Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Materials Science and Engineering

Mats Stensrud Skui

Machine Learning for Predicting Polarization Curves and Their Features

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Andreas Erbe
Co-supervisor: Iman Taji
June 2023

NTNU
Norwegian University of
Science and Technology

Mats Stensrud Skui

Machine Learning for Predicting Polarization Curves and Their Features

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Andreas Erbe
Co-supervisor: Iman Taji
June 2023

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Materials Science and Engineering

Preface

This work was conducted as part of the author's master's thesis at the Department of Materials Science and Engineering, Norwegian University of Science and Technology (NTNU) in Trondheim, during the spring of 2023. I would like to thank my supervisor Andreas Erbe, co-supervisor Iman Taji, and laboratory assistants Anita Storsve and Marthe Folstad for their valuable assistance throughout this project. Additional thanks goes to Norsk Hydro ASA for providing samples for testing.

Finally, I would like to thank my good friends and fellow students, especially the "Mattek gang", for making these years so great.

Abstract

Typically, important electrochemical parameters are derived from empirical data. This is largely due to the fact that analytical models, such as various forms of the Butler-Volmer equation, often fall short in accurately capturing the important features. Unfortunately, obtaining these polarization curves requires utilization of time consuming instruments. Therefore, employing machine learning could prove advantageous. In this work, machine learning was applied in the attempt of predicting important polarization curves and some of their important features in various pH environments on an aluminium alloy AA6060 + 0.0043 wt% nickel. The polarization curve features under investigation were (i) the corrosion potential, (ii) the cathodic Tafel slope, (iii) the corrosion current density and (iv) the critical pitting potential.

Determining the most suitable machine learning algorithm for a specific task can be challenging. Therefore, in this study, five distinct algorithms were employed. These included four decision tree algorithms, namely Random Forest, Categorical Boosting, eXtreme Gradient Boosting, and Light Gradient Boosting Machine, along with an Artificial Neural Network. The results delivered by these algorithms were comparable, though the Random Forest algorithm demonstrated a slight edge in terms of both accuracy and training speed. Random Forest's marginally enhanced predictive abilities were explained by the small input feature dimension of solely pH and electrochemical potential. This lead to large amounts of unknown behaviour in the training data set as variables such as surface cracks and segregation were not accounted for. As a result, the error observed on unseen data was considerably larger, reaching many multiples of the error observed on the training data. The large variation in data lead to a smaller effect of the applied hyperparameter tuning of certain models. Thus, viability of using an Artificial Neural Network for the given machine learning task came under scrutiny, given its need for extensive hyperparameter tuning, which, despite the effort, only yielded average results compared to the other algorithms.

Finally, the algorithms demonstrated intriguing results, achieving a mean absolute percentage error of roughly 7% for the logarithmic current density (the target output) over the applied potential range. Furthermore, the algorithms showed strong predictive abilities for the features under investigation, with average errors of 5%, 13% and 19% for the corrosion potential, cathodic Tafel slope and corrosion current density, respectively. Additionally, the algorithms located the pitting potential with perfect precision. Moreover, the algorithms proved high capability of capturing key transitions in the polarization curves, such as the occurrence of meta-stable pitting at certain pH regions. Therefore, the results underscore the potential of these algorithms to effectively replace time-consuming instruments.

Sammendrag

Vanligvis er viktige elektrokjemiske parametere hentet fra empiriske data. Dette skyldes i stor grad at analytiske modeller, som ulike former av Butler-Volmer-ligningen, ofte kommer til kort i nøyaktig å fange de viktige egenskapene. Dessverre krever innhenting av disse polarisasjonskurvene bruk av tidkrevende instrumenter. Derfor kan bruk av maskinlæring være fordelaktig. I dette arbeidet ble maskinlæring brukt i forsøket på å forutsi viktige polarisasjonskurver og noen av deres viktige egenskaper i forskjellige pH-miljøer på en aluminiumslegering AA6060 + 0.0043 vektprosent nikkel. Polariseringskurveegenskapene som ble undersøkt var (i) korrosjonspotensialet, (ii) den katodiske Tafel-hellingen, (iii) korrosjonsstrømtettheten og (iv) det kritiske potensialet for gropkorrosjon.

