing process k-times. Because cross validation is computationally intensive, the number of splits largely depends on the size of the dataset. In this thesis the number of splits is chosen between 3 and 5. Figure 24 displays the process of k-fold cross validation for k = 4. More information and other forms of cross validation techniques like *stratified k-fold cross validation* or *shuffle split* are provided on the website of scikit-learn user guide:

https://scikit-learn.org/stable/modules/cross_validation.html

Data	Dataset				
Train_test_split	Training data				Test data
1st fold	Validate	Train			Not used for validation
2nd fold	Train	Validate	Train		
3rd fold	Train		Validate	Train	
4th fold	Train			Validate	

Figure 24: K-fold cross validation with 4 splits

4.2.4.2 Bias-variance trade-off

The ability of a model to deliver accurate prediction results is basically influenced by two origins of errors. The error due to bias is the difference between the average prediction compared to the real values. The error due to variance is the difference between predictions for equal input features. Errors due to bias are the cause for underfitting models and errors due to variance for overfitting models. A visualisation of these errors is provided in Figure 25.

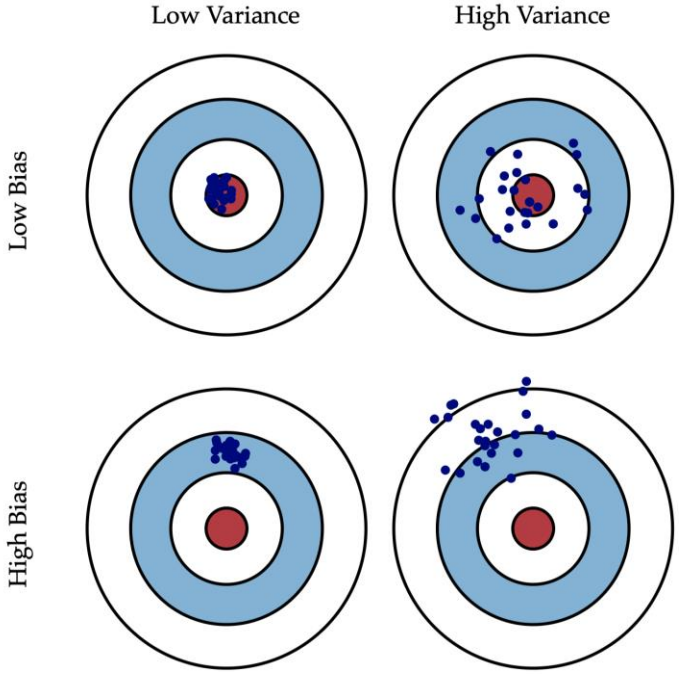

Figure 25: Bias and variance (Fortmann-Roe, 2012)

Adding more samples to a learning model, which also means raising the complexity leads usually to decreasing bias and increasing variance. This problem is called bias-variance trade-off. Figure 26 displays bias and variance contribution to the total error of a model. The optimum model quality is near the intersection point of bias and variance.

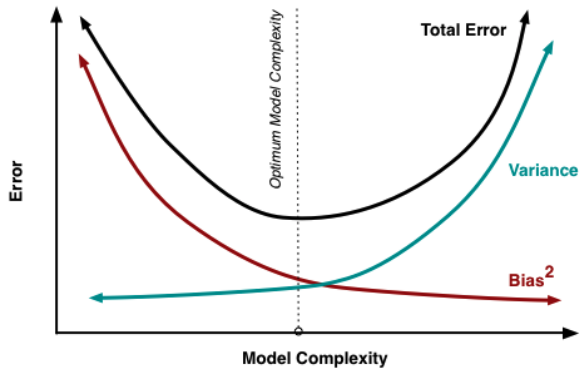

Figure 26: Bias and variance contribution to total error (Fortmann-Roe, 2012)

4.2.4.3 Learning and validation curves

The bias of a model is usually evaluated by plotting learning curves and variance by plotting validation curves. A learning curve is defined as the score (e.g., accuracy) or the error against the number of training data. A validation curve is defined as the score (e.g., accuracy) or error against a selected hyperparameter (e.g., number of trees in a random forest). The data for the curves is gained through the validation process of the model. The following code provides an example how first a learning curve and second a validation curve is generated for a MLP Classifier.

To get the necessary parameters for the learning curve, the module *learning curve* of scikit-learn is applied, whereof `estimator` defines the classifier, `X` and `y` define the training data, `train_sizes` defines the fractions of the dataset which are used, `cv` defines the applied cross validation module, and `n_jobs` defines the number of processor cores which are used for computing (-1 means all available cores are used)

```
from sklearn.model_selection import learning_curve

train_sizes, train_scores, test_scores =\
          learning_curve(estimator=MLPClassifier(hidden_layer
_sizes = (100,100,100), max_iter=600), X = X_train, y =
y_train, train_sizes = np.linspace(0.1,1,5), cv  = 3, n_jobs
= -1)
```

The parameters for the validation curve are generated, using the module *validation curve* of the scikit-learn library. For the validation curves, cross validations for various hyperparameter must be exercised. The desired hyperparameter is defined by `param_name`. The range of the defined parameter is defined by `param_range`. Fixed hyperparameter (e.g., maximum of iterations) must be defined directly in the `estimator`. All other classes are defined as above.

```
from sklearn.model_selection import validation_curve

train_scores, test_scores = validation_curve(estimator =
MLPClassifier(max_iter=500), X = X_train, y = y_train,
param_name = 'hidden_layer_sizes',
param_range = [(10), (10,10), (10,10,10), (10,10,10,10)],
cv = 5)
```

Figure 27 provides the three basic conditions of learning curves. On the upper left side, a model with high bias (underfitting) is displayed, contrary to that a model with high variance (overfitting) is displayed on the upper right side. On the lower left side, a desired trade-off for a model is given. Training accuracy describes the accuracy of the model fitting only the training data, and validation accuracy is the accuracy of the model fitting only the validation data.

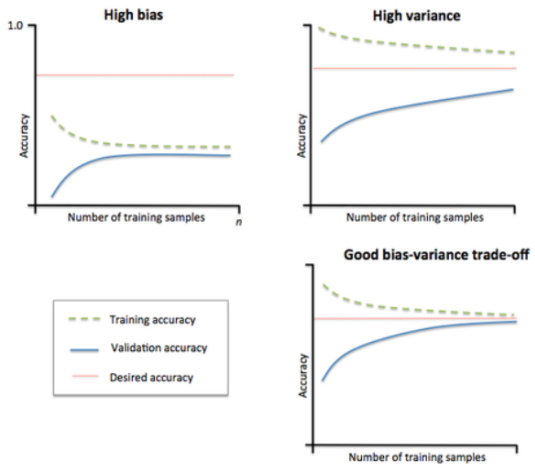

Figure 27: Explanation of learning curves (Raschka, 2020)

4.2.5 Model testing

After the model is trained and the optimum hyperparameters are evaluated it is tested on unseen data to evaluate the model performance. As mentioned before a split of the data at the ratio of 80 % for training to 20 % for testing is chosen for this thesis. The data for testing is randomly selected by the module `train_test_split` of scikit-learn. The performance of the model is visualized in a confusion matrix. In this matrix, four different types of right or wrong classifications are distinguished:

- **True positive (TP)**

  The model predicts positive and the actual state is positive.

- **False Negative (TP)**

  The model predicts negative, but the actual state is positive.

- **False positive (FP)**

  The model predicts positive, but the actual state is negative.

- **True negative (TN)**

  The model predicts negative and the actual state is negative

The summary of these four types of right or wrong classifications in one matrix is shown in Figure 28.

		actual	
		positive	negative
predicted	positive	True Positive (TP)	False Positive (FP)
	negative	False Negative (FN)	True Negative (TN)

Figure 28: Confusion matrix

The results of the confusion matrix for all classifications of the test set are summarized in the classification report. This report consists of four different classification scores, namely the accuracy, precision, f1-score and recall. The accuracy describes what percent of the predictions were correct and is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

The precision is defined as what percent of positive predictions were correct:

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

The recall describes what percent of positive cases did the model catch and is also a strong indicator if the model is sensitive to class balance in unbalanced datasets

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

The f1-score is defined as a ration between precision and recall and is often used as indicator for the model performance:

$$f1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{15}$$

The ability of each algorithm to learn and predict from the dataset is evaluated in the following chapters.

5 Support Vector Machine

The initial plan to build 8 learning models had to be discarded due to bad training performance of the models with respect to the time and quality. The training process for the model *OC1_SVM* lasted about 12 hours and the resulting model performance was not sufficient. To estimate the training time for a soil behaviour type model (SBT, SBTn or ModSBTn), where the dataset has more than double the size of the dataset for *Oberhollenzer_classes*, the model *SBT2_SVM* was trained vicariously. After 25 hours, the training process was cancelled manually. Due to the fact that the training time for models based on other algorithms (ANN, RF) is usually within a few minutes. Additionally, the performance of ANN and RF models is better than the performance of the SVM model, hence it has been decided that the models for the *Oberhollenzer_classes* are just trained and tested, and not validated. The SVM models targeting the different Soil Behaviour Types are not further evaluated due to the vast amount of necessary training times. For the support vector machine, only *OC1_SVM* and *OC2_SVM* are evaluated.

5.1 Oberhollenzer_classes

The classification of the soil samples via grain size distribution according to Oberhollenzer, et. al., (2021) is based on adjacent bore hole samples. The classification was done with laboratory tests and the resulting classes were added to the CPT database. Hence, there is no empirical or statistical correlation between the measured values of the CPT tests (tip resistance q_c, sleeve friction f_s, etc.) and the resulting classes.

5.1.1 OC1_SVM

The first model build with a support vector classifier is called *OC1_SVM*. The model uses the depth, tip resistance, sleeve friction and friction ratio as input features. The 7 soil classes defined by Oberhollenzer, et. al., (2021) are used as targets. The only hyperparameter which is adjusted is the kernel function. The default kernel in scikit-learn uses a radial basis function which requires more computational effort than a linear one, hence it is decided to set the kernel function to ‘linear’. All other parameters are set to default.

Table 12: Parameter of OC1_SVM

Model information		
Target	Features	Hyperparameter
Oberhollenzer_classes	Depth, q_c, f_s, R_f	kernel = 'linear'

As mentioned before, the steps of validation and evaluation of learning curves are omitted due to the vast amount of training time compared to the resulting scores. After a training time of about 10 hours, the performance of the model was evaluated on the test data. The model yielded an accuracy of about 45 %. Table 13 provides the entire classification report. The scores are distinguished in macro average:

$$macro\ avg = \frac{\sum score\ of\ each\ class}{number\ of\ classes} \tag{16}$$

which provides the average score for all classes and weighted average:

$$weighted\ avg = \frac{\sum(score\ of\ each\ class * n_{classifications\ in\ class})}{n_{all\ classifications}} \tag{16}$$

which provides the average score for all classes weighted with the number of classifications.

Table 13: Classification report for OC1_SVM

OC1_SVM	**Classification report**	
	macro avg	weighted avg
accuracy	0,38	
precision	0,35	0,41
recall	0,29	0,38
f1-score	0,28	0,35

The confusion matrix is provided in Table 14. Green cells indicate right classifications, and all other cells are wrong classifications. Similarly to the classification report, the confusion matrix shows very plainly that the model is not able to classify the data correctly. Another thing that should be mentioned is that class 3 is not predicted once which indicates a high sensitivity to the class balance. Since the model's ability to learn from the data is worse than a best guess approach (50% right), not any other improvement steps are done. The influence of the class balance is considered by the next model (OC2_SVM).

Table 14: Confusion matrix for OC1_SVM

Confusion matrix									
OC1_SVM		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	1749	2085	505	0	129	3210	7459	657
	1	818	8049	684	0	606	7250	2397	138
	2	585	1111	9798	0	271	5578	11657	9471
	3	42	59	281	0	6	274	750	0
	4	825	3271	1411	0	999	10568	2420	48
	5	547	1384	3708	0	188	22923	11838	1161
	6	446	476	643	0	71	8584	18083	7860
	7	527	117	334	0	13	1684	13473	16636

5.1.2 OC2_SVM

The second model which uses a support vector machine for classification is called OC2_SVM. Additionally to the model OC1_SVM, the vertical total and effective stresses and the hydrostatic pore pressures caused by the groundwater table are used as input features. The kernel is again set to linear.

Table 15: Parameter of OC2_SVM

Model information		
Target	Features	Hyperparameter
Oberhollenzer_classes	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	kernel = 'linear'

The classification report shows that even with more input features the prediction accuracy does not increase. Only the precision and f1-score increase slightly by one percent. It is assumed that the model complexity is obviously too low for the data.

Table 16: Classification report for OC2_SVM

OC2_SVM	Classification report	
	macro avg	weighted avg
accuracy	0.38	
precision	0,36	0,42
recall	0,30	0,38
f1-score	0,29	0,36

Similar to the first SVM model, class 3 was not covered in the prediction, because of unbalanced data. The training time was about 11.5 hours.

Confusion matrix									
OC2_SVM		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	1161	2132	489	0	94	2819	8403	696
	1	527	8049	649	0	590	7082	2920	125
	2	264	1028	9663	0	339	5482	11996	9699
	3	28	72	264	0	0	264	781	3
	4	517	3649	1378	0	539	10697	2715	47
	5	295	1353	3550	0	145	22417	12838	1151
	6	196	437	560	0	112	8421	18742	7695
	7	229	124	349	0	9	1645	13529	16899

5.1.3 Influence of class balance

To evaluate the influence of class balance in the dataset, the model OC2_SVM is trained a second time. The inputs stay the same, except the parameter `class_weight` is set to `'balanced'`. This leads to a larger penalty assigned to wrong predictions of the minority class, and therefore, the imbalance between classes gets considered. The inputs are provided in Table 17.

Table 17: Parameter of OC2_SVM 'balanced'

Model information		
Target	Features	Hyperparameter
Oberhollenzer_classes	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	kernel = 'linear'

The resulting scores of this model are given in Table 18. The scores of the classification report clearly indicate that the model quality is decreasing with the `class_weight` set to `'balanced'`. The prediction accuracy of this model is only 31%.

Table 18: Classification report for OC2_SVM 'balanced'

OC2_SVM	Classification report	
	macro avg	weighted avg
accuracy	0,31	
precision	0,35	0,42
recall	0,37	0,31
f1-score	0,29	0,33

The confusion matrix shows an improvement of the model in predicting underrepresented classes like class 3, but overall, the prediction performance got worse. The training time of this model also increased compared to the unbalanced one to about 14.5 hours.

Table 19: Confusion matrix for OC2_SVM 'balanced'

Confusion matrix									
OC2_SVM		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	3898	1987	113	6824	234	465	1532	741
	1	4043	7797	346	4149	2066	1056	334	151
	2	1303	1176	8613	6961	1871	2782	4808	10957
	3	59	56	6	1235	14	5	37	0
	4	3744	3083	1397	4236	3455	3045	490	92
	5	7396	1338	3661	6916	2503	10475	7598	1862
	6	4584	507	343	6025	962	3535	9700	10507
	7	2275	118	138	5030	117	360	6211	18535

5.1.4 Discussion

The prediction performance of the evaluated SVM models is consistently below 50% which clearly indicates underfitting. This means that the model is not complex enough to learn and predict from the data. Another drawback of the SVM models is the vast amount of time that it takes to train them. Considering the combination of bad prediction performances with long training times, it can be assumed that the support vector classifier is not a preferable tool to classify soil types from CPT data. Also, the change of the decision function from linear to a radial basis function or polynomial function does not improve the result significantly.

5.2 Soil Behaviour Types – SBT

The training of the first SVM-model is terminated after about 26 hours of elapsed time. The training time of other models like Random Forest and ANN is comparatively much lower and within a few minutes. Since all models for soil behaviour type classification (SBT, SBTn, ModSBTn) are very similar, it is decided that all of these models get discarded. Information about input values for the executed model is provided in Table 20.

Table 20: Model information for SBT1_SVM

Model information		
Target	Features	Hyperparameter
Soil Behaviour type - SBT	Depth, q_c, f_s, R_f	kernel = 'linear'

6 Artificial Neural Network

For the classification with neural networks, 8 different models are built and evaluated. The influence of different hyperparameters on the model performance is assessed using grid search techniques, in which the model is trained and validated using different combinations of predefined hyperparameters. This is done once, vicariously for the Oberhollenzer_classes and for the soil behaviour type classifications, respectively.

All models are evaluated using learning and validation curves, before they are tested on the test dataset.

6.1 Grid Search for Oberhollenzer_classes

To identify relevant hyperparameter for the neural network model, a grid search model is built. In this model, a range of different values or settings for hyperparameters are defined. The model finds the best combination of this parameter via cross validation. Since the influence of the number of hidden layers and neurons on the model is obvious, they are set constant at one hidden layer with 10 neurons for the first grid search. The number of maximum iterations is also set constant at 500. The relevant parameters which are identified are provided in Table 21.

Table 21: Parameter for Grid Search for Oberhollenzer_classes

Grid Search for OC	
Parameter	Defined range
number of hidden layers	1
number of neurons in each layer	10
max_iter	500
activation function	'identity', 'logistic', 'tanh', 'relu'
learning_rate	'constant', 'adaptive'
solver	'sgd', 'adam'

The best combination of parameters is provided in Table 23. The best settings found for the learning rate and solver were actually default ones, additionally the best working setting for the activation function (`'tanh'`) is not essentially better than the default one (`'relu'`), therefore, it is decided to build all models with the default parameters except, the number of hidden layers, neurons and maximum iterations. A description of the hyperparameter according to the scikit-learn explanation is given in Table 22.

Table 22: Meaning of hyperparameter according to the scikit-learn website (scikit-learn developers, 2020).

activation function	identity	no-op activation, useful to implement linear bottleneck, returns f(x) = x
	logistic	the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x))
	tanh	the hyperbolic tan function, returns f(x) = tanh(x)
	relu	the rectified linear unit function, returns f(x) = max(0, x)
learning rate	constant	is a constant learning rate given by 'learning_rate_init'.
	adaptive	keeps the learning rate constant to 'learning_rate_init' as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least tol, or fail to increase validation score by at least tol if 'early_stopping' is on, the current learning rate is divided by 5.
solver	sgd	refers to stochastic gradient descent
	adam	refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba

Table 23: Best parameters found

Best parameters of Grid Search for OC	
Parameter	Best parameter
number of hidden layers	1
number of neurons in each layer	10
activation function	'tanh'
learning_rate	'constant'
solver	'adam'

The influence of the number of hidden layers and neurons is evaluated in another grid search, first, the number of hidden layers is increased, and in a second model the number of neurons is increased for one layer. Note: An increased number of layers and neurons is strongly affecting the training time of a model, therefore, the influence of an increased number of neurons and layers is evaluated in separate ways. The number of layers is evaluated with 10 neurons in each. The results are provided in Table 24.

Table 24: Evaluation of the number of hidden layers

Grid Search for hidden layers		
Number of hidden layers	Cros_val_score	relative improvement in %
1	0,435	--
2	0,459	5,5
3	0,461	0,4
4	0,468	1,0

The results show that the cross-validation score (accuracy) basically increases with the addition of hidden layers, but not by much. Increasing the number from one to four, improves the performance by only about 7.5%. Based on this result it is decided to build all models targeting Oberhollenzer_classes with maximum 3 hidden layers.

The influence of the number of neurons is evaluated with one hidden layer and shown in Table 25.

Table 25: Evaluation of the number of neurons

Grid Search for number of neurons		
Number of neurons	Cros_val_score	relative improvement in %
10	0,435	--
50	0,467	7,4
100	0,479	2,5
150	0,491	2,5

Again, with an increased number of neurons, the cross-validation score improves slightly. The improvement of the model performance is about 13% when increasing the number of neurons from 10 to 150. The training time is increasing from 2 minutes and 14 seconds to 13 minutes and 46 seconds, which is roughly about 600%. Since the neural network built in this thesis is not a *deep neural network* with a high number of hidden layers and neurons, the number of hidden layers and neurons are selected according to the rule of thumbs provided in chapter 4.2.1 and evaluated with validation and learning curves.

6.2 Oberhollenzer_classes

6.2.1 OC1_ANN

The first model using a neural network classifier is built with only measured data as input features and the 7 Oberhollenzer_classes as targets.

Table 26: Evaluation of the number of neurons

Model information		
Target	Features	Hyperparameter
Oberhollenzer_classes	Depth, q_c, f_s, σ_v	hidden_layer_sizes = (10,10,10)

The model performance is evaluated using the modules `learning_curve` and `validation_curve` of the scikit-learn library. The learning curve for the first model is created using 5-fold cross validation and is displayed in Figure 29. The training accuracy indicates how the model fits to the training data and the validation accuracy indicates how the model fits to the data used for validation. The difference, or distance between those curves is called variance (overfitting).

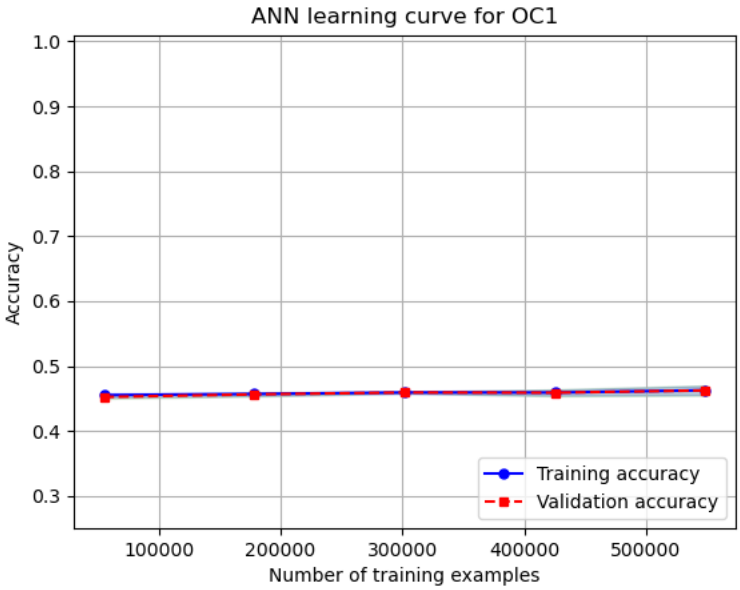

Figure 29: ANN learning curve for OC1

The training and validation accuracy are nearly on the same line through the whole training process which indicates that the model is strongly underfitting and the data is too complex. The necessary time to generate the learning curve is about 16 minutes. The influence of the number of hidden layers is plotted in Figure 30.

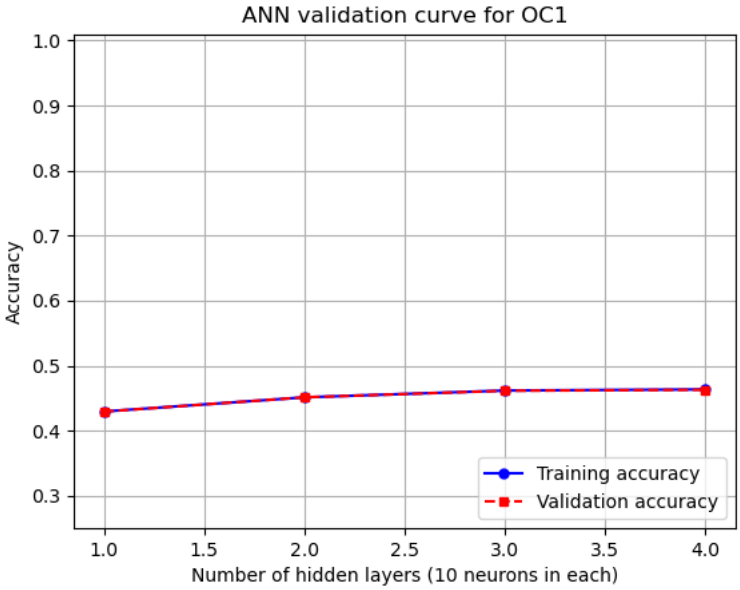

Figure 30: ANN validation curve for OC1 (values from Table 24)

Also, in the validation curve, the graph of the training and validation accuracy are throughout on the same level. To evaluate the robustness of this model and its setting of hyperparameters, the learning curve is plotted again, but now with 3 hidden layers and 100 neurons in each. The number of folds for the cross validation

is decreased from 5 to 3 (which means it is only trained and validated 3 times) but the elapsed time still increased to about 12 hours.

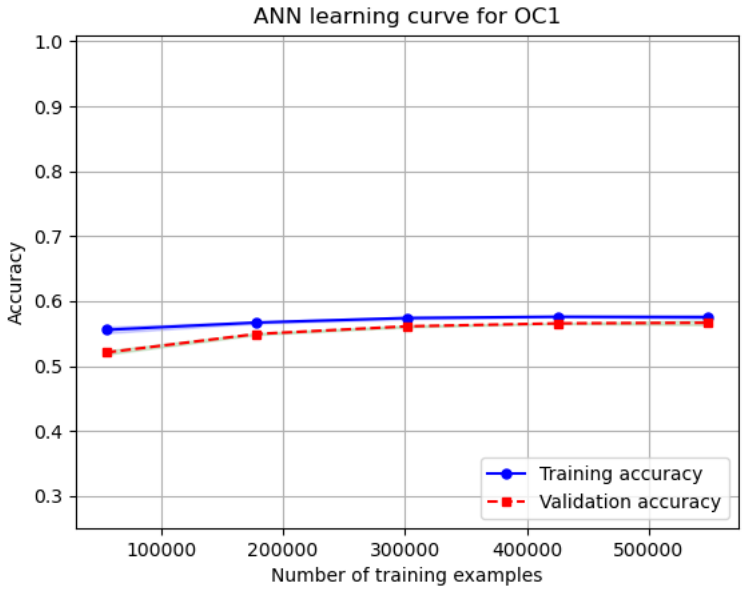

Figure 31: ANN learning curve for OC1 with 3 hidden layers and 100 neurons in each.

The validation score increases to about 56-57% but it is still insufficient for soil classification. The elapsed time increases from a few minutes to several hours although the cross-validation folds are decreased. Also, the learning curve gets nearly horizontal and therefore it is assumed that the score will not exceed a maximum of 60% no matter how many hidden layers and neurons are introduced. Considering the long validation time of about 12 hours in combination with the small gain in validation score, it is assumed that the neural network model with 3 hidden layers and 10 neurons is the best trade-off between model performance and effort of training.

The prediction performance is evaluated with the test dataset. The classification report is given in Table 27. The test accuracy of the model is similar to the validation accuracy at 46% (Figure 29).

Table 27: Classification report for OC1_ANN

OC1_ANN	**Classification report**	
	macro avg	weighted avg
accuracy	0,46	
precision	0,44	0,46
recall	0,39	0,46
f1-score	0,40	0,45

The confusion matrix is displayed below. Compared to the support vector classifier, the MLP classifier is able to identify and predict the underrepresented class 3 although, the dataset is highly unbalanced.

Table 28: Confusion matrix for OC1_ANN

Confusion matrix									
OC1_ANN		Predicted							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Actual	**0**	5333	2401	1691	27	832	2177	2232	1101
	1	1766	8025	1498	12	3418	4327	582	314
	2	842	620	17109	43	1979	5559	4407	7912
	3	95	118	652	93	74	180	81	119
	4	1387	3819	1952	9	5072	6305	706	292
	5	944	1595	3511	6	2914	25045	5233	2501
	6	793	571	3444	21	898	9572	13397	7467
	7	534	278	2422	39	381	2653	6016	20461

6.2.2 OC2_ANN

The second neural network model for the classification of Oberhollenzer_classes introduces the effective and total vertical stresses and the hydrostatic porewater pressures as additional input features.

Table 29: Parameter for OC2_ANN

Model information		
Target	Features	Hyperparameter
Oberhollenzer_classes	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	hidden_layer_sizes = (10,10,10)

First, the learning and validation curves are plotted to gain information about bias and variance of the model. Both, the learning (Figure 32) and validation curve (Figure 33) show a high bias and no variance of the model which indicates that the model is strongly underfitting. The learning curve is generated using 3 hidden layers with 10 neurons, respectively.

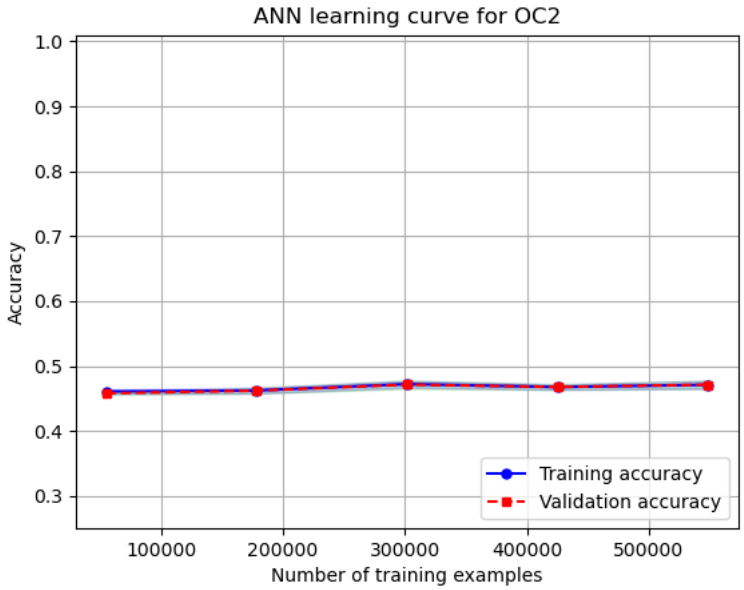

Figure 32: ANN learning curve for OC2

The validation curve (Figure 33) is plotted using one, two, three and four hidden layers with 10 neurons in each layer. Even with 3 more input features, the model performance is not significantly increasing.

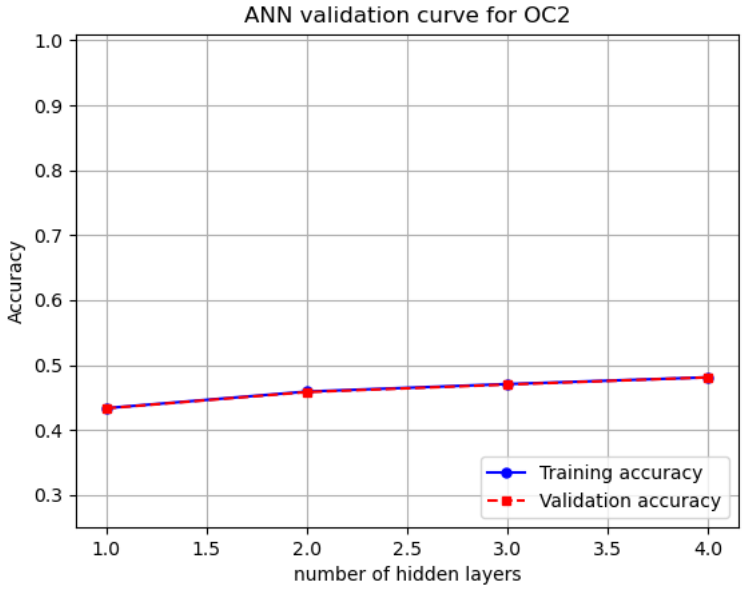

Figure 33: ANN validation curve for OC2

The learning and validation curves show that the model is not complex enough to find and learn patterns in the data. An increased number of hidden layers and neurons will also increase the model performance slightly but at the cost of a highly increased training time. As stated before, in this thesis a good trade-off between model performance and training time is preferably searched.

In the next step the model is tested on the test dataset to generate the classification report and confusion matrix. The classification report shows that the prediction performance of the model is not essentially better than the model *OC1_ANN*.

Table 30: Classification report for OC2_ANN

OC2_ANN	**Classification report**	
	macro avg	weighted avg
accuracy	0,47	
precision	0,43	0,47
recall	0,40	0,47
f1-score	0,40	0,47

The overall bad performance is also visible in the confusion matrix, but again, compared to the SVM, the ANN is able to identify the underrepresented class 3. Since the MLP-Classifier of scikit-learn doesn't have an option for the class balance like the support vector classifier or random forest classifier, the class balance has to be improved manually. The elapsed time of training is about 12.5 minutes.

Table 31: Confusion matrix for OC2_ANN

Confusion matrix									
OC2_ANN		Predicted							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Actual	**0**	5982	2968	1532	20	389	2135	1962	806
	1	2017	10293	1011	28	1512	4270	488	323
	2	924	1108	16576	18	1702	6046	4697	7400
	3	123	139	707	44	23	185	27	164
	4	1656	5161	1447	62	4505	5830	560	321
	5	1122	2925	2888	44	1702	25393	5326	2349
	6	1381	1053	3001	18	509	8324	15314	6563
	7	1039	523	2359	46	146	2340	6949	19382

To assess if a better class balance of the dataset improves the model performance, the classes are balanced using the sampling algorithm of the imbalanced learn library *"SMOTETomek"*. Further information on under- and oversampling algorithms is provided on the website of imbalanced learn:

https://imbalanced-learn.readthedocs.io/en/stable/api.html

The implementation of the resampling module `SMOTETomek` in the learning model is done after splitting the data into train and test data:

```
from imblearn.combine import SMOTETomek

sm = SMOTETomek(random_state=42)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train)
```

After resampling the dataset, the distribution of the classifications changes as follows (before sampling on the left side and after sampling right):

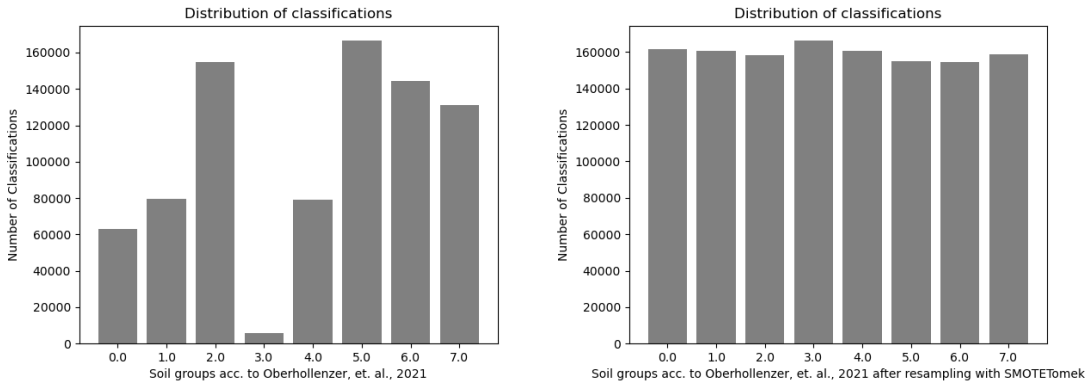

Figure 34: Distribution of classifications before (left) and after (right) resampling with `SMOTETomek`

The classification report shows that the scores are all decreasing compared to the model trained on unbalanced data. The reason therefore is part of ongoing research.

Table 32: Classification report for OC2_ANN 'balanced'

OC2_ANN 'balanced'	**Classification report**	
	macro avg	weighted avg
accuracy	0,42	
precision	0,40	0,46
recall	0,49	0,42
f1-score	0,39	0,42

The confusion matrix indicates an improvement in predicting underrepresented classes but also here the overall decrease of the model performance is visible. The training time increased by 2 minutes to ca. 14.5 minutes.

Table 33: Confusion matrix for OC2_ANN 'balanced'

Confusion matrix									
OC2_ANN 'balanced'		Predicted							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Actual	**0**	8863	2761	453	1001	450	928	415	923
	1	2997	10577	471	851	2957	1524	246	319
	2	3195	1655	10802	3250	3197	3574	3005	9793
	3	92	7	7	1256	13	1	1	35
	4	2231	5433	480	1095	6475	2914	622	292
	5	2966	4673	3295	1690	4252	16470	4967	3436
	6	3821	1884	1502	1585	1212	5687	11501	8971
	7	2340	669	856	1724	363	871	5574	20387

6.2.3 Discussion

A simple neural network model is not able to learn and predict the soil classes according to Oberhollenzer (2021) from the CPT test data sufficiently. The prediction accuracy is consistently below 50%. The performance of the models could basically be improved by adding hidden layers and neurons but only at the cost of a highly increased training time. Additionally, the class balance must be evaluated and optimized beforehand but the application of a resampling algorithm should be done carefully because the structure of the data could be changed completely.

6.3 Grid Search for Soil Behaviour Types

Since the targets of all soil behaviour type models are very similar, the influence of hyperparameters except the number of hidden layers and neurons is evaluated once using the model *SBT2_ANN* representative for all other behaviour type models (SBT, SBTn, ModSBTn). Similar to the *OC1_ANN* models, the learning rate, activation function and solver are set as variable parameters. The number of hidden layers and neurons is set constant at one layer with 10 neurons. The number of maximum iterations is set to 500.

Table 34: Range of parameters for Grid Search

Grid Search for Soil Behaviour Types	
Parameter	Defined range
number of hidden layers	1
number of neurons in each layer	10
max_iter	500
activation function	'identity', 'logistic', 'tanh', 'relu'
learning_rate	'constant', 'adaptive'
solver	'sgd', 'adam'

The best parameters found are again the same as the first grid search yielded. Except the activation function all favourable parameters are also the default ones. Since the difference between the activation function `'tanh'` to the default `'relu'` is very small, the default value is used for all parameters. The results are provided in Table 35.

Table 35: Results of hyperparameter tuning for SBT

Best parameters found for Soil Behaviour Types	
Parameter	Defined range
number of hidden layers	1
number of neurons in each layer	10
max_iter	500
activation function	'tanh'
learning_rate	'constant'
solver	'adam'

The influence of the number of hidden layers and number of neurons is again checked in two different steps. First, the number of hidden layers is increased from one to four with a constant number of 10 neurons in each. Table 36 shows that the best result is reached with two hidden layers. The performance of the network with a number of hidden layers beyond two is decreasing slightly.