Å bestemme den mest passende maskinlæringsalgoritmen for en spesifikk oppgave kan være utfordrende. Derfor ble fem forskjellige algoritmer brukt i dette arbeidet. Disse inkluderte fire beslutningstrealgoritmer, nemlig Random Forest, Categorical Boosting, eXtreme Gradient Boosting, og Light Gradient Boosting Machine, sammen med et kunstig nevralnettverk. Resultatene som disse algoritmene leverte var sammenlignbare, selv om Random Forest-algoritmen viste en liten fordel i både nøyaktighet og treningshastighet. Random Forests marginalt forbedrede prediktive evner ble forklart ved at den lille dimensjonen til treningsdata kun var pH og elektrokjemisk potensial. Dette førte til store mengder ukjent oppførsel i treningsdatasettet da variabler som overflatesprekker og segregasjon ikke ble tatt hensyn til. Som et resultat var feilen observert på usett data betydelig større, og nådde mange ganger feilen observert på treningsdataene. Den store variasjonen i data førte til en mindre effekt av den anvendte hyperparameterjusteringen av visse modeller. Derfor ble det konkludert usikkert hvorvidt det kunstige nevralnettverket var passende for maskinlæringsoppgven under vurdering, gitt dets behov for omfattende hyperparameterjustering, som, til tross for innsatsen, kun ga gjennomsnittlige resultater sammenlignet med de andre algoritmene.

Til slutt viste algoritmene motiverende resultater, og oppnådde en gjennomsnittlig absolutt prosentvis feil på omtrent 7% for den logaritmiske strømtettheten (variablen som skulle predikere) over det anvendte potensialområdet. Videre viste algoritmene sterke prediktive evner for de undersøkte egenskapene, med gjennomsnittlige feil på 5%, 13% og 19% for korrosjonspotensialet, den katodiske Tafel-hellingen og korrosjonsstrømtettheten, respektivt. I tillegg kunne algoritmene identifisere potensialet for gropkorrosjon med perfekt presisjon. Videre viste algoritmene høy evne til å fange opp nøkkeloverganger i polarisasjonskurvene, som forekomsten av meta-stabil gropkorrosjon i visse pH-områder. Derfor understreker resultatene potensialet til disse algoritmene til effektivt å erstatte tidkrevende instrumenter.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Description	Unit
R	Universal gas constant	J mol^{-1}K^{-1}
n	Number of electrons transferred	-
F	Faraday's constant	C mol^{-1}
T	Temperature	K
E_{corr}	Corrosion potential	V
i_{corr}	Corrosion current density	A cm^{-2}
i_0	Exchange current density	A cm^{-2}
$i_{\text{lim,c}}$	Cathodic limiting current density	A cm^{-2}
$i_{\text{lim,a}}$	Anodic limiting current density	A cm^{-2}
i_{net}	Net current density	A cm^{-2}
b_{a}	Anodic tafel slope	mV/decade
b_{c}	Cathodic tafel slope	mV/decade
w	Weight in Artificial Neural Network	-
b	Bias in Artificial Neural Network	-
σ	Standard deviation	N/A
$\bar{\sigma}$	Standard error	N/A
z	Z-score in normal distribution	-

Symbol	Description	Unit
α	Charge transfer coefficient	-
η_{a}	Anodic overpotential	V
η_{c}	Cathodic overpotential	V
η_{act}	Activation overpotential	V
δ	Nernst diffusion layer thickness	µm

Abbreviations

ORR Oxygen Reduction Reaction

HER Hydrogen Evolution Reaction

WE Working Electrode

CE Counter Electrode

RE Reference Electrode

SCE Standard Calomel Electrode

OCP Open Circuit Potential

CV Cyclic Voltammetry

ML Machine Learning

DT Decision Tree

GBDT Gradient Boosted Decision Tree

RF Random Forest

CatBoost Categorical Boosting

XGBoost eXtreme Gradient Boosting

LightGBM Light Gradient Boosted Machine

ANN Artificial Neural Network

HP Hyperparameter

ReLU Rectified Linear unit

ADAM Adaptive Moment Estimation

IL Input Layer

HL Hidden Layer

OL Output Layer

MSE Mean Squared Error

RMSE Root Mean Squared Error

TE Training Error

VE Validation Error

RS Random Search

GS Grid Search

CPU	Central Processing Unit
GPU	Graphics Processing Unit
SMA	Simple Moving Average

1 Introduction

Aluminium is the third most abundant metal in Earth's crust and is extensively utilized for many applications due to its low density and ability to withstand corrosion in many environments[1]. However, on its own, aluminium presents challenges due to its low strength. Thus, aluminium is often alloyed with other metals. Unfortunately, this can simultaneously present electrochemical complications.

Aluminium and aluminium alloys are generally resistant to severe corrosion in neutral environments due to the immediate formation of a protective alumina layer on their surfaces. However, in solutions rich in chloride or of acidic or alkaline nature, this alumina layer can break down, leaving the aluminium or aluminium alloys unprotected and susceptible to corrosion. Understanding the rate of corrosion is crucial when selecting materials for various applications. For a comprehensive understanding of the ongoing surface reactions, polarization curves are extensively analyzed. Unfortunately, these polarization curves are derived from time-consuming empirical experiments. Therefore, the use of machine learning to generate these polarization curves and subsequently discern significant features, trends, and transitions is highly desirable.

As technology advances, machine learning has seen significant development in recent years. However, choosing the appropriate model can be a complex task, given the unique nature of every data set. In this work, several prominent machine learning algorithms from recent years will be utilized to predict the polarization curves of AA6060 + 0.0043 wt% nickel in varying pH environments. From these polarization curves, the algorithms will aim to estimate the corrosion potential, cathodic Tafel slope, corrosion current density, and pitting potential. Additionally, the same models will attempt to identify significant trends and transitions in the polarization curves, such as pinpointing regions where meta-stable pitting occurs.

2 Theory

2.1 Electrochemistry

2.1.1 Background and corrosion of aluminium and aluminium alloys

aluminium is the third most abundant metal in the Earth's crust, making up over 8% of its weight[1]. Aluminium is not found pure in nature, but usually in minerals such as bauxite and cryolite. Due to aluminium's versatile properties, it is widely utilized in e.g. packaging, cans, and even for components in cars. Aluminium is the second most used metal after iron, and on average, the global demand for aluminium per person is estimated to be 11 kg[1;2]. Compared to other metals, aluminium exhibits low density, approximately one third of the density of iron, making it attractive for use in a broad variety of applications.

An important characteristic of aluminium is its ability to resist corrosion under normal circumstances. Corrosion resistance is achieved by the rapid formation of an alumina film on the surface of aluminium, which provides protection against further oxidation. This film is stable under pHs ranging from approximately 5 to 8. Outside of this range, corrosion products besides aluminium oxide are formed[3]. This phenomenon is also demonstrated in the Pourbaix diagram of aluminium, as shown in Figure 2.1, where the reduction potential of aluminium is lower than that of the cathodic reactions involving the hydrogen evolution reaction (HER) and oxygen reduction reaction (ORR), implying that oxidation of aluminium is preferred over reduction of Al^{3+}.

Pure aluminium often does not possess sufficient mechanical properties for many applications, and therefore, aluminium is often alloyed with other metals to enhance its mechanical properties. There are two types of aluminium alloys, *wrought* and *cast* alloys. Wrought alloys are formed through mechanical methods such as rolling, forging, and extrusion, while cast alloys are directly cast into the final shape[1]. Wrought aluminium alloys exhibit several excellent properties, such as low density, high strength-to-weight ratio, and great corrosion resistance in various environments. However, like pure aluminium, aluminium alloys face challenges in acidic and alkaline media with chloride causing depassivation of the surface[4].

Localized corrosion is a great concern as intermetallics may act as cathodic sites relative to aluminium, resulting in corrosion of the latter. One such form of localized corrosion is pitting corrosion. Pitting corrosion takes four stages: (i) processes occuring on the passive film and the solution; (ii) reactions inside of the passive film where no visible microscopic alterations are observed in the film; (iii) formation of meta-stable pits which can grow for a brief duration below the pitting potential E_{pit}; and (iv) the growth of stable pits, which occurs above a specific potential referred to as the pitting potential E_{pit}[3]. An example of how E_{pit} may be visualized graphically is shown in Figure 2.2, detected by a rapid change in anodic slope.