Table 36: Grid search for number of hidden layers

Grid Search for hidden layers		
Number of hidden layers	Cros_val_score	relative improvement in %
1	0,977	--
2	0,985	0,8
3	0,983	-0,2
4	0,981	-0,2

The influence of the number of neurons is evaluated using one hidden layer with 10, 50, 100 and 150 neurons. Table 37 provides the results of the grid search model.

The cross-validation score increases slightly to 0.985% when the number of neurons is increased from 10 to 50. After 50 neurons the score reaches a peak of 0.985%. Therefore, it is assumed that the model is not able to predict with a higher accuracy.

Table 37: Grid search for number of neurons in hidden layer

Grid Search for number of neurons		
Number of neurons	Cros_val_score	relative improvement in %
10	0,977	--
50	0,985	0,8
100	0,985	0,0
150	0,985	0,0

The evaluation of hyperparameter using grid search techniques yielded the best set of parameters for the soil behaviour type models. The chosen values for all models are given in Table 38

Table 38: Resulting model parameters of grid search.

Best parameters found for Soil Behaviour Types	
Parameter	Defined range
number of hidden layers	3
number of neurons in each layer	10
max_iter	600

6.4 Learning and validation curves for Soil Behaviour Types

Since the targets of all Soil Behaviour Type models are highly similar, the learning and validation curves are only generated and visualized once, vicariously for all. The chosen model for the chart is *SBT2_ANN*. The number of hidden layers is set as evaluated in the grid search model to 3 layers with 10 neurons and the maximum iterations are set to 600.

The learning curve is provided in Figure 35. The model shows little to no variance (training and validation accuracy are nearly on the same level) and generally fits the data very good. The cross-validation score of the model is consistently above 95%.

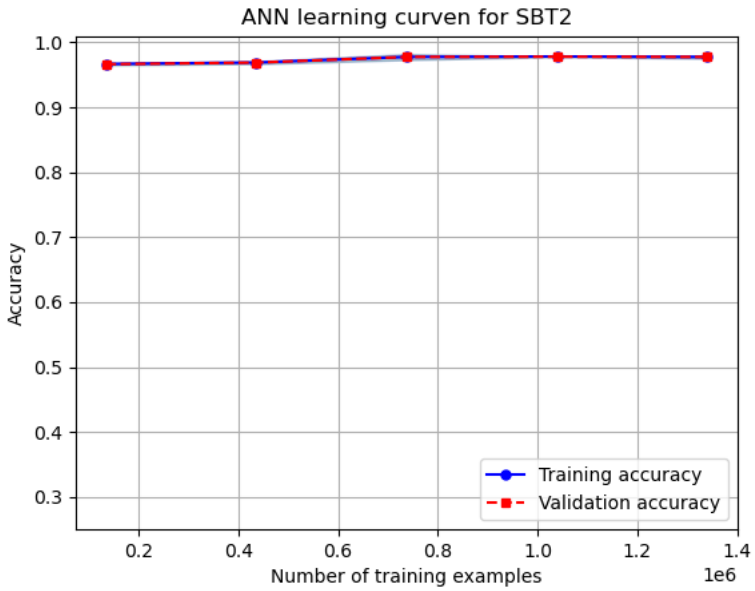

Figure 35: Learning curve for Soil Behaviour Type models

The validation curve is displayed in Figure 36. The training and validation curve are throughout on the same line and at a high level. This indicates a good model with little to no underfitting or overfitting.

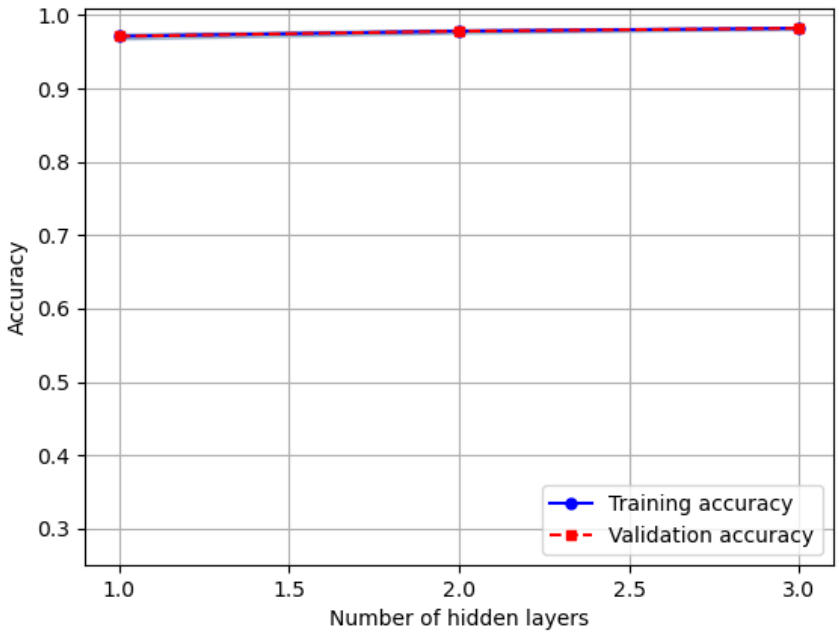

Figure 36: ANN validation curve for the Soil Behaviour Type Models

The validation step for Soil Behaviour Type models is now finished. In the next step, all models are tested on the test data and their respective classification report and confusion matrix is plotted.

6.5 Soil Behaviour Type models

In this chapter the results of the test procedure for all Soil Behaviour Type models are evaluated and discussed. Table 39 provides an overview of the models and their input features.

Table 39: Soil Behaviour Type models using an ANN

Target	Model ID	ML algorithm	Features
SBT	SBT1_ANN	ANN	Depth, q_c, f_s, R_f
	SBT2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f
SBTn	SBTn1_ANN	ANN	Depth, q_c, f_s, R_f
	SBTn2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f
Mod.SBTn	MSBTn1_ANN	ANN	Depth, q_c, f_s, R_f
	MSBTn2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f

The results as well as the learning and prediction characteristics of all neural network models for Soil Behaviour Type classification are quite similar, hence all classification reports and confusion matrices are provided in the following section without any respective comments. The results for all models are compared and discussed at the end of this chapter.

6.5.1 SBT1_ANN

Table 40: Confusion matrix for SBT1_ANN

Confusion matrix											
SBT1_ANN		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	10816	585	282	0	0	5	97	27	0	0
	1	290	6473	0	2	5	0	2	0	0	0
	2	5	13	16178	171	0	0	0	0	0	0
	3	0	100	381	111685	1435	0	0	0	0	4
	4	0	346	0	548	100645	935	0	0	0	5
	5	77	100	0	0	687	120078	929	0	1	3
	6	238	0	0	0	1	233	113665	49	0	0
	7	106	0	0	0	0	0	617	12620	0	0
	8	0	0	0	0	2	153	95	0	946	22
	9	0	0	0	2	44	1	0	0	0	1149

Table 41: Classification report for SBT1_ANN

SBT1_ANN	Classification report	
	macro avg	weighted avg
accuracy	0,98	
precision	0,97	0,98
recall	0,95	0,98
f1-score	0,96	0,98

6.5.2 SBT2_ANN

Table 42: Classification report for SBT2_ANN

SBT2_ANN	Classification report	
	macro avg	weighted avg
accuracy	0,98	
precision	0,95	0,98
recall	0,97	0,98
f1-score	0,96	0,98

Table 43: Confusion matrix for SBT2_ANN

Confusion matrix											
SBT2_ANN		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	11030	462	123	0	0	14	101	69	0	0
	1	447	6432	0	0	3	17	0	0	0	0
	2	161	4	16083	23	0	0	0	0	0	0
	3	0	171	666	113124	208	0	0	0	0	13
	4	0	266	0	1642	100390	226	0	0	0	55
	5	57	73	0	0	1338	119815	267	0	71	0
	6	213	0	0	0	0	950	112393	176	46	0
	7	76	0	0	0	0	0	212	13010	0	0
	8	0	0	0	0	0	8	2	1	1156	42
	9	0	0	0	2	1	1	0	0	0	1213

6.5.3 SBTn1_ANN

Table 44: Confusion matrix for SBTn1_ANN

Confusion matrix											
SBTn1_ANN		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	15507	42	205	4	0	70	594	330	2	0
	1	309	5341	1	288	45	10	2	1	0	0
	2	244	34	24893	2638	0	0	0	0	0	3
	3	13	143	1894	141899	2189	6	0	0	0	432
	4	2	60	1	3189	53608	3619	0	0	69	297
	5	18	17	0	1	880	104721	4182	2	259	2
	6	27	0	0	0	0	1958	103595	743	75	0
	7	210	0	0	0	0	0	910	17601	0	0
	8	0	0	0	0	8	155	148	0	3559	44
	9	1	0	0	168	132	1	0	0	139	5313

Table 45: Classification report for SBTn1_ANN

SBTn1_ANN	Classification report	
	macro avg	weighted avg
accuracy	0,95	
precision	0,93	0,95
recall	0,93	0,95
f1-score	0,93	0,95

6.5.4 SBTn2_ANN

Table 46: Confusion matrix for SBTn2_ANN

Confusion matrix										
SBTn1_ANN		Predicted								
		0	1	2	3	4	5	6	7	8	9
Actual	0	16080	75	234	50	0	23	75	211	6	0
	1	343	5056	12	545	35	3	1	2	0	0
	2	172	37	24208	3390	0	0	0	0	0	5
	3	33	23	1001	144588	628	6	0	0	0	297
	4	25	30	0	1541	57893	1080	3	0	31	242
	5	72	0	0	4	354	108050	1260	0	336	6
	6	496	0	0	0	15	599	103747	1461	80	0
	7	446	0	0	0	0	0	94	18181	0	0
	8	0	0	0	0	0	8	2	1	1156	42
	9	0	0	0	2	1	1	0	0	0	1213

Table 47: Classification report for SBTn2_ANN

SBTn1_ANN	Classification report	
	macro avg	weighted avg
accuracy	0,97	
precision	0,94	0,97
recall	0,95	0,97
f1-score	0,94	0,97

6.5.5 ModSBTn1_ANN

Table 48: Confusion matrix for ModSBTn1_ANN 'balanced'

Confusion matrix									
ModSBTn1_ANN		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	17058	543	168	4	208	89	367	470
	1	507	36775	1335	7	541	0	14	4
	2	33	2671	114714	3003	659	475	0	0
	3	6	4	694	33209	1	750	0	0
	4	21	575	540	0	26831	1163	1578	480
	5	5	0	26	568	696	30369	2	1036
	6	103	5	0	0	433	0	60637	4397
	7	256	0	0	2	41	946	2522	155312

Table 49: Classification report for ModSBTn1_ANN

ModSBTn1_ANN	Classification report	
	macro avg	weighted avg
accuracy	0,94	
precision	0,93	0,94
recall	0,93	0,94
f1-score	0,93	0,94

6.5.6 ModSBTn2_ANN

Table 50: Confusion matrix for ModSBTn1_ANN 'balanced'

Confusion matrix									
ModSBTn2_ANN		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	17619	149	90	0	168	86	360	435
	1	1042	36199	1524	0	409	0	3	6
	2	149	1327	118959	725	356	38	0	1
	3	45	0	191	33532	0	896	0	0
	4	52	90	193	1	30080	108	645	19
	5	5	0	2	111	139	31629	7	809
	6	258	1	0	0	110	0	64418	788
	7	287	2	0	0	19	175	476	158120

Table 51: Classification report for ModSBTn2_ANN

ModSBTn2_ANN	Classification report	
	macro avg	weighted avg
accuracy	0,98	
precision	0,96	0,98
recall	0,96	0,98
f1-score	0,96	0,98

6.5.7 Summary of results

All models predict consistent results with an accuracy between 94-98%. Except the *SBT_ANN* models, the performance of all other models increases by adding σ_v, u_0 and σ'_v to the feature set.

The models for SBT predict both on a high level regardless the number of input features. The reason for that is that the SBT-classes have a direct empirical relationship to the tip resistance q_c and sleeve friction f_s.

The models for SBTn and Mod.SBTn predict better when more input features, especially those considering the vertical stresses and groundwater table are added. The SBTn- and Mod.SBTn-classes have a 'normalized' empirical relationship to the measured data. Normalized means that the values are adjusted with respect to the influence of the groundwater table.

Table 52: Results of SBT models

Target	Model ID	Algorithm	Features	Accuracy
SBT	SBT1_ANN	ANN	Depth, q_c, f_s, R_f	0,98
	SBT2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,98
SBTn	SBTn1_ANN	ANN	Depth, q_c, f_s, R_f	0,95
	SBTn2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,97
Mod.SBTn	MSBTn1_ANN	ANN	Depth, q_c, f_s, R_f	0,94
	MSBTn2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,98

6.6 Discussion

The ANN models for Soil Behaviour Type classification predict the correct classes throughout with a high accuracy. In contrary, the models built for Oberhollenzer_classes are not able to sufficiently predict the right classes. The main reason for that is that compared to SBT, this soil classes have no empirical correlations to the CPT test data.
In order to optimize the neural network to the best possible prediction performance (for Oberhollenzer_classes), building a deep neural network should be considered, but since this requires also an upgrade in hardware, it is not part of this thesis.

7 Random Forest

Similar to the Neural Network models, 8 different models are evaluated using a Random Forest algorithm. The relevant hyperparameters for Random Forests are the number of trees `n_estimators`, and the maximum size of each tree `max_depth`. The influence of these parameters and the optimum combination is evaluated with learning and validation curves. Again, the learning and validation curves for all Soil Behaviour Type models are only plotted once, vicariously for all.

7.1 Oberhollenzer_classes

7.1.1 OC1_RF

The input parameters for generating the first learning curves are set to 100 for the number of trees and no limit is set for the maximum size of each tree. For the cross-validation, the folds are set to 5. The first model yields the best cross-validation score so far. However, the model has also the highest variance (overfitting the data) which is visible in the gap between the training and validation accuracy displayed in Figure 37 .

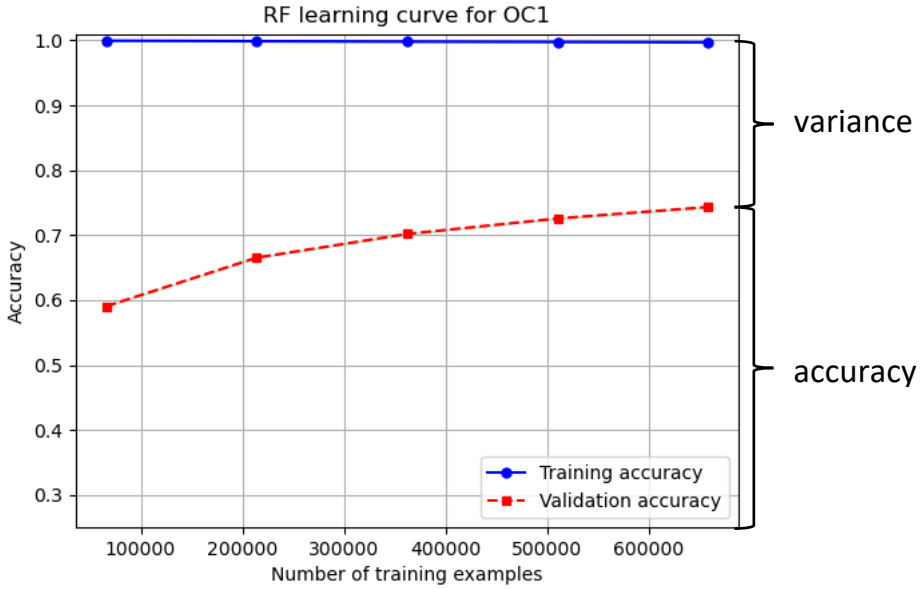

Figure 37: First RF Learning curve for OC1

To evaluate the influence of the size of each tree on the bias and variance of the model, the validation curves are generated by setting a range for the parameter `max_depth` to 10, 15, 20, and 25. The cross-validation is again set to 5 folds. The influence of the tree size can be seen clearly in Figure 38. The gap between training and validation accuracy is strongly increasing with the size of each tree.

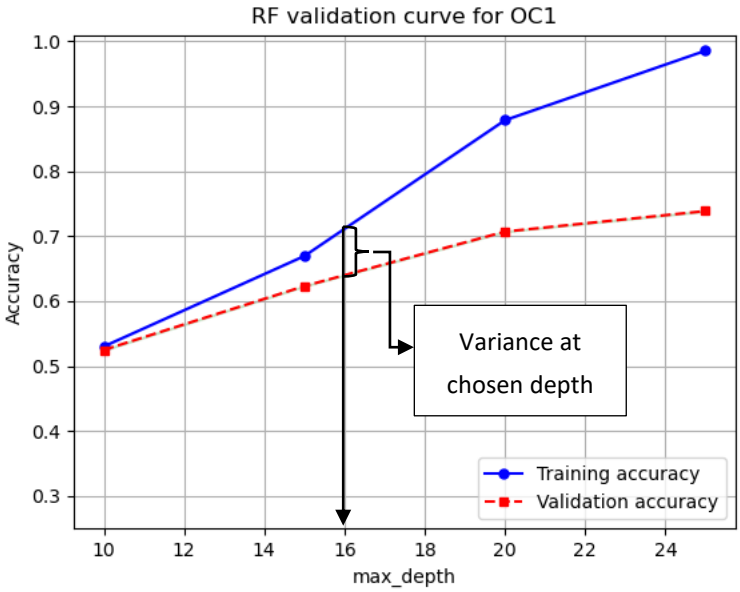

Figure 38: RF validation curve for OC1

In order to get a better bias variance trade-off, the maximum size of each tree is set to 16. The learning curve is generated again and visualized in Figure 39. The variance decreased considerably, but also the cross-validation score decreased by about 10 %. A further reduction of the variance would also reduce the model accuracy. Therefore, this is assumed to be the best trade-off for this model.

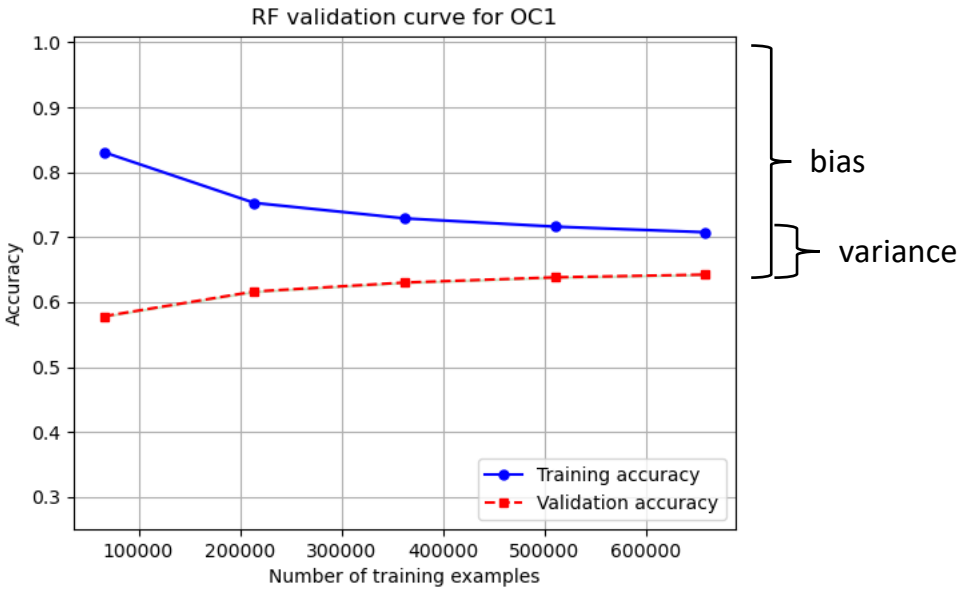

Figure 39: Second validation curve for OC1

The number of trees in a Random Forest does not influence the susceptibility of the model to overfitting. To proof this, a validation curve is generated using variating number of trees. The range for `n_estimators` is set to 10, 25, 50 and 100. The validation curve in Figure 40 indicates that the bias and also the variance stays

constant after a specific number of trees. Additionally, it can be assumed that the number of estimators set to 50 is be enough for this model and more trees do not increase the model quality.

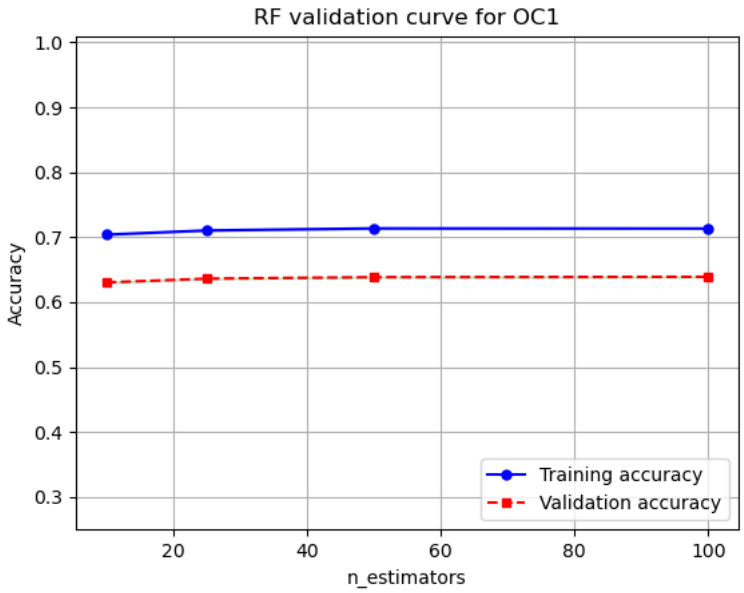

Figure 40: RF validation curve with varying number of estimators

In the next step, the model performance gets evaluated on the test dataset to generate the confusion matrix and the classification report. The training time is about 3 minutes and the model reached an accuracy of 65%. Both, accuracy and training time are the best until now. All other scores are provided in Table 53.

Table 53: Classification report for OC1_RF

OC1_RF	**Classification report**	
	macro avg	weighted avg
accuracy	0,65	
precision	0,64	0,65
recall	0,57	0,65
f1-score	0,59	0,64

The confusion matrix is given in Table 54. Compared to the SVM and ANN models, the RF performs quite well with respect to the class balance. Underrepresented classes like class 3 are also caught by the algorithm.

Table 54: Confusion matrix for OC1_RF

Confusion matrix									
OC1_RF		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	9116	1685	1209	25	702	1573	820	664
	1	1304	12001	644	9	1977	2890	750	367
	2	578	621	26878	31	1215	4000	3192	1956
	3	75	108	457	284	81	217	103	87
	4	1037	2958	1079	7	8985	4333	746	397
	5	937	1784	1787	10	1744	29756	4156	1575
	6	995	680	2641	42	572	6540	21784	2909
	7	534	345	1554	27	299	2143	3886	23996

7.1.2 OC2_RF

The second model for this soil classes is again trained using additional input features. Before training the model, the optimum set of hyperparameters is again searched by plotting the learning and validation curves. The validation curve (Figure 41) again indicates a growing variance when the size of the trees in the forest is increased. It is decided that the best bias-variance trade-off is at a `max_depth` of 18.

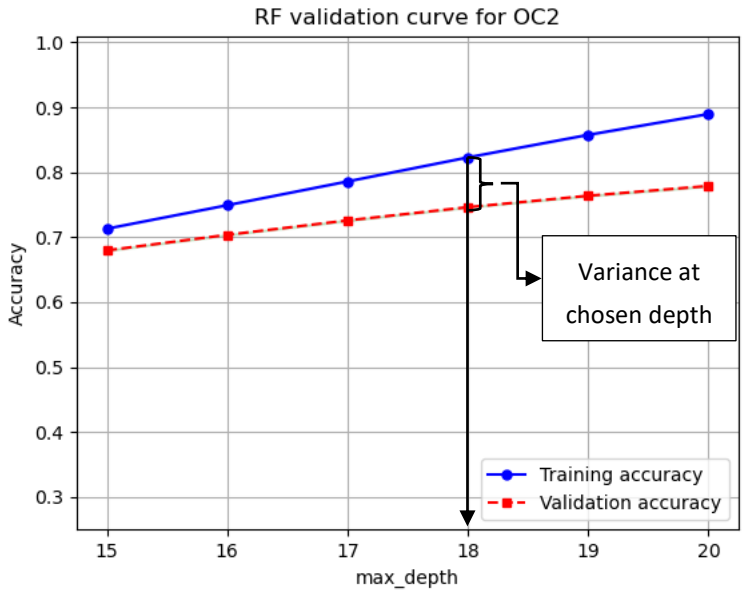

Figure 41: RF validation curve for OC2

The learning curve for the desired parameter is provided in Figure 42. In order to retain a sufficient accuracy, the variance is not further decreased. The influence of the maximum tree size is displayed in Figure 43.

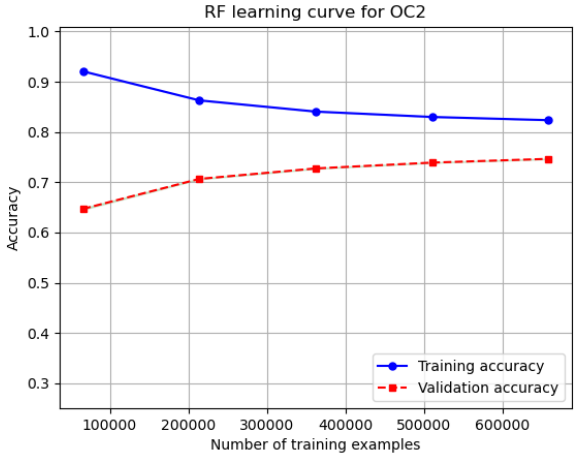

Figure 42: RF learning curve for OC2 (`max_depth` = 18)

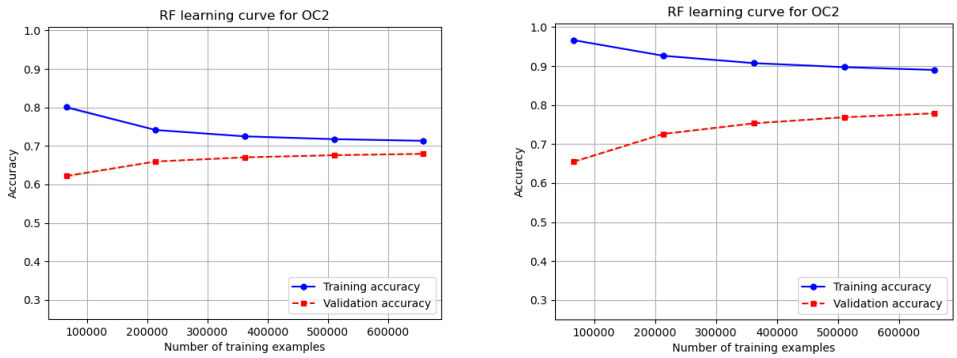

Figure 43: RF learning curve for OC2 with 15 (left) and 20 (right) for `max_depth`

The model is tested on the test dataset. Compared to other models, the reached accuracy of 75% is best for the classification of Oberhollenzer_classes. The confusion matrix is provided in Table 55 and indicates a solid performance of the model both in predicting underrepresented classes as well as the overall prediction performance.

Table 55: Confusion matrix for OC2_RF

Confusion matrix									
OC2_RF		Predicted							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Actual	**0**	10789	1358	716	12	588	1322	588	421
	1	1084	13968	505	19	1240	2236	609	281
	2	474	409	30722	38	729	2630	2066	1403
	3	51	39	413	689	32	68	55	65
	4	870	2015	800	14	12104	2908	534	297
	5	719	1304	1121	13	985	33760	2805	1042
	6	800	581	1596	50	526	4713	26080	1817
	7	509	285	993	43	220	1433	2579	26722

The classification report is given in Table 56. The model yielded the best results for all evaluated scores in predicting Oberhollenzer_classes.

Table 56: Classification report for OC2_RF

OC2_RF	Classification report	
	macro avg	weighted avg
accuracy	0,75	
precision	0,75	0,76
recall	0,70	0,75
f1-score	0,72	0,75

7.2 Soil Behaviour Types

The procedure for the determination of the model parameters for all Soil Behaviour Type models is similar to the ANN models. The best parameter set is evaluated with one model, vicariously for all. Therefore, the learning curves and validation curves are generated using the model *SBT2_RF* and *ModSBTn1_RF*.

To assess the bias-variance behaviour of the models, the learning curves of the aforementioned models are generated and visualized. The number of trees (100) and the maximum size of the trees (unrestricted) is set to default. Figure 44 provides the learning curve for the model *SBT2_RF*. As visible in the curves, the model has a very high accuracy and nearly zero variance, therefore the default hyperparameters are obviously appropriate.

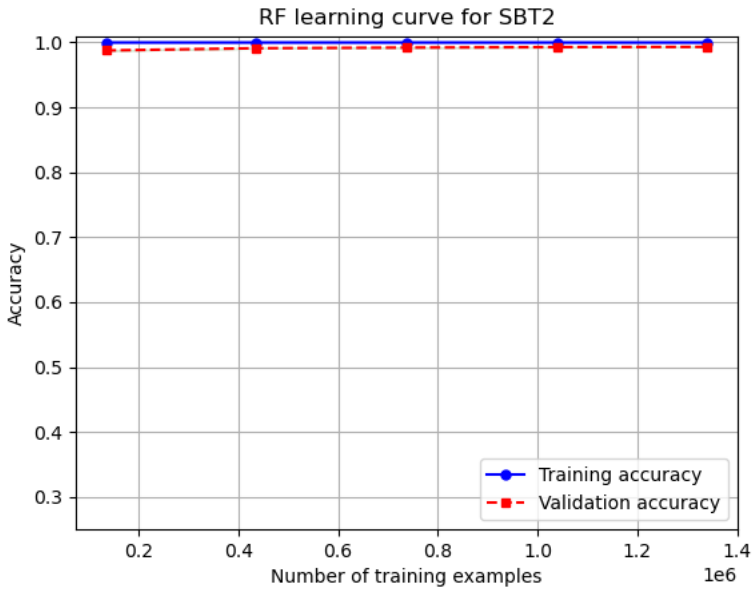

Figure 44: RF learning curve for SBT2

The learning curve for *ModSBTn1_RF* and *ModSBTn2_RF* is provided in Figure 45. As expected, the validation accuracy of *ModSBTn1_RF* is slightly lower compared to *SBT1_RF* and *ModSBTn2_RF*. This is mainly due to the fact that the ModSBTn (modified normalized Soil Behaviour Type) is calculated from normalized parameters and this model (*ModSBTn1_RF*) has no input features considering the vertical stresses and the groundwater conditions. However, the bias and variance of both Mod.SBTN models is very low and therefore the hyperparameter are also set to default.

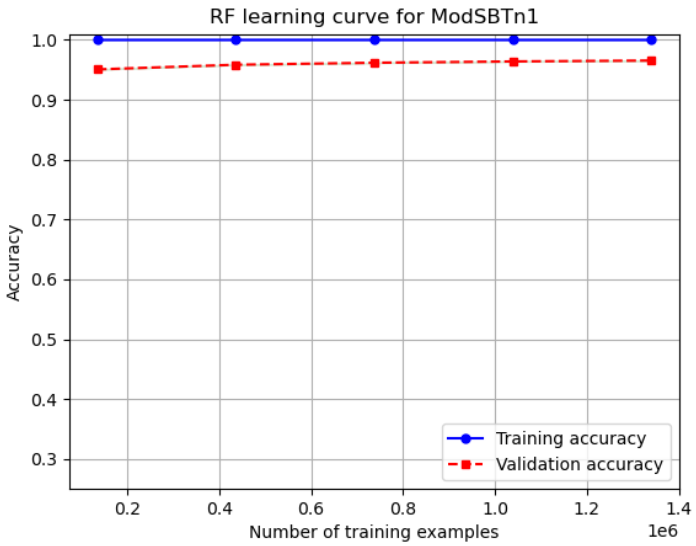

Figure 45: RF learning curve for ModSBTn1

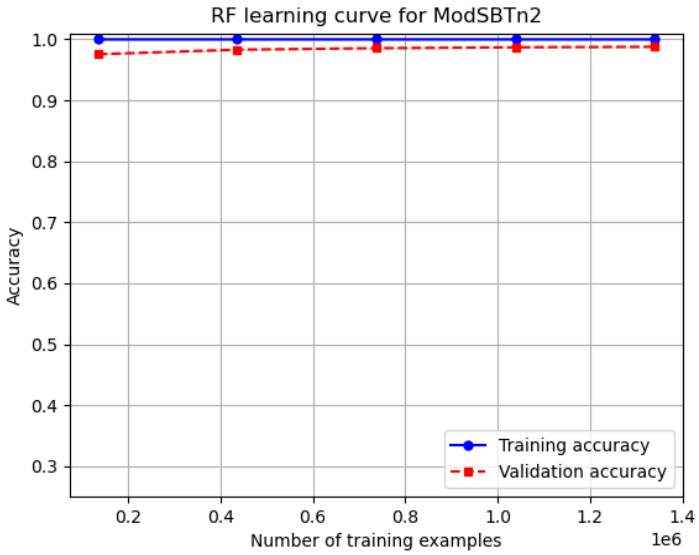

Figure 46: RF learning curve for ModSBTn1

The influence of the number of trees is also checked for the Soil Behaviour Type Models. In Figure 47, the validation curve for the model *ModSBTn2_RF* is

displayed. The range for the number trees is set to 10, 25, 50 and 100. The curves indicate that only a few estimators are necessary to get the best performance.

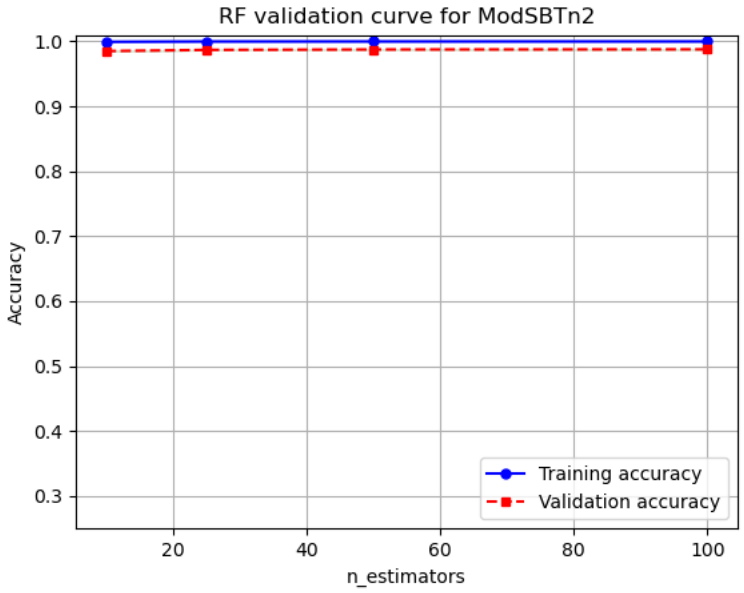

Figure 47: RF validation curve for a varying number of trees

In the next step, the performance of all models, targeting the different Soil Behaviour Types is evaluated using classification reports and confusion matrices. The reports and matrices are provided in the following sections and their results are discussed at the end.