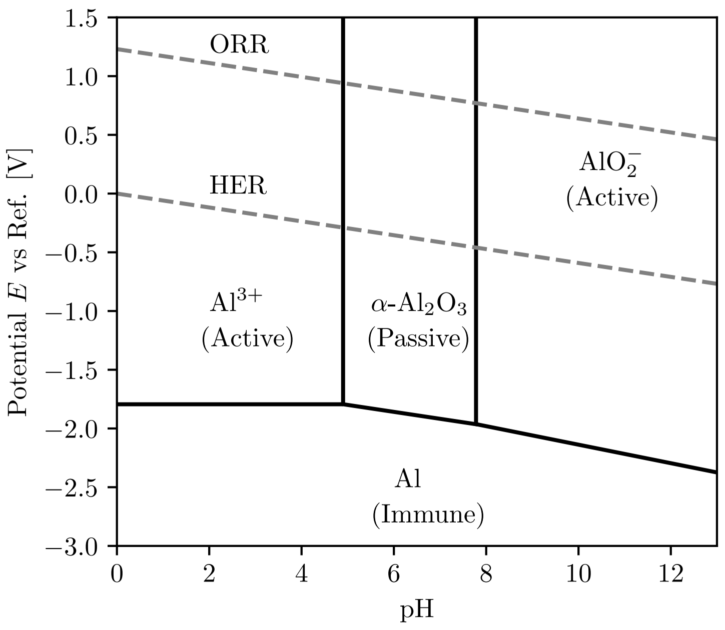


Figure 2.1: Pourbaix diagram of α-aluminium with passive, immune and active regions included. The applied activity of Al^{3+} is 10^{-6} mol L^{-1} [5]

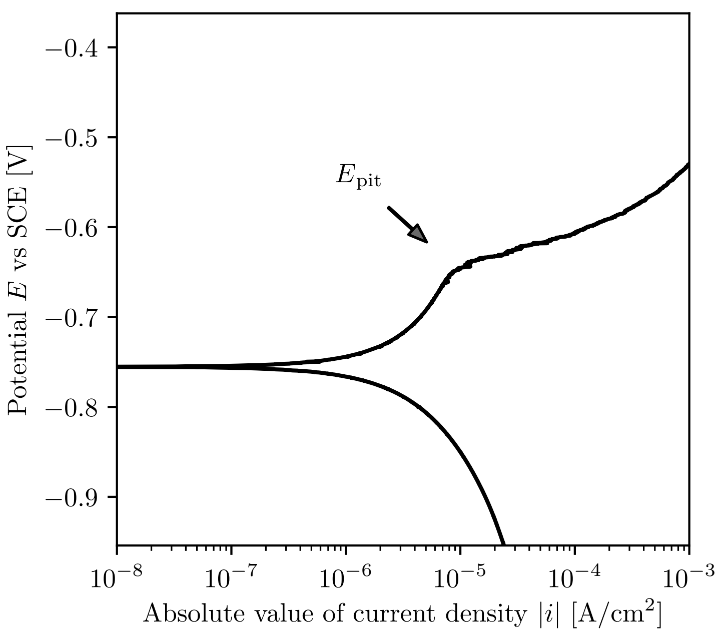


Figure 2.2: The pitting potential E_{pit} is seen by a sharp decline in the anodic Tafel slope

2.1.2 Electrochemical reactions

As depicted in the pourbaix diagram of aluminium, in acidic and alkaline media,aluminium undergoes oxidation through Eq. 2.1 (acidic) and Eq. 2.2:

$$\mathrm{Al} \rightleftharpoons \mathrm{Al}^{3+} + 3\,\mathrm{e}^-, \tag{2.1}$$

$$\mathrm{Al} + 2\,\mathrm{H_2O} \rightleftharpoons \mathrm{AlO_2^-} + 4\,\mathrm{H}^+ + 4\,\mathrm{e}^-. \tag{2.2}$$