7.2.1 SBT1_RF

Table 57: Confusion matrix for SBT1_RF

Confusion matrix											
SBT1_RF		Predicted									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Actual	**0**	11578	116	1	0	0	8	80	28	1	0
	1	66	6676	0	9	14	7	0	0	0	0
	2	2	1	16320	44	0	0	0	0	0	0
	3	0	11	62	113237	293	0	0	0	0	2
	4	0	22	0	334	101835	278	0	0	0	10
	5	8	14	0	0	309	121238	291	0	15	0
	6	69	1	0	0	0	297	113728	87	4	0
	7	27	0	0	0	0	0	108	13208	0	0
	8	0	0	0	0	1	9	3	0	1201	4
	9	0	0	0	2	8	0	0	0	4	1182

Table 58: Classification report for SBT1_RF

SBT1_RF	Classification report	
	macro avg	weighted avg
accuracy	0,99	
precision	0,99	0,99
recall	0,99	0,99
f1-score	0,99	0,99

7.2.2 SBT2_RF

Table 59: Classification report for SBT2_RF

SBT2_RF	Classification report	
	macro avg	weighted avg
accuracy	0,99	
precision	0,99	0,99
recall	0,99	0,99
f1-score	0,99	0,99

Table 60: Confusion matrix for SBT2_RF

Confusion matrix											
SBT2_RF		Predicted									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Actual	**0**	11552	101	2	0	0	8	95	41	0	0
	1	51	6810	1	8	19	10	0	0	0	0
	2	1	0	16183	87	0	0	0	0	0	0
	3	0	17	87	113691	380	0	0	0	0	7
	4	0	29	0	498	101709	325	0	0	0	18
	5	10	17	0	0	398	120832	354	0	10	0
	6	89	1	0	0	0	345	113224	117	2	0
	7	46	0	0	0	0	0	136	13116	0	0
	8	0	0	0	0	1	13	8	0	1181	6
	9	0	0	0	4	6	0	0	0	5	1202

7.2.3 SBTn1_RF

Table 61: Classification report for SBTn1_RF

SBTn1_RF	Classification report	
	macro avg	weighted avg
accuracy	0,97	
precision	0,96	0,97
recall	0,96	0,97
f1-score	0,96	0,97

Table 62: Confusion matrix for SBTn1_RF

Confusion matrix											
SBTn1_RF		Predicted									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Actual	**0**	16487	42	75	2	0	2	75	68	3	0
	1	75	5815	4	63	32	7	1	0	0	0
	2	57	4	26708	1039	1	0	0	0	0	3
	3	0	41	641	143441	2227	6	0	0	0	220
	4	0	27	1	1267	57091	2195	0	0	49	215
	5	6	9	0	1	1193	106018	2675	0	180	0
	6	33	2	0	0	0	1597	104070	636	60	0
	7	58	0	0	0	0	0	413	18250	0	0
	8	0	0	0	0	23	154	54	0	3631	52
	9	0	0	0	115	116	0	0	0	70	5453

7.2.4 SBTn2_RF

Table 63: Classification report for SBTn2_RF

SBTn2_RF	Classification report	
	macro avg	weighted avg
accuracy	0,99	
precision	0,98	0,99
recall	0,98	0,99
f1-score	0,98	0,99

Table 64: Confusion matrix for SBTn2_RF

Confusion matrix											
SBTn2_RF		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	16471	41	85	1	0	2	79	72	3	0
	1	91	5843	4	38	16	4	1	0	0	0
	2	66	6	27107	627	1	0	0	0	0	5
	3	2	28	448	145430	589	7	0	0	0	72
	4	0	21	0	695	59509	567	0	0	5	48
	5	5	4	0	1	568	108797	634	0	73	0
	6	30	0	0	0	0	623	105506	203	36	0
	7	41	0	0	0	0	0	226	18454	0	0
	8	0	0	0	0	1	81	33	1	3769	29
	9	0	0	1	47	43	2	0	0	35	5626

7.2.5 ModSBTn1_RF

Table 65: Classification report for ModSBTn1_RF

ModSBTn1_RF	Classification report	
	macro avg	weighted avg
accuracy	0,97	
precision	0,96	0,97
recall	0,96	0,97
f1-score	0,96	0,97

Table 66: Confusion matrix for ModSBTn1_RF

Confusion matrix									
ModSBTn1_RF		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	18221	114	67	0	167	74	113	151
	1	108	38036	567	1	462	0	6	3
	2	16	416	118454	1836	497	336	0	0
	3	0	1	894	33193	0	576	0	0
	4	13	230	343	0	28480	897	884	341
	5	7	0	83	316	335	31134	0	827
	6	34	3	0	0	533	1	62236	2768
	7	73	0	0	0	73	510	1409	157014

7.2.6 ModSBTn2_RF

Table 67: Classification report for ModSBTn2_RF

ModSBTn2_RF	Classification report	
	macro avg	weighted avg
accuracy	0,99	
precision	0,99	0,99
recall	0,98	0,99
f1-score	0,98	0,99

Table 68: Confusion matrix for ModSBTn2_RF

Confusion matrix									
ModSBTn2_RF		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	18243	114	69	0	142	78	122	139
	1	110	38476	477	0	113	0	4	3
	2	17	367	120717	286	154	13	0	1
	3	0	2	267	34202	1	192	0	0
	4	15	113	220	0	30499	89	238	14
	5	6	0	3	197	111	32106	1	278
	6	35	2	0	0	245	2	64811	480
	7	48	0	0	0	5	244	491	158291

7.2.7 Summary of results

The models built with random forest algorithm predict with the highest scores for all targets and input features, compared to the SVM and ANN model. The Random Forest models for the Soil Behaviour Type classifications predict throughout with a high accuracy between 97 % and 99 %. The training time of the models is also better compared to the neural network models and lays within 10.5 to 11.5 minutes. Additionally, all classes were captured with nearly the same accuracy regardless the class balance.

Table 69: Results of SBT models

Target	Model ID	Algorithm	Features	Accuracy
SBT	SBT1_RF	RF	Depth, q_c, f_s, R_f	0,99
	SBT2_RF	RF	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,99
SBTn	SBTn1_RF	RF	Depth, q_c, f_s, R_f	0,97
	SBTn2_RF	RF	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,99
Mod.SBTn	MSBTn1_RF	RF	Depth, q_c, f_s, R_f	0,97
	MSBTn2_RF	RF	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,99

7.3 Discussion

The Random Forest models predict best, both in prediction performance (accuracy) and training time. Additionally, the Random Forest algorithm is the easiest to apply compared to the SVM and ANN. For the prediction of the Soil Behaviour Types, no hyperparameter tuning was necessary and all models are built with the default parameter set. Since the Random Forest reached the best scores for all evaluated combinations of input features and targets, this model will be used for the prediction of soil classes from unknown CPT test data.

8 Prediction and classification of soil strata

Since the best prediction accuracies obtained in this thesis are from the Random Forest models, these models are used to predict soil strata and classes from unseen CPT data from sites in Austria and the Netherlands. The accuracies of all models are summarized in Table 70. To compare the predicted with the actual soil classes and interpret the resulting soil strata, the classes are visualized over depth, together with q_c and R_f from the respective CPT test. (Parts of the code to generate the charts are based on a template from the blog "Geotech_4.0" (Yogatama, 2018).)

Table 70: Summary of the accuracies of all evaluated models

Target	Model ID	Algorithm	Features	Accuracy
OC	OC1_SVM	SVM	Depth, q_c, f_s, R_f	0,38
	OC2_SVM	SVM	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,38
	OC1_ANN	ANN	Depth, q_c, f_s, R_f	0,46
	OC2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	O,47
	OC1_RF	RF	Depth, q_c, f_s, R_f	0,65
	OC2_RF	RF	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,75
SBT	SBT1_ANN	ANN	Depth, q_c, f_s, R_f	0,98
	SBT2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,98
	SBT1_RF	RF	Depth, q_c, f_s, R_f	0,99
	SBT2_RF	RF	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,99
SBTn	SBTn1_ANN	ANN	Depth, q_c, f_s, R_f	0,95
	SBTn2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,97
	SBTn1_RF	RF	Depth, q_c, f_s, R_f	0,97
	SBTn2_RF	RF	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,99
Mod.SBTn	MSBTn1_ANN	ANN	Depth, q_c, f_s, R_f	0,94
	MSBTn2_ANN	ANN	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,98
	MSBTn1_RF	RF	Depth, q_c, f_s, R_f	0,97
	MSBTn2_RF	RF	Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f	0,99

8.1 CPT data from Austria

The test data used for the prediction is also part of the dataset used in the sections before but was excluded for the training process. The used tests are ID_934 from Salzburg basin and ID_1317 from Zell basin.

To predict the soil strata and classes (acc. to Oberhollenzer, et.al., (2021)) and the modified normalized Soil Behaviour types (acc. to Robertson (2016)), the Random Forest models are trained again using the entire dataset (without applying train test split). With the input features (Depth, q_c, f_s, σ_v, u_0, σ'_v, R_f) from the respective test a soil class of Soil Behaviour Type is predicted for each depth level where data is measured by the CPT.

8.1.1 ID_934

8.1.1.1 Oberhollenzer_classes

The results for test ID_934 are visualized in Figure 48 and in Figure 49 as comparison to the real results. The charts of Figure 48 provide the following information (from left to right): First, the CPT test data q_c and R_f vs the depth, second, the soil classes predicted by the Machine Learning model and third the probability of each class vs the depth of the prediction. The last two charts of Figure 49 provide the actual soil classes and soil model determined by laboratory tests from core drilling samples by Oberhollenzer, et. al. (2021).

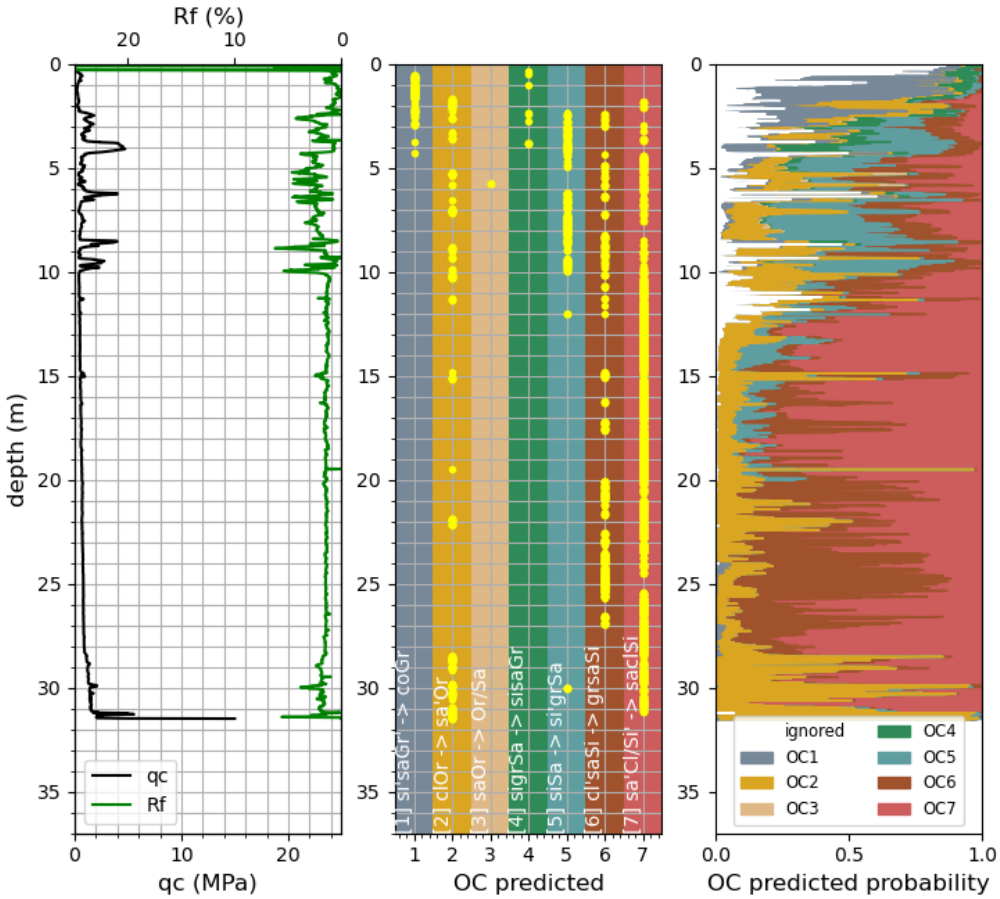

Figure 48: Predicted soil classes of test ID_934 with probability plot

The yellow points in “OC predicted” and “OC real” indicated the predicted soil class from the CPT data and the actual soil class based on grain size distribution. For this example, the Machine Learning model yielded adequate predictions compared to the real soil model. Down to 5 m beneath ground elevation the model identifies a mixture of sand and gravel and from 5-10 m a mixture of sand, silt and clay, and below 10 m the model predicts mainly clay and silt. Parts where no predicted or actual soil class is visualized belong to the ignored classifications (a full list of the soil classes is provided in the appendix and more information about the determination of these classes is provided in the publication of Oberhollenzer et. al., (2021)).

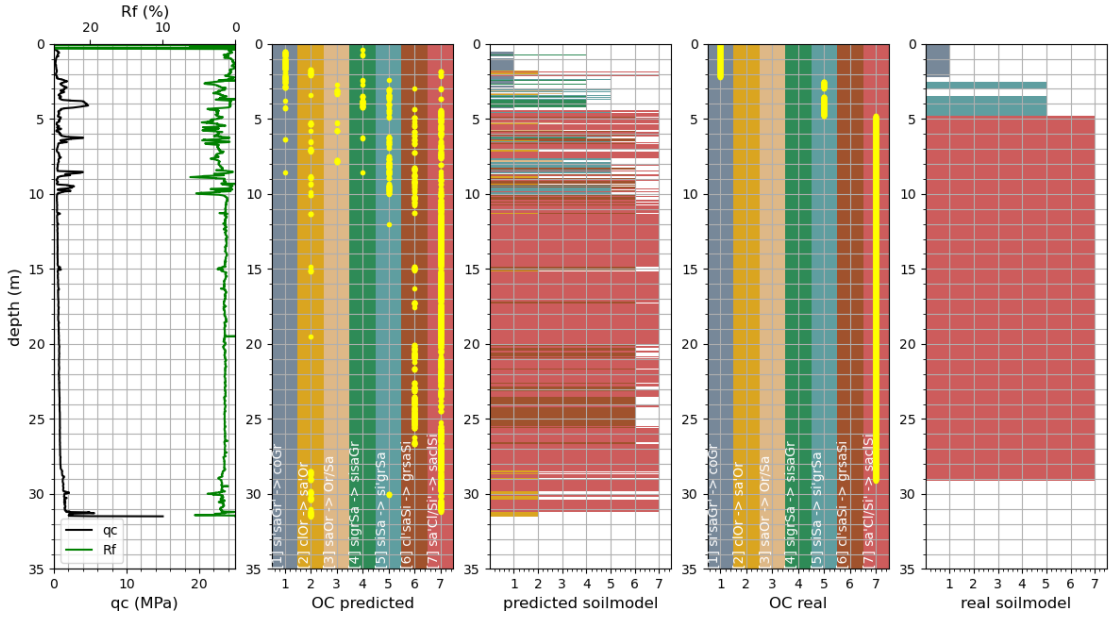

Figure 49: Predicted vs actual soil classes of test ID_934

The confusion matrix and classification report are given in Table 72 and Table 71. The accuracy of this prediction is lower than obtained in the building process of the Random Forest model (chapter 7).

Table 71: Confusion matrix test ID_934

Confusion matrix									
ID_934 OC		Predicted							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Actual	**0**	8	21	109	2	8	19	12	145
	1	59	116	26	0	9	0	1	9
	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0
	5	35	27	15	0	14	51	10	23
	6	0	0	0	0	0	0	0	0
	7	26	4	180	2	9	203	563	1443

Table 72: Classification report for test ID_934

ID_934 OC	Classification report	
	macro avg	weighted avg
accuracy	0,57	
precision	0,22	0,75
recall	0,17	0,57
f1-score	0,19	0,65

8.1.1.2 Soil Behaviour Types

The 3 different Soil Behaviour Types (SBT, SBTn and Mod. SBTn) are quite similar. Therefore, the results for the prediction of the Machine Learning model with unseen CPT data are plotted only for the modified normalized Soil Behaviour Type – Mod. SBTn, representative for all. The results are displayed in Figure 50. As expected, the model predicted with nearly the same accuracy as in the building process and only a few datapoints are misclassified. Due to the empirical relationship between test data and Soil Behaviour Types, the accuracy is better than for predicting Oberhollenzer_classes.

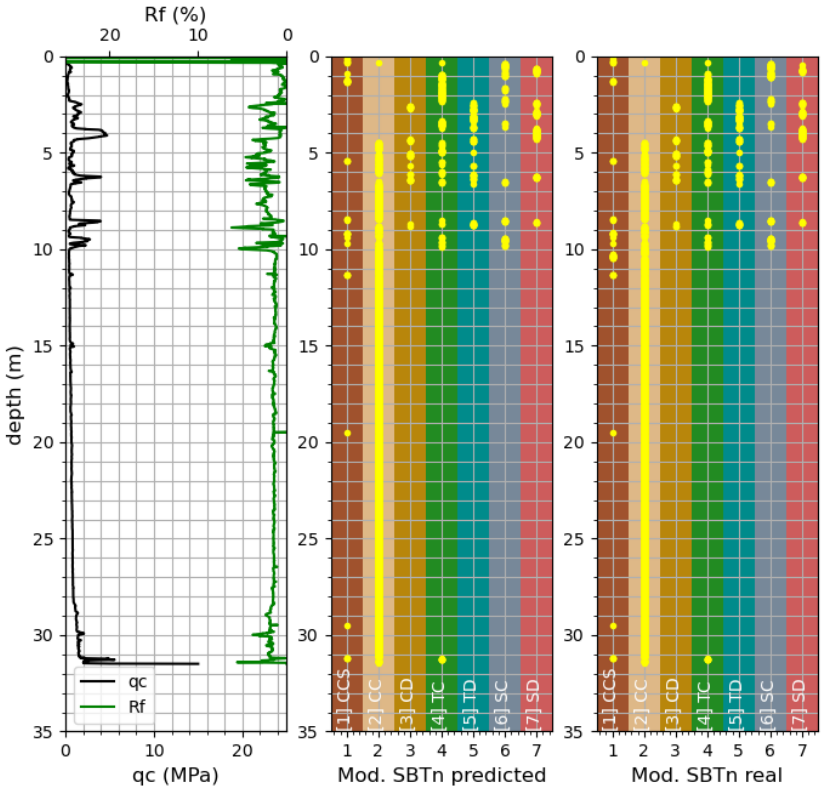

Figure 50: Predicted vs actual Soil Behaviour Type (Mod. SBTn) of test ID_934

The classification report and confusion matrix are provided in Table 73 and Table 74.

Table 73: Classification report for test ID_934

ID_934 Mod. SBTn	Classification report	
	macro avg	weighted avg
accuracy	0,98	
precision	0,94	0,98
recall	0,90	0,98
f1-score	0,92	0,98

Table 74: Confusion matrix test ID_934

Confusion matrix									
ID_934 Mod. SBTn		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	45	1	0	0	0	0	0	0
	1	0	34	18	0	0	0	0	0
	2	0	0	2468	0	0	0	0	0
	3	0	0	3	45	0	1	0	0
	4	1	6	8	0	183	1	10	2
	5	0	0	0	1	1	97	0	0
	6	0	0	0	0	10	0	88	9
	7	0	0	0	0	0	0	3	114

8.1.2 ID_1317

8.1.2.1 Oberhollenzer_classes

The results for the CPT data with the ID 1317 are provided in Figure 52. The model predicted mainly class 2 for the first 2 m, then up to 7 m class “0” (= ignored classes). From 7 to 20 m, the model predicted predominantly sand with some minor layers of silt and organic soils. The soil model generated by the Machine Learning model is quite similar to the actual soil model which was determined from adjacent boreholes. The obtained accuracy is 78 % and therefore higher than obtained while building the Random Forest model (75 %).

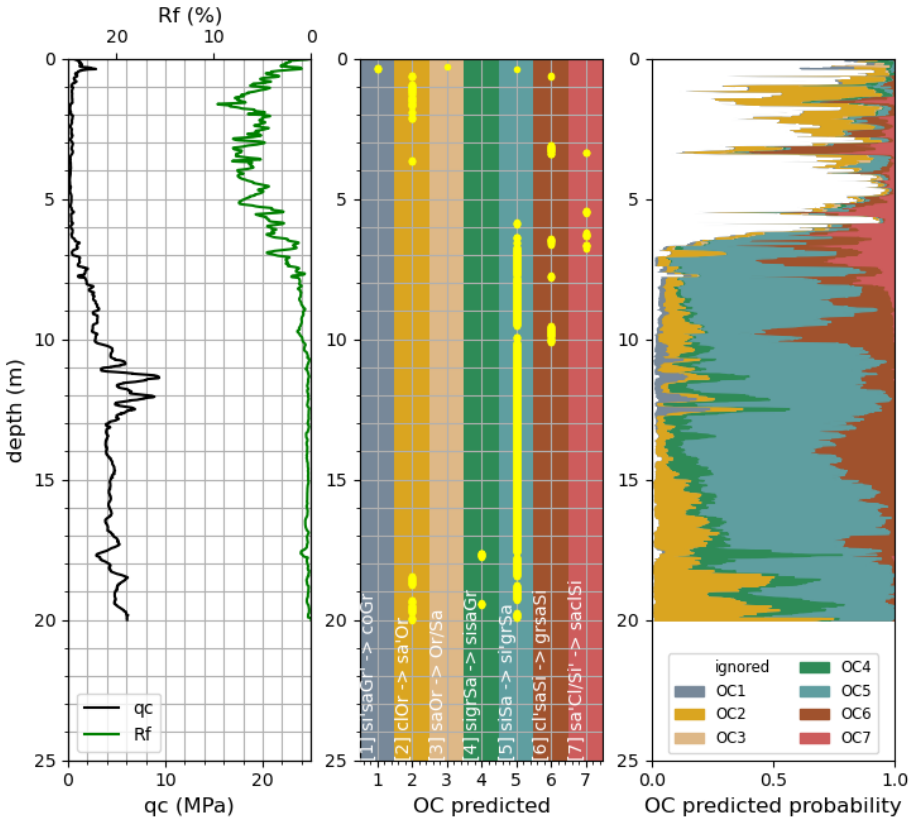

Figure 51: Predicted soil classes of test ID_1317 with probability plot

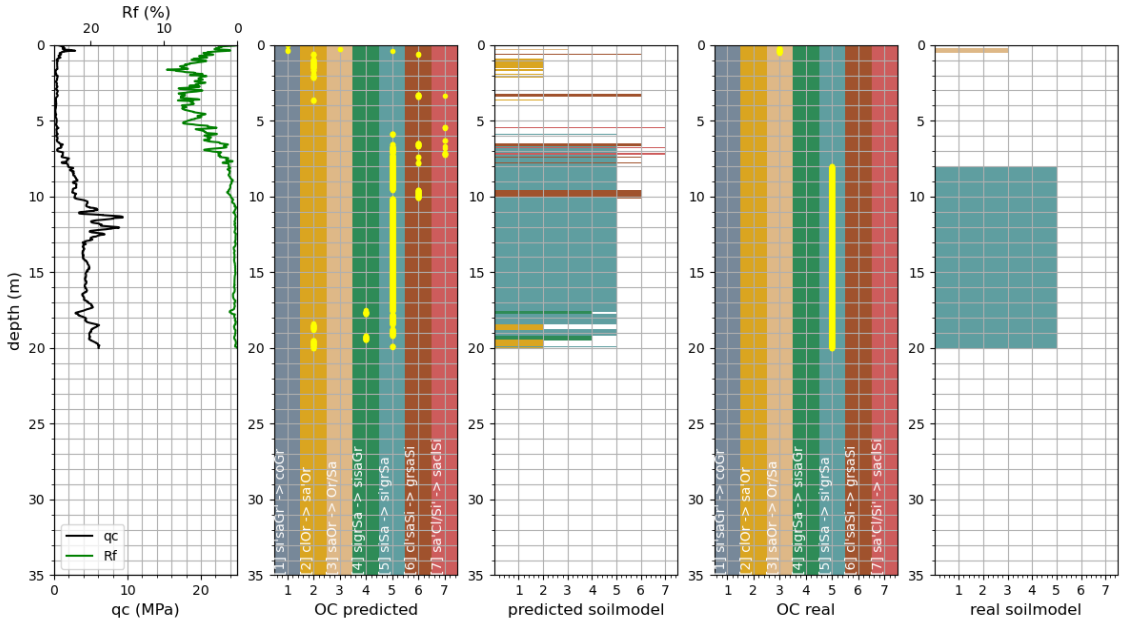

Figure 52: Predicted vs actual soil classes of test ID_1317

The classification report and the confusion matrix also indicate a solid model performance. They are provided in Table 75 and Table 76.

Table 75: Classification report for test ID_1317

ID_1317 OC	Classification report	
	macro avg	weighted avg
accuracy	0,78	
precision	0,36	0,92
recall	0,21	0,78
f1-score	0,24	0,83

Table 76: Confusion matrix test ID_1317

Confusion matrix									
ID_1317 OC		Predicted							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Actual	**0**	520	1	71	0	0	125	42	11
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
	3	20	6	0	4	0	0	0	0
	4	0	0	0	0	0	0	0	0
	5	0	0	80	0	47	1026	47	0
	6	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0

8.1.2.2 Soil Behaviour Types

Similar to test ID_934, the results for the Soil Behaviour Type predictions are plotted only once. Again, the modified normalized Soil Behaviour Type is predicted and the results are visualized in Figure 53 The confusion matrix and classification report are provided in Table 77 and Table 78.

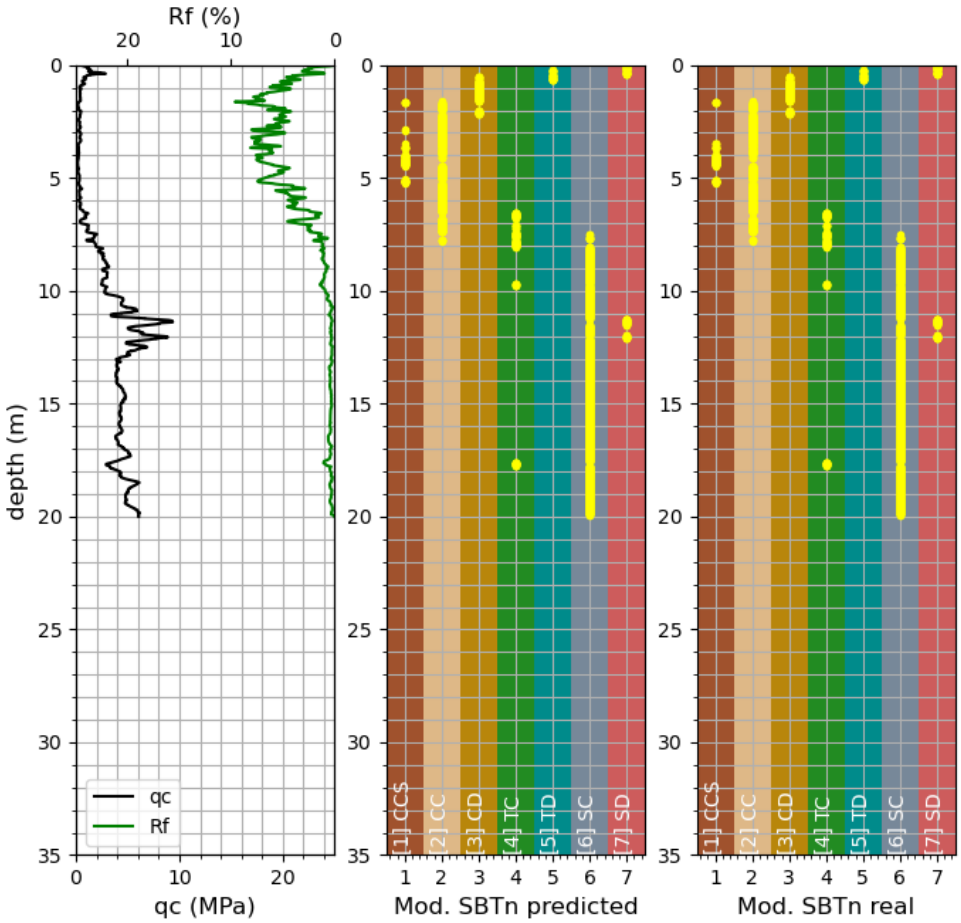

Figure 53: Predicted vs actual Soil Behaviour Type (Mod. SBTn) of test ID_1317

Table 77: Confusion matrix test ID_1317

Confusion matrix									
ID_1317 Mod.SBTn		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	8	0	0	0	0	0	2	0
	1	0	70	3	0	0	0	0	0
	2	0	6	451	2	6	0	0	0
	3	0	0	0	116	0	0	0	0
	4	0	0	0	0	105	0	0	0
	5	0	0	0	0	0	26	0	1
	6	0	0	0	0	0	0	1131	1
	7	0	0	0	0	0	0	0	72

Table 78: Classification report for test ID_1317

ID_1317 Mod. SBTn	Classification report	
	macro avg	weighted avg
accuracy	0,99	
precision	0,98	0,99
recall	0,96	0,99
f1-score	0,97	0,99

8.2 CPT data from the Netherlands

The CPT data used for the prediction is from sites in the Netherlands and very generously provided by the Dutch company "*Witteveen+Bos*". To compare the prediction of the model with the soil model from adjacent boreholes, the borehole classification is converted to the soil classes according to Oberhollenzer (2021). An example for the conversion of the test with ID 1417_NL is given in Figure 54. It as to be kept in mind that this conversion includes some assumptions and thus uncertainties.

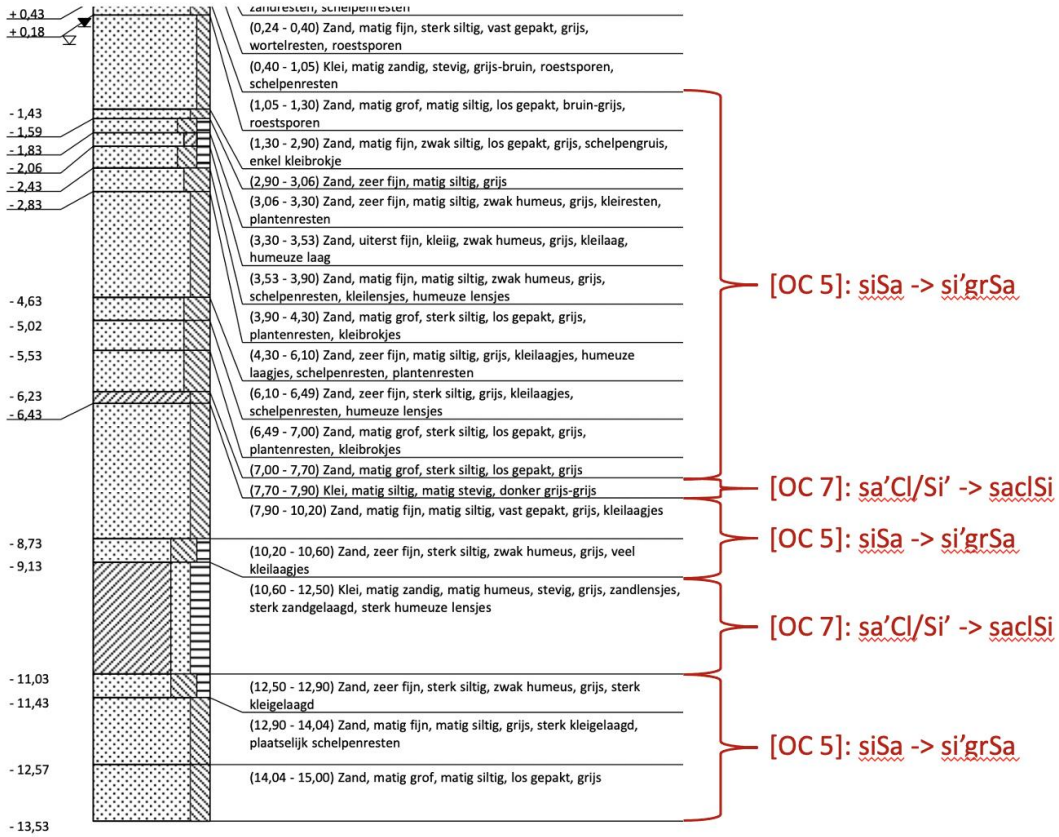

Figure 54: Conversion from soil classification of Dutch borehole samples to Oberhollenzer_classes

8.2.1 ID_1417_NL

8.2.1.1 Oberhollenzer_classes

The resulting predicted soil classes for the Dutch CPT data with the ID 1417_NL are provided in Figure 55 and Figure 56. The borehole classification is available to a depth of about 13.5 m. For this part, the Oberhollenzer_classes are determined as mentioned before and it is assumed to be predominantly class 5 (sand) with interlayers of class 7 (clay). The Machine Learning model predicts majorly sandy to gravely soil until a depth of about 7 m. After that, mainly a mixture of sand and silt. Although the model is not able to identify the correct soil strata and classes for the whole core sample, it indicates the distribution of fine- and coarse-grained soils over the depth.

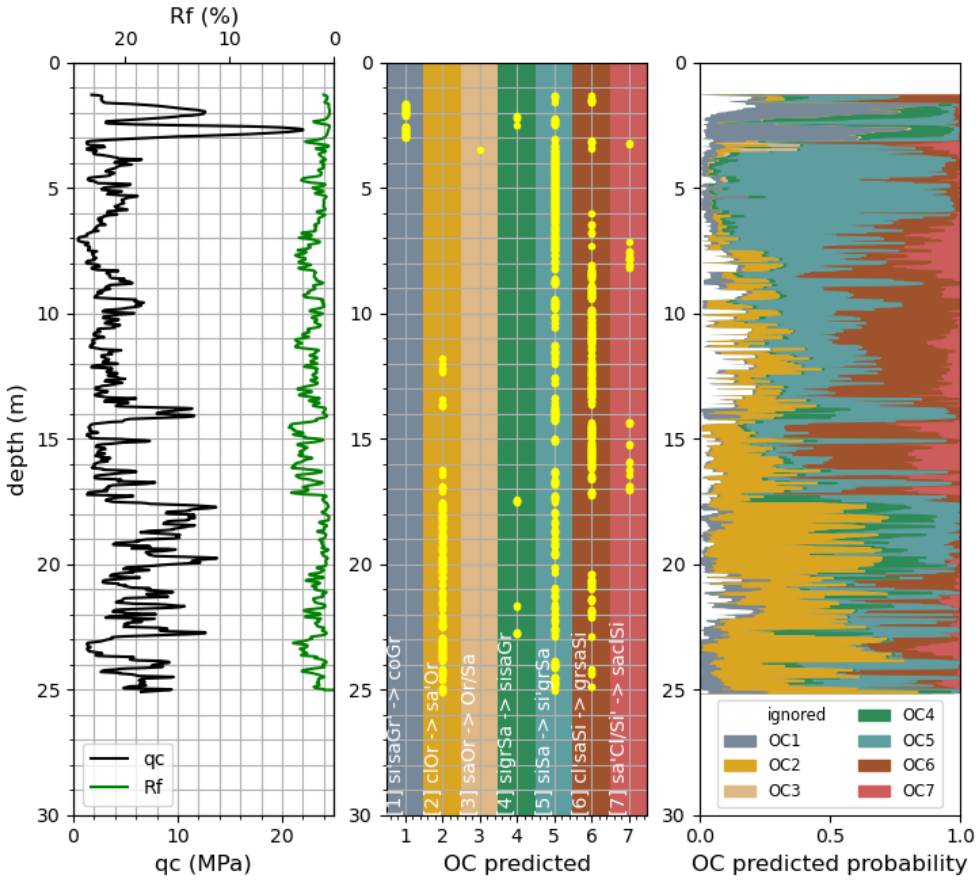

Figure 55: Soil classification for CPT test ID_1417_NL from *Witteveen+Bos* with the Random Forest Model

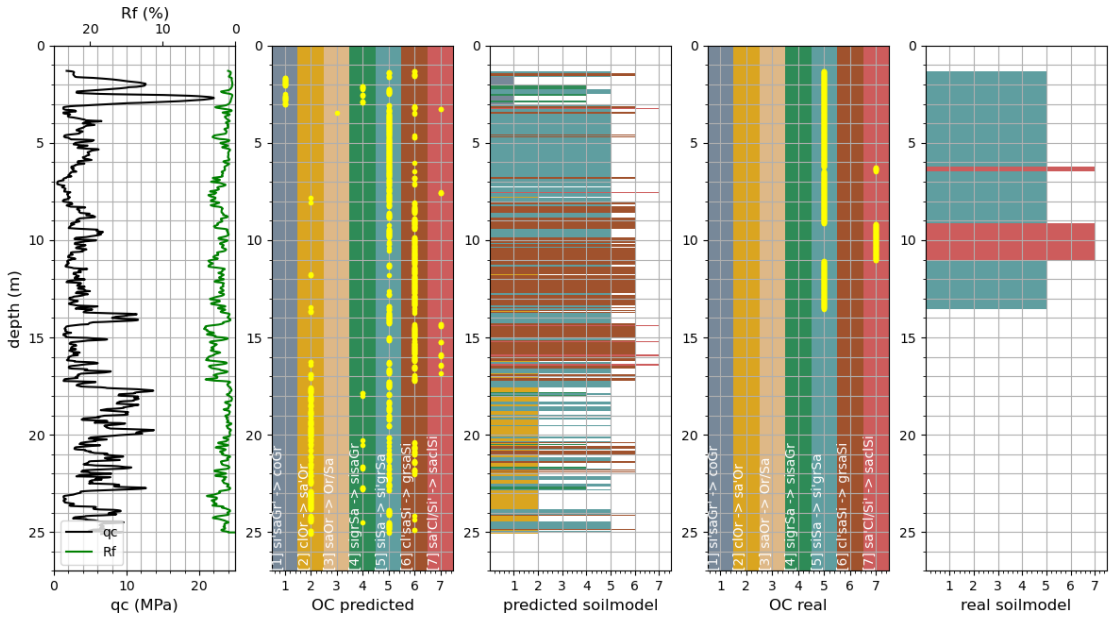

Figure 56: Comparison of predicted with real soil classes of CPT ID_1417_NL

8.2.1.2 Soil Behaviour Types

The classification report and confusion matrix of the modified normalized Soil Behaviour Type prediction are also generated for the CPTs from the Netherlands and displayed in Figure 57.