Possible cathodic reactions except reduction of aluminium are Hydrogen Evolution Reaction (HER) and Oxygen Reduction Reaction (ORR). The reversible potential of ORR is higher for any given pH relative to HER and thus thermodynamically more likely than HER[6]. In reality, the ORR is complex and can involve multiple reactions and reaction steps[7]. Common net reactions are Eq. 2.3 and Eq. 2.4:

$$\mathrm{O_2(g,\ dissolved)} + 4\,\mathrm{H^+(aq)} + 4\,\mathrm{e}^- \rightleftharpoons 2\,\mathrm{H_2O\,(l)}. \tag{2.3}$$

$$\mathrm{O_2(g,\ dissolved)} + 2\,\mathrm{H_2O\,(l)} + 4\,\mathrm{e}^- \rightleftharpoons 4\,\mathrm{OH^-(aq)}. \tag{2.4}$$

Furthermore, HER takes two different forms in acidic and alkaline media, given as[6]:

$$2\,\mathrm{H}^+ + 2\,\mathrm{e}^- \rightleftharpoons \mathrm{H_2}, \tag{2.5}$$

$$2\,\mathrm{H_2O} + 2\,\mathrm{e}^- \rightleftharpoons \mathrm{H_2} + 2\,\mathrm{OH}^-. \tag{2.6}$$

2.1.3 Charge transfer controlled kinetics

Assuming charge transfer controlled kinetics, the net electrode current density i_{net} can be given by the simplest form of the Butler-Volmer (BV) equation[6]:

$$i_{\text{net}} = i_0 \cdot \left\{ \exp\left[\frac{\alpha_a n F}{RT}(\eta_{\text{act}})\right] - \exp\left[-\frac{\alpha_c n F}{RT}(\eta_{\text{act}})\right] \right\}, \tag{2.7}$$

where i_0 is the exchange current density, α is the charge transfer coefficient for the anodic (a) and cathodic (c) reaction, R is the universal gas constant , T is the temperature, and n_{act} is the activation overpotential.

Figure 2.3 depicts a polarization curve of a fictitious reaction under kinetics limited by charge transfer with equal charge transfer coefficient α for the anodic and cathodic reaction, resulting in symmetry around the reversible potential E_{rev}.

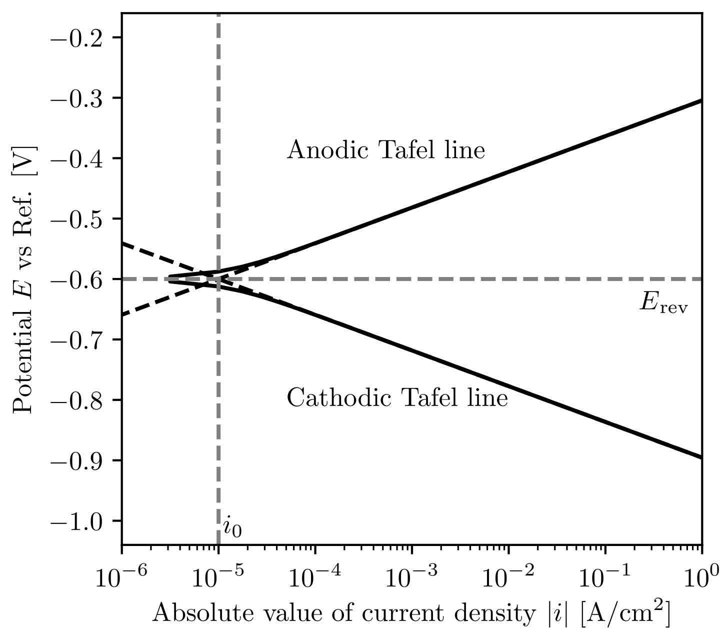


Figure 2.3: Polarization curve of a fictive reaction controlled by charge transfer kinetics. Both anodic and cathodic α is 0.5, making the cathodic and anodic part symmetrical. E_{rev} = -0.60 V, n = 2.[5]