Similar to the predictions for Austrian CPTs, the Soil Behaviour Types are identified with high accuracy by the Random Forest Model for the Dutch CPTs, too. The confusion matrix and classification report are provided in Table 80 and Table 79

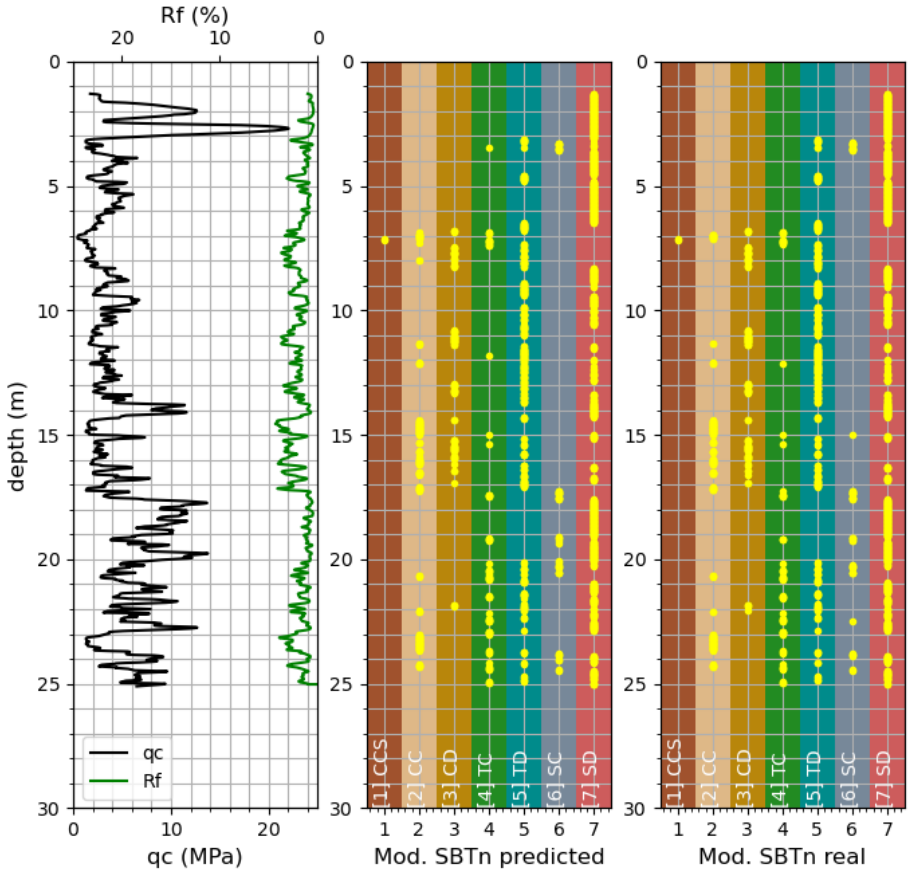

Figure 57: Predicted vs actual Soil Behaviour Type (Mod. SBTn) of test ID_1417_NL

Table 79: Confusion matrix test ID_1417_NL

Confusion matrix									
ID_1417_NL Mod. SBTn		Predicted							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Actual	**0**	3	0	0	0	2	2	0	1
	1	0	2	0	0	0	0	0	0
	2	0	0	119	1	0	0	0	0
	3	0	0	5	83	0	8	0	0
	4	0	0	4	0	58	2	0	0
	5	0	0	0	1	6	254	0	4
	6	0	0	0	0	3	0	40	2
	7	0	0	0	0	0	3	8	580

Table 80: Classification report for test ID_1417_NL

ID_1417_NL Mod. SBTn	Classification report	
	macro avg	weighted avg
accuracy	0,96	
precision	0,94	0,96
recall	0,87	0,96
f1-score	0,89	0,96

8.2.2 ID_3578_NL

8.2.2.1 Oberhollenzer_classes

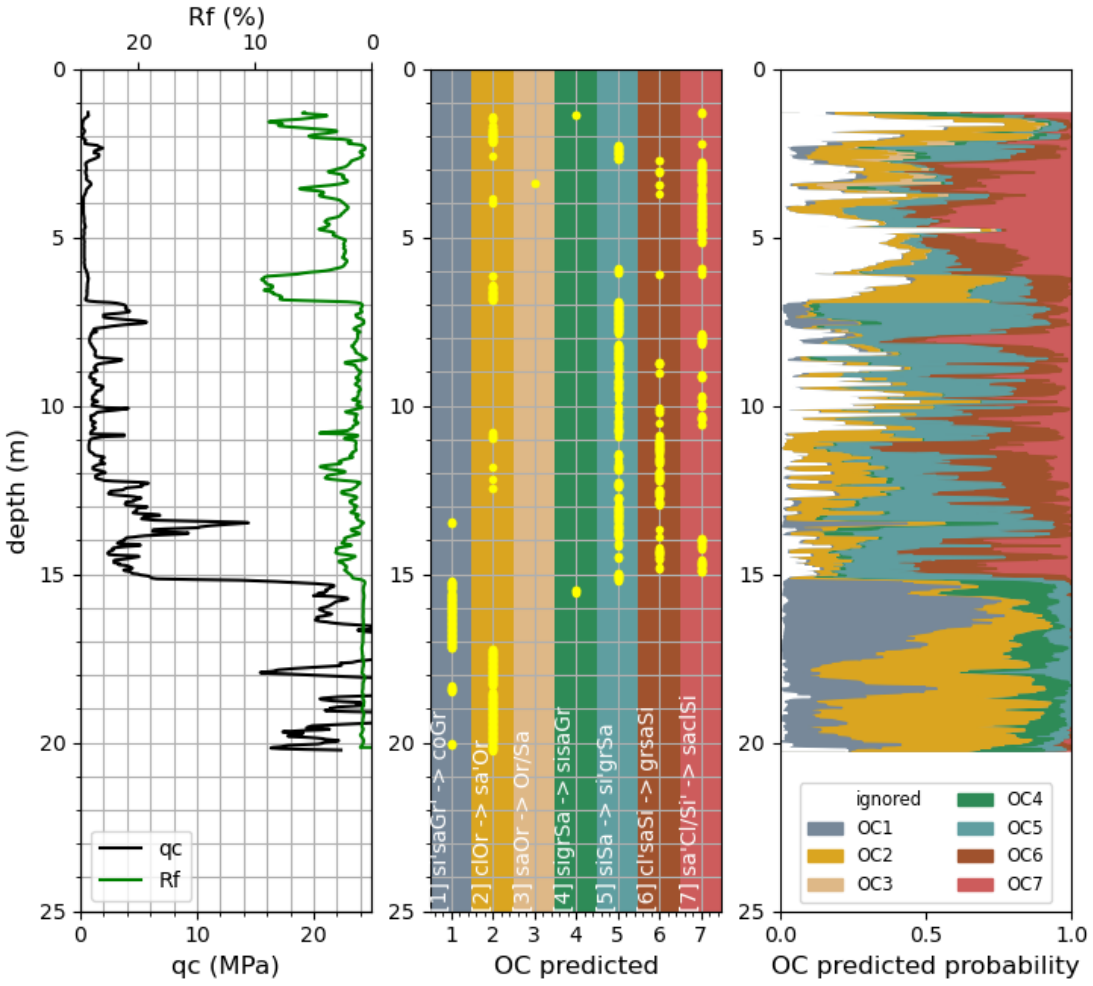

Figure 58: Soil classification for CPT test ID_3578_NL

A second CPT is used to evaluate the model performance on foreign soils. The results are displayed in Figure 58 and Figure 59. The soils identified from the borehole are as follows: Until a depth of 5 m a layer of predominantly clay, at 5 m a layer of peat, and after 5.5 m alternate layers of sand and clay until the final depth of about 14 m.

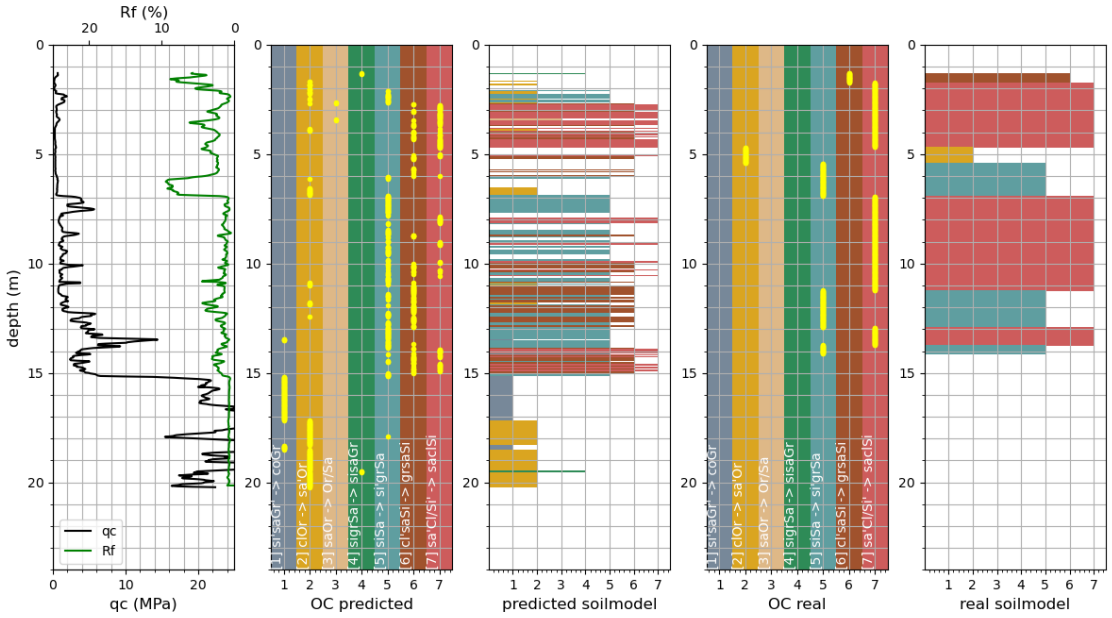

Figure 59: Predicted vs actual Soil Behaviour Type (Mod. SBTn) of test ID_3578_NL

The soils predicted by the Random Forest model are: Down to 5 m mainly sandy, silty clay. At 5 m a layer of soils ignored by Oberhollenzer et.al., (class 0). Between 6 and 7 m a layer of organic soils. From 7 to 9 a mixture of sand and clay and from 9 to 14 m predominantly a mixture of sand, silt and clay.

The model was again not able to identify all soil classes and strata, but similar to the test before it indicates the distribution of fine- and coarse-grained soils in the subsurface. The prediction could be improved by adding more training data from similar ground conditions. This is part of ongoing research at the Institute of Soil Mechanics, Foundation Engineering and Computational Geotechnics at the Graz University of Technology.

8.2.2.2 Soil Behaviour Types

The results for the test with ID_3578 from the Netherlands are provided in Figure 60.

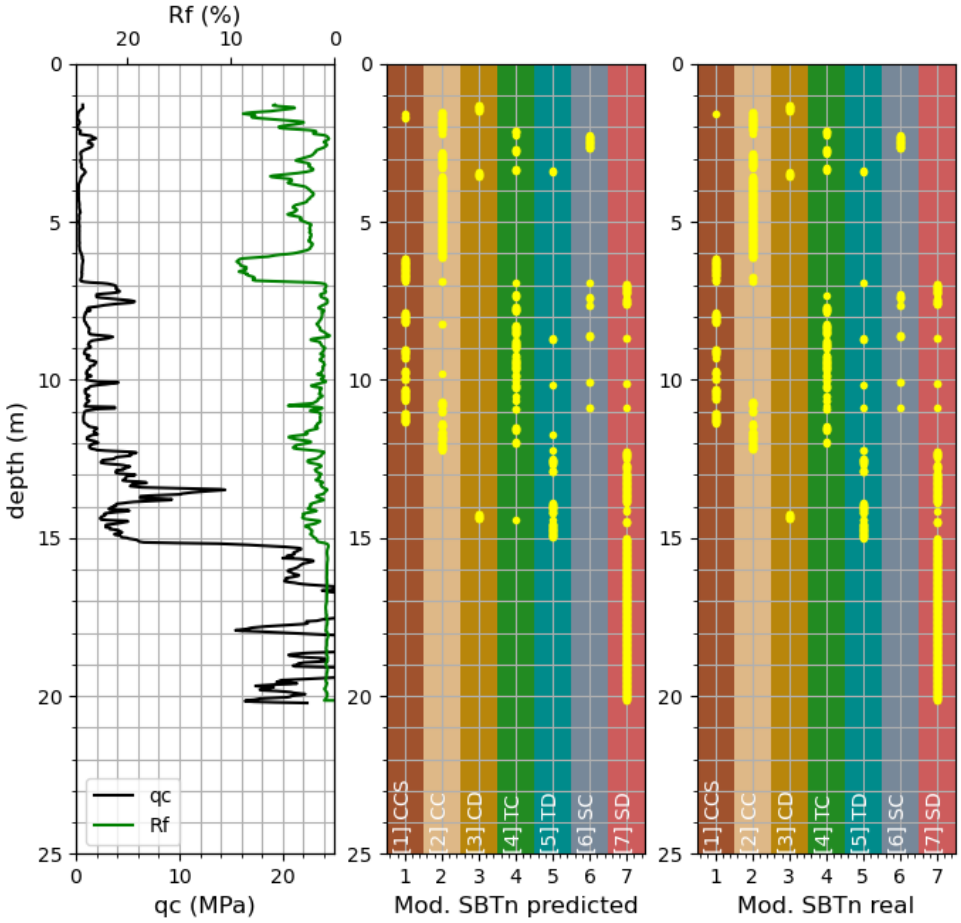

Figure 60: Predicted vs actual Soil Behaviour Type (Mod. SBTn) of test ID_3578_NL

The classification report and confusion matrix are provided in Table 81 and Table 82. The model is again able to identify most of the classes and yields an accuracy of 96 % which is 3 % less than while building the model (chapter 7.2.6).

Table 81: Classification report for test ID_3578_NL

ID_3578_NL Mod. SBTn	**Classification report**	
	macro avg	weighted avg
accuracy	0,96	
precision	0,96	0,96
recall	0,88	0,96
f1-score	0,91	0,96

Table 82: Confusion matrix test ID_3578_NL

Confusion matrix									
ID_3578_NL Mod. SBTn		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	4	0	2	0	2	0	1	0
	1	0	103	2	0	0	0	0	0
	2	0	4	227	0	0	0	0	0
	3	0	0	0	27	0	0	0	0
	4	0	8	8	0	94	0	0	0
	5	0	0	1	0	2	60	0	4
	6	0	0	0	0	3	0	33	1
	7	0	0	0	0	0	0	1	360

8.2.3 ID_14001_NL

For the third Dutch Cone Penetration Test data is no soil classification based on grain size distribution available. Hence the Random Forest model is only evaluated with respect to the *Mod. SBTn*. The confusion matrix and classification report are provided in Table 83 and Table 84 and the results are displayed in Figure 61. The results show again good consistency with the Soil Behaviour Types determined with empirical correlations.

Table 83: Confusion matrix test ID_14001_NL

Confusion matrix									
ID_934 Mod. SBTn		Predicted							
		0	1	2	3	4	5	6	7
Actual	0	6	0	0	0	1	0	0	0
	1	0	17	11	0	0	0	0	0
	2	0	0	276	6	2	0	0	0
	3	0	0	1	181	0	3	0	0
	4	0	0	3	0	105	1	1	0
	5	0	0	0	10	5	116	0	0
	6	0	0	0	0	8	0	679	41
	7	0	0	0	0	1	19	52	2129

Table 84: Classification report for test ID_14001_NL

ID_934 Mod. SBTn	Classification report	
	macro avg	weighted avg
accuracy	0,96	
precision	0,93	0,96
recall	0,89	0,96
f1-score	0,91	0,96

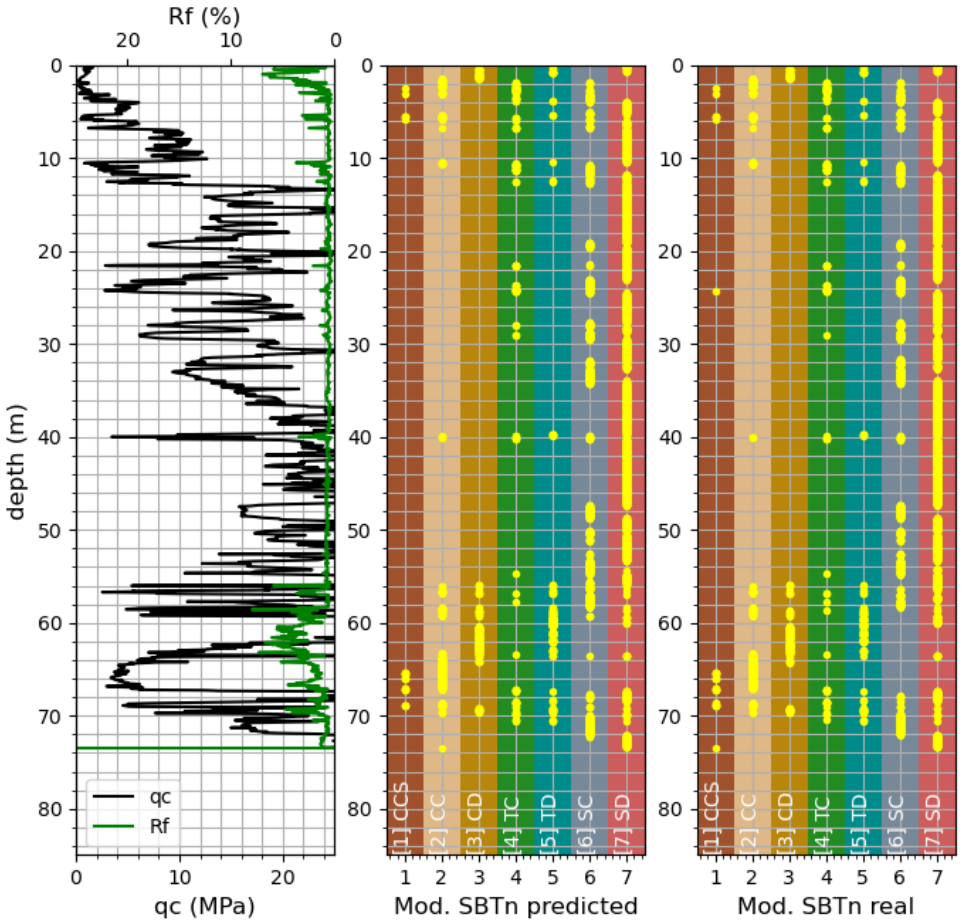

Figure 61: Predicted vs actual Soil Behaviour Type (Mod. SBTn) of test ID_14001_NL

9 Conclusion

In this master thesis, the ability of a Machine Learning algorithm to classify soils from Cone Penetration Test data was evaluated. In addition, the applied algorithms were compared, and the best performing was then used to classify soil strata and classes from unseen CPT data.

The study of the current literature and publications regarding Machine Learning in geotechnics revealed an increased research interest over the past 10 years in the fields of artificial intelligence in geosciences. In recent publications, various studies showed that Machine Learning could be a feasible tool for the interpretation of subsurface conditions.

To evaluate if a Machine Learning model is capable of soil classification from CPT data, 24 models based on three different algorithms, namely Support Vector Machine, Artificial Neural Network and Random Forest were built and tested. As input features, direct measured data (raw data) like tip resistance q_c and sleeve friction f_s as well as defined data, e.g., vertical stresses σ_v or hydrostatic pore pressures u_0 are used. As targets, soil classes based on grain size distribution (acc. to Oberhollenzer et. al., (2021)) and soil classes based on the Soil Behaviour Types (acc. to Robertson (1990, 2009, 2016)). Furthermore, the models were compared mainly based on their prediction accuracy and secondarily on the necessary training time.