2.1.4 Charge transfer and mass transport kinetics

Kinetics controlled solely by charge transfer is not realistic for most environments. The generalized BV equation accounts for both charge transfer and mass transport kinetics and is given by[6]:

$$i_{\mathrm{net}} = i_0 \left\{ \frac{c_{\mathrm{R,s}}}{c_{\mathrm{R,b}}} \exp\left[\frac{\alpha_{\mathrm{a}} n F \eta}{RT}\right] - \frac{c_{\mathrm{O,s}}}{c_{\mathrm{O,b}}} \exp\left[-\frac{\alpha_{\mathrm{c}} n F \eta}{RT}\right] \right\}, \tag{2.8}$$

where $c_{\mathrm{R,s}}$, $c_{\mathrm{R,b}}$, $c_{\mathrm{O,s}}$ and $c_{\mathrm{O,b}}$ represent the concentration of the reduced (R) and oxidized (O) species on the surface (s) and in the bulk (b). The surface concentrations of the reactive species are complex to derive and will depend on the cathodic reaction that occur[7]. Nevertheless, by assuming diffusion limited kinetics, the relation between the surface and bulk concentration of a species B can be given as[6]:

$$\frac{c_{\mathrm{B,s}}}{c_{\mathrm{B,b}}} = 1 - \frac{i}{i_{\mathrm{lim}}}. \tag{2.9}$$

where i_{lim} is the limiting current density, known as the maximum net current on an electrode due to limited transport of reactive species to the electrode surface. The limiting current density i_{lim} is graphically demonstrated in Figure 2.4 by the vertical asymptote at the cathodic branch, and is mathematically approximated as:

$$i_{\text{lim}} = -\frac{nFD_{\text{B}}c_{\text{B,b}}}{\delta}, \tag{2.10}$$

where D_{B} is the diffusion rate of the reactive species on the electrode surface, F is Faraday's constant, δ is the Nernst layer thickness, $c_{\text{B,b}}$ is the concentration in the bulk and n is the number of electrons transferred.

By utilizing Eq. 2.10 for both the anodic and cathodic reaction, Eq. 2.8 is transformed into[6]:

$$i_{\text{net}} = \frac{i_0 \exp\left[\frac{\alpha_a nF}{RT}\eta\right]}{1 + \frac{i_0}{i_{\text{lim, a}}} \exp\left[\frac{\alpha_a nF}{RT}\eta\right]} - \frac{i_0 \exp\left[\frac{-\alpha_c nF}{RT}\eta\right]}{1 + \frac{i_0}{i_{\text{lim, c}}} \exp\left[\frac{-\alpha_c nF}{RT}\eta\right]}. \tag{2.11}$$

The net current density following Eq. 2.11 is depicted in Figure 2.4. From Eq. 2.8 it is noted that for large negative overpotentials, the cathodic term dominates relative to the anodic current contribution. When the concentration at the surface $c_{\text{O,s}}$ becomes infinitely small, the net current density i_{net} converges to zero for any change in overpotential. Thus, the diffusion limited current density $i_{\text{lim, c}}$ is obtained. Similarly, for large positive η, the anodic limiting current density $i_{\text{lim, c}}$ is attained.

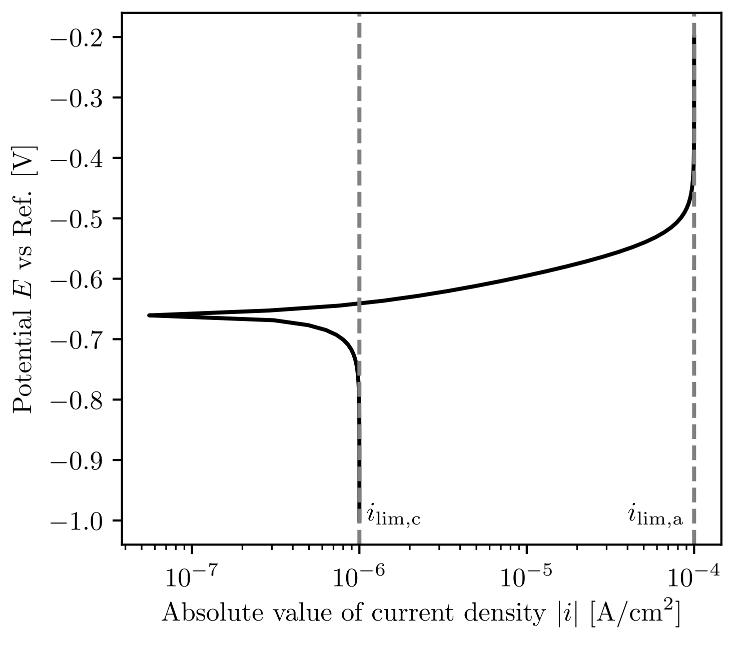


Figure 2.4: Polarization curve with diffusion controlled kinetics. The limiting current densities are observed by the vertical asymptotes.[5]

Eq. 2.8 does not elaborate the surface concentration of oxidizing and reducing agents for kinetics deviating from the first order. The kinetics behind this is complex and will not be derived. However, T. Shinagawa et al. (2015)[7] performed a comprehensive study investigating the higher order kinetics for reactions like HER and ORR. Their results supported that the BV equation (Eq. 2.8) in most cases fails to accurately describe the kinetics on an electrode surface as that requires a series of assumptions regarding the surface coverage and rate determining steps, neglecting higher order kinetics. However, BV is often a suitable first approximation.

2.1.5 Tafel slopes

The Tafel equation applicable under charge transfer controlled kinetics simplifies the BV equation (Eq. 2.7) for large overpotentials ($|\eta| > 0.1$ V) and thus outlines the linear parts of the polarization curve. By solving for the activation overpotential, η_{act}, the net overpotential, $\eta_{\mathrm{act,net}}$, can be obtained with b_{c} and b_{a} being the cathodic and anodic Tafel slope, respectively[6]:

$$\begin{aligned}\eta_{\mathrm{act,net}} &= \eta_{\mathrm{act,c}} + \eta_{\mathrm{act,a}} \\ &\approx \frac{-\ln(10)RT}{\alpha_c nF}\log_{10}\left(\frac{|i_c|}{i_0}\right) + \frac{\ln(10)RT}{\alpha_a nF}\log_{10}\left(\frac{i_a}{i_0}\right),\end{aligned} \tag{2.12}$$

with b_{c} and b_{a} representing the cathodic and anodic Tafel slope, respectively. A more general expression of the potential E applicable for any case of linearity no matter the rate determining reactions is given as:

$$E = \frac{\partial E}{\partial \mathrm{log}_{10}(|i|)}\mathrm{log}_{10}(|i|) + \mathrm{B}, \tag{2.13}$$

where $\frac{\partial E}{\partial \mathrm{log}_{10}(|i|)}$ is the Tafel slope of either the anodic or cathodic part of the polarization curve, and where B represents the intercept on the axis of E. A clear Tafel region is defined when the slope is the same for over at least one decade, as multiple reactions may occur at various potentials[8].

As addressed in Section 2.1.4, neglecting mass transfer kinetics is incorrect in most electrochemical environments. Therefor, incorporating the concentration of species on the electrode surface when expressing the Tafel slope yields a more accurate description of the kinetics rather than solely through charge transfer controlled kinetics. Determining the reaction rate on the electrode surface is however a very complex task, which is why Tafel slopes often are estimated empirically[7].

As depicted in Figure 2.4, interpreting a valid Tafel region can be complicated due to rapid change in slope caused by diffusion limitations. The cathodic Tafel slope constantly becomes more negative when approaching the limiting current as the oxygen deposits on the surface become infinitely low.

2.1.6 Influence of pH on electrochemical properties

Results from: (i) B. Zaid et al. on their study of AA6061 in various chloride rich acidic, neutral and alkaline electrolytes; (ii) B. Tilak and C. Chen on their study of HER in alkaline media and (iii) M. F. Li et al. and their investigation of the ORR in alkaline media are applied to provide insight on how pH affects important corrosion properties[4;9;10]. Data is susceptible to variations and should not be treated deterministically, but work as plausible outcomes.

- **Corrosion potential, $E_{\mathbf{corr}}$**: In both acidic (pH = 2) and neutral (pH = 6) solutions, it is anticipated that the corrosion potential will be relatively similar, approximately -0.75 V vs SCE, and relatively unaffected by chloride content for a concentration $\geq$ 0.1 mol/L, suggested by B. Zaid et al.[4]. E_{corr} is not expected to change substantially during polarization, as the measured corrosion potential is similar to the E_{corr} measured during as a function of time. Contrary, E_{corr} is expected to decrease in a significant manner in strong alkaline media (pH = 12) to $\approx$ -1.4 V vs SCE, either due to thinning of the surface caused by hydroxide ion attack or due to complete absence of the surface film. In less alkaline environments, E_{corr} could reach -0.7 V vs SCE.

- **ORR Tafel slopes, $b_{\mathbf{c}}$**: The ORR Tafel slopes are expected to vary between -60 and -120 mV/decade depending on the rate determining reactions[10].

- **Corrosion current density, $i_{\mathbf{corr}}$**: The study by B. Zaid et al.[4] demonstrated higher cathodic current densities outside of the passive region of aluminium. In a 3.5 wt% ($\approx$ 0.6 mol/L) solution, the cathodic current density for pH = 6 varied between 10^{-6} to 10^{-5} A/cm^2. A pH of 2 implied almost 10 times higher cathodic current density, while for pH = 12, the cathodic current density could reach as high as 10^{-2} A/cm^2. This also resulted in a much higher corrosion current density i_{corr}, where the weight loss of aluminium was approximately five times higher for pH = 12 compared to pH = 2 in a 2.5 wt% ($\approx$ 0.43 mol/L) chloride solution. Further, the weight loss in the pH 6 solution was insignificant in comparison, making up less than a third of the weight loss as for a solution of pH = 2.

- **Pitting potential, $E_{\mathbf{pit}}$**: B. Zaid et al.[4] found that the pitting potential E_{pit} is expected to be unaffected by the change in pH[4]. In a $\approx$ 0.6 mol/L solution, E_{pit} was approximately -0.75 V vs SCE and expected to decrease with an increase in chloride content.

2.2 Machine Learning

Machine learning (ML) and artificial intelligence are widely applied for solving real time problems in fields like health care, finance, industry, military and scientific fields such as chemistry problems[11]. A category within ML is *supervised learning*, a learning method attempting to imitate human intelligence by training on labeled data[12]. In ML, the amount and quality of the data is the key to success. However, a developer can increase the chances of succeeding by applying a suitable model with correct architecture. In this section, five algorithms that have gained traction during recent years is discussed. These consist of four decision tree (DT) algorithms and an Artificial Neural Network (ANN).

The four DT algorithms under examination are Random Forest (RF), eXtreme Gradient Boosting (XGBoost), Light Gradient Boosted Machine (LightGBM), and Categorical Boosting (CatBoost). RF, the oldest of these algorithms, was first developed in 2001[13]. The remaining DT algorithms XGBoost, LightGBM, and CatBoost, were introduced in 2014[14], 2017[15], and 2017[16], respectively. All four DT algorithms are ensembles of DTs, serving as weak learners. The key difference between them is that while RF builds independent trees, the other three utilize the residual between the predicted and empirical value to construct the next tree in the sequence. Consequently, the latter three algorithms are called Gradient Boosted Decision Trees (GBDTs). GBDTs have the ability to outperform RF for a few reasons discussed in section 2.2.1.

The invention of Artificial Neural Networks (ANNs) dates back to 1952 when Rosenblatt introduced a perceptron element with a single neuron to solve linear problems[17]. An ANN is unique in its architecture, but shares certain similarities with DTs. In the following sections, the fundamental aspects of the four DT algorithms and ANNs are elaborated. To begin with, a general introduction to DTs and GBDTs is provided, followed by an overview of common techniques used to enhance machine learning performance.

2.2.1 Decision trees and gradient boosting

Decision provide predictions by recursively splitting initially randomly sampled non-normalized data from the total available training data into two or more nodes[18]. In this process, data can be selected less or more than once. This allows each tree to be trained on a unique subset of data and variables, which can be a great technique to capture general trends in the training data. This does howver also face challenges, as randomly drawing samples can make less significant data influence the learning process in an unwanted manner. On the other hand, important data can be picked more frequently. Various algorithms have methods to tackle this particular issue of random sampling by continuing learning on those samples providing highest negative error gradient and filter out less significant data, which will be explained further under each algorithm's section.

Figure 2.5 depicts an example of a simple decision tree with three decision nodes and four leaf nodes. The parameter x represents the sampled variable, whereas y denotes the predicted output. Assuming this was real training, it was determined that splitting data based on $x < 1$, $x > 3$, and $x > 2$ yielded the lowest prediction error. When