The studies showed that models built with a Support Vector Machine are not suitable for the desired problem. The models based on Artificial Neural Network yielded adequate results for the prediction of Soil Behaviour Types but performed not as good as Random Forest models when predicting soil classes based on grain size distribution (Oberhollenzer_classes). The models built with a Random Forest classifier performed best for each examined set of input features and target classes, hence they were used to identify soil strata and predict the respective soil class or type from unseen CPT data:

- **Classification based on grain size distribution**

  The Random Forest model was able to classify soils into classes based on grain size distribution from CPT test data with an accuracy of about 75 %. In addition, the results of the predicted soil strata and classes compared to the real ones were sufficient enough to indicate the mostly present class on a specific depth for CPT from Austria. In contrary, the model performed worse on unseen CPT data from the Netherlands. This leads to the following concluding remarks:

- A random forest model is principally able to classify soils from CPT data into soil classes based on grain size distribution.
- The accuracy of the prediction is, seen from a geotechnical perspective, amongst others also influenced by the origin of the training samples, therefore, the models trained on Austrian soils performed worse when tested on CPT data from Dutch sites. This could be improved by adding additional training samples from Dutch sites.
- Since the learning models were able to distinguish between different soil types (coarse to fine grained), it can be assumed that it could also be a helpful tool in the determination of physical properties of the subsurface, e.g., determination of friction angle.

- **Classification based on Soil Behaviour Types**

  The Random Forest Model was able to classify soils from CPT data targeting the Soil Behaviour Types with an accuracy of 96 % and beyond. The main findings for this type of classification can be summarized as follows:

  - A Machine Learning model is able to predict Soil Behaviour Types from CPT data with a high accuracy. The empirical relationship between the input and output is assumed to be the main reason for that.
  - Based on the obtained accuracies Machine Learning could be a helpful tool to interpret raw CPT data.
  - Besides Random Forest models, also models based on Artificial Neural Networks yielded accurate results.
  - The performance of the Support Vector Machine was not competitive compared to Random Forest and Artificial Neural Networks.

The present study showed that Machine Learning can be a suitable tool for soil classification in geotechnical engineering. However, the models evaluated in this study are not yet fully developed in terms of technology and there might be room for optimization. Additionally, the Artificial Neural Network is only applied in a simple form by using a multilayer perceptron. A Deep Neural Network may help to improve the prediction accuracies. Furthermore, the chosen input feature sets are determined only with respect to geotechnical properties and sequential feature selection algorithms were not applied.

The findings of this master thesis showed that Machine Learning could be a helpful tool in geotechnical engineering if a high amount of data is available. Besides soil classification, the application of Machine Learning models for parameter identification may be a promising field for future research.

10 Bibliograpy

Alkroosh, I., Bahadori, M., Nikraz, H., & Bahadori, A. (2015). Regressive approach for predicting bearing capacity of bored piles from cone penetration test data. *Journal of Rock Mechanics and Geotechnical Engineerinng, 7*, 584-592.

ASTM D2487-17e1. (2017). Standart Practice for Classification of Soils for Engineering Purposes (Unified Soil Classification System). West Conshohocken, PA: ASTM International.

EN ISO 14688-2. (2019). Geotechnical investigation and testing - Identification and classification of soil Part 2: Principles for a classification. Austrian Standards International.

Erzin , Y., & Ecemis, N. (2017). The use of neural networks for the prediction of cone penetration resistance of silty sands. *Neural Computing and Applications, 28*, 727-736.

Fortmann-Roe, S. (2012). *scott.fortmann-roe.com.* Retrieved 11 2020, from http://scott.fortmann-roe.com/docs/BiasVariance.html

Gajurel, A., Mukherjee, P. S., & Chittoori, B. (2019). Estimating Optimal Additive Content for Soil Stabilization Using Machine Learning Methods. In *Geo-Congress 2019: Geotechnical Materials, Modeling and Testing (GSP 310)* (pp. 662-672). American Society of Civil Engineers.

Goh, A. (1999). SOIL LABORATORY DATA INTERPRETATION USING GENERALIZED REGRESSION NEURAL NETWORK. *Civil Engineering and Environmental Systems, 16*(3), 175-195.

Harlianto, P. A., Adji, T. B., & Setiawan, N. A. (2017). Comparison of Machine Learning Algorithms for Soil Type Classification. *2017 3rd International Conference on Science and Technology - Computer (ICST)* (pp. 7-20). Yogyakarta: IEEE.

Heaton, J. (2015). *Artificial Intelligence for Humans, Volume 3: Neural Networks and Deep Learning* (Volume 3 ed.). Heaton Research, Inc.

Javadi, A., Mehravar, M., Faramarzi, A., & Ahangar-Asr, A. (2009). An Artificial Intelligence Based Finite Element Method. *ISAST Transactions on Computers and Intelligent Systems , 1*(2), 1-7.

Jin, Y.-F., & Yin, Z.-Y. (2020). An intelligent multi-objective EPR technique with multi-step model selection for correlations of soil properties. *Acta Geotechnica, 15*, 2053-2073.

Kirts, S., Panagopoulos, O., Xanthopoulos, P., & Nam, B. H. (2018). Soil-Compressibility Prediction Models Using Machine Learning. *Journal of Computing in Civil Engineering, 32*(1).

Kurup, P., & Griffin, E. (2006). Prediction of Soil Composition from CPT Data Using General Regression Neural Network. *Journal of Computing in Civil Engineering , 20*(4), 281-289.

Lemaitre, G., Nogueira, F., Oliveira D., & Aridas , C. (2020). *Imbalanced-learn*. Retrieved November 2020, from https://imbalanced-learn.readthedocs.io/en/stable/

Lunne, T., Robertson, P., & Powell, J. (1997). *Cone Penetration Testing in Geotechnical Practice.* Oxon: E & FN Spon.

Oberhollenzer , S., Premstaller, M., Marte, R., Tschuchnigg, F., Erharter, G., & Marcher, T. (2021). Cone Penetration test dataset Premstaller Geotechnik. *Data in Brief, 34*.

Padarian, J., Minasny, B., & Mcbratney, A. (2020). Machine learning and soil sciences: a review aided by machine learning tools. *SOIL, 6*(1), 35-52.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Brucher, M. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Puri, N., Prasad, H. D., & Jain, A. (2018). Prediction of Geotechnical Parameters Using Machine Learning Techniques. *Procedia Computer Science, 125*, 509-517.

Python Software Foundation. (2020). *Python.org*. Retrieved Oktober 2020, from https://www.python.org/doc/

Raschka, S. (2020). *sebastianraschka.com*. Retrieved 12 2020, from https://sebastianraschka.com/faq/docs/ml-solvable.html

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning* (Third Edition ed.). Birmingham: Packt Publishing Ltd.

Reale, C., Gavin, K., Librić, L., & Jurić-Kaćunić, D. (2018). Automatic classification of fine-grained soils using CPT measurements and Artificial Neural Networks. *Advanced Engineering Informatics, 36*, 207-215.

Robertson , P. (1990). Soil classification using the cone penetration test. *Canadian Geotechnical Journal, 27*, 151-158.

Robertson , P. (2009). Interpretation of Cone Penetration Tests - a unified approach. *Canadian Geotechnical Journal, 46*(11), 1337-1355.

Robertson , P. (2016). Cone penetration test (CPT)-based soil behaviour type (SBT) classification system - an update. *Canadian Geotechnical Journal , 53*, 1910-1927.

Samui, P. (2008). Prediction of friction capacity of driven piles in clay using the support vector machine. *Canadian Geotechnical Journal , 45*(2), 288-295.

scikit-learn developers. (2020). *scikit-learn* . Retrieved October 2020, from https://scikit-learn.org/stable/index.html

Shahin, M. (2015). A review of artificial intelligence applications in shallow foundations. *International Journal of Geotechnical Engineering, 9*(1), 49-60.

The Matplotlib development team. (2020). *Matplotlib*. Retrieved October 2020, from https://matplotlib.org/index.html#

The pandas development team. (2020). *Pandas*. Retrieved October 2020, from https://pandas.pydata.org/pandas-docs/stable/index.html

Tsiaousi, D., Travasarou, T., Drosos, V., Ugalde, J., & Chacko, J. (2018). Machine Learning Applications for Site Characterization Based on CPT Data. In *Geotechnical Earthquake Engineering and Soil Dynamics V: Slope Stability and Landslides, Laboratory Tesing and In Situ Testing* (pp. 461-472). USA: American Society of Civil Engineers.

Wang, H., Wang, X., Wellmann, J., & Liang, R. (2019). A Bayesian unsupervised learning approach for identifying soil stratification using cone penetration data. *Canadian Geotechnical Journal, 56*(8), 1184-1205.

Wang, Y., Huang, K., & Zijun, C. (2013). Probabilistic identification of underground soil stratification using cone penetration tests. *Canadian Geotechnical Journal , 50*(7), 766-776.

Waskom, M. (2020). *Seaborn*. Retrieved October 2020, from https://seaborn.pydata.org/index.html

Wu, C., Zhong, H., Zhang, W., Li, Y., & Wang, L. (2020). Prediction of undrained shear strength using extreme gradient boosting and random forest based on Bayesian optimization. *Geoscience Frontiers*.

Yogatama, B. A. (2018). *geotech40.blogspot.com*. Retrieved January 2021, from https://geotech40.blogspot.com/2018/12/interpretation-of-soil-type-using-python.html

Zhang , W., Wang, L., & Wu, C. (2019). Assessment of pile driveability using random forest regression and multivariate adaptive regression splines. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*.

Zou, Y., & Boley, C. (2019). Eigenschaften und Klassifikation von Böden. In C. Boley (Ed.), *Handbuch Geotechnik Grundlagen - Anwendungen - Praxiserfahrung* (pp. 13-56). Wiesbaden: Springer Vieweg.

11 Appendix

11.1 Soil classification

11.1.1 Oberhollenzer_classes

Group 1:	Group 2:	Group 3:	Group 4:	Group 5:	Group 6:	Group 7:	Ignored classifications
Gr,sa,si' → Gr,co	**Or,cl → Or,sa'**	**Or,sa → Or/Sa**	**Sa,gr,si → Gr,sa,si**	**Sa,si → Sa,gr,si'**	**Si,sa,cl' → Si,sa,gr**	**Cl/Si,sa' → Si,cl,sa**	
Gr,Sa	Si,or	Sa,Or	Gr,Si	FSa,msa,fgr	Sa,Si,gr-	Cl,Si,fsa	Si,cl,gr,co
Gr,Sa,si-	Si,or-	Sa,or	Gr,sa+,si	CSa	Si,gr,sa	Cl,si,gr-	FGr,MGr,MSa,CSa,fsa,si-
Gr,sa	Or	Gr,sa,si,or	Gr,sa,si	FSa	Si,gr	Si,Cl,FSa	K,sa
Gr,sa+	Or,Si	FSa,si,or	Gr,sa,si+	FSa, msa, csa	Si,gr-,fsa	Si,Cl,gr	Mgr,fsa,cl,gr
Gr,sa+,si-	Or,si	Si,fsa,or	FGr,MGr, cgr-,si+	FSa,MSa	Si,gr,sa,co	Cl,si,fsa	Predrill
Gr,sa,si-	Or,si-	FSa,Si,or	FGr,MGr, si	FSa,msa	Si,gr,sa,cl	Cl,si,sa	MGr,cgr,co
Sa,Gr, co-	Or,si,cl	Si,cl,sa,or	MGr,CGr, si	FSa,cl-	Si,sa+,gr+	Cl,Sa	MGr,cgr,fgr
FGr	Or,Si,sa-	Or,FSa	FGr,csa	MSa,FSa	Si,sa,gr	Cl,fgr-	FGr,MGr,Sa,cgr,si,co
FGr,CGr,sa+	Si,cl,or		Gr,si,sa+	MSa,FSa, si--	Si,sa,gr-	Cl,fsa	M
FGr,MGr	Or,sa-		Gr,si,sa+, co	Sa	Si,FGr,sa+,cl-	Cl,gr-	A,fgr,mgr,sa
FGr,MGr, sa+			CGr,si	FSa,msa,csa,si-	Si,gr+,sa-,cl	Cl,mgr	Tm
FGr,sa+			CGr,co,si ++,sa++	Sa,fgr,si	Si,fsa,fgr,msa,csa	Cl,si-,fsa	Gr,sa,co,si
CGr,FGr, CSa,FSa			Gr,Cl,Sa	Sa,fgr-	Si,sa,cgr++	Cl,si+,sa	CGr,CSa
MGr			Gr,sa,co-,si	Sa,fsa,si	Si,cl,sa,gr-	Cl,fsa-	Gr,sa,co,bo
MGr,CSa			Sa,Gr,Co, si	Sa,si-,fgr-	Si,gr+	Si	Gr,co,sa,bo-
MGr,sa			Gr,si,sa	FGr,Sa,si	FSa,Si,MSa	Si,grcl	FGr,FSa,si
MGr,sa+			Gr,si,sa-,co-	FSa,fgr-	FSa,CSi	Si,cl	Si,Cl,Sa,Gr,co,bo

MGr,sa-			Gr,si,co-	FSa,CSa	FSa,MSa,msi+,cl	Cl,Si	FSa,gr,si
Sa,FGr			Gr,si,sa,co-	FSa,CSa,si	Si,Gr,sa	Cl,sa-,si	A,sa,si+
Si,Sa,Gr			Gr,si,sa-	FSa,MSa,si+,csa	Si,Sa	Cl,si	FSa,MSa,CSa
FSa,MGr,gr			Sa,Co,Gr,si	FSa,sa,si-,fgr-	Si,FSa	Si,Cl	A,gr,sa+,si
Gr,Co			Gr,Sa,Si	CSa,fgr,si--	Si,FSa,MSa	Si,co	Tst,si
CGr,FGr,CSa,FSa,si			FGr,Sa	CSa,si--	Si,fsa	Cl	Mgr,gr,sa
A,gr,sa			MGr,CGr,si+,sa	MSa,csa+	Si,fsa,fsa+	Si,cl-	Si,Cl,Sa,Gr,co
FGr,CSa,si,gr			Sa,FGr,MGr	Sa,fgr,mgr,cgr-	Si,sa	Si,cl	A
A,sa,si			Sa,gr	FGr,si	Si,sa,cl-	Si,cl+	Gst
Gr,co,sa-			Sa,gr,co	Sa,si,gr-	Si,sa-		A,Gr
Gr,sa,co-			Sa,fgr,mgr	Sa,si,cl-	FSa,Si		MGr,cgr,fgr,co
A,sa,mgr,fgr,cgr			Sa,gr,si	CSa,MSa	FSa,Si,cl-		CSa,Gr,Co
Co,Gr,A			Sa,gr,si-	CSa,si-	Si,fsa-		A,Gr,Sa,co,si-
CGr,MGr			FSa,FGr	CSa, fsa	Si,fsa-,fsa		A,Gr,sa,si-
A,msa,mgr,cgr,co			FGr,FSa	FSa,csa	Si,fsa+		A,Gr,sa,si
Gr,sa,co			FGr,MGr,Sa	Sa,si	Si,fsa+,sa		Br,Si,fsa
MGr,CGr,sa			CSa,fgr+	Sa,si+,gr-	Si,fsa,gr-		A,Gr,co,sa,si
A,gr,sa,si-			Gr,csa+,msa-	Sa,si-	Si,msa,csa-		Gr
A,Gr,sa			FGr,csa,si	FSa,csi	Si,sa,gr--		Gr,si
CGr,co			CSa,gr,co,sa,si	FSa,MSa,si+	Si,sa+,gr-		Sa,Si,gr
A,Gr,sa+,si-			MGr,sa+,si	FSa,MSa,si-	FSa,cl,si		Sa,si,fgr
A,gr,sa,si			FSa,gr	FSa,MSa,si-,si	Si,CSa		Co,sa,si
A,fsa,csa,fgr			FSa,MSa,si, fgr+,mgr+	FSa,sa,si-	Si,sa-,gr--		Z
Gr,co,fsa,si			CSa,FSa,gr	FSa,si	FSa,si,cl		CSa,fgr,si

A,sa			MSa,CSa,si-,mgr-,cgr-	FSa,si+	CSi,fsa,cl-		MGr,FGr,sa,si-
MGr,CGr,si,si+			MSa,mgr,cgr	FSa,si+,cl-	Si,fsa-,cl		A,sa,si-
FSa,cgr,csi			MSa,FSa,fgr++,csa+	FSa,si-	Si,fsa-,cl-		Gr,co
Bo			Sa,si,gr	FSa,si-,msa-	Si,cl,fgr-		Grn
A,Gr,fsa+,csa+			MGr,si	FSa,si-,si	Si,cl,fsa		Kst
CGr,sa			CSa,FSa,si,gr,cl	MSa, si	Si,cl,fsa-		Sa,Si
MGr,cgr			CSa,si	FSa,MSa,si	Si,cl,sa-		Sa,Si,fgr
A,sa,gr,co			FSa,MSa,si-,gr	FSa,sa,si	Si,fsa,cl		Tst
Gr,co,sa			Sa,fgr,si-	FSa,sa-,si	Si,FSa,cl		Si,cl,gr
MGr,cgr,sa			Sa,fsa,si-,fgr-,mgr-	FSa,cl	Si,fsa,cl-		Co,Bo
A,Gr,Sa			FSa,msa,fgr-	FSa,si,msa	Si,FSa,MSa,Cl		FGr,mgr,sa
FGr,CGr,fsa			MSa,CSa,fgr-,si-		Si,cl,sa		FSa,si,co-
FGr,MGr,csa			Sa,gr-		Si,Sa,gr,cl		MGr,fgr,cgr
FGr,MGr,sa			Si,fsa,csa,fgr-,co-		Si,cl+,sa		Cl,Gr
FGr,MGr,sa+,si-			Gr,fsa,si		Si,gr,cl-		Phy
MGr,FGr,sa+			FSa,si,gr,co		Si,cl+,fsa-		Sa,mgr-
MGr,FGr,si-			FSa,si,gr--		Si,FSa,cl--		Cl,gr,si
MGr,cgr-,fgr+,sa			GSa,MSa,si,gr-				Gr,si,cl
Gr,Sa,si			Sa,gr-,si-				
Gr,sa+,co			Sa,gr+,si-				
Gr,sa,cl			CSa,fgr,fsa				
Gr,sa,cl-			FSa,si,gr,cl-				
Sa,gr+,co			Sa,fgr				
MGr,CGr							

MGr,CGr,cl+							
Si,Sa,gr							
Co,sa							
Gr,co,si-							
Co							
Gr,sa,si-,co							
CGr							
MGr,co							
A,Gr,sa,si-,co							
Gr,co+,sa,si-							
Gr,sa,co,si-							
MGr,csa,fgr							
Gr,Sa,co							
Gr,co,si-,sa-							
Sa,Gr							
Gr,sa+,co,si-							
Sa,Gr,si-							
Gr,Sa,co-							
Gr,si-,sa-							
CGr,MGr,co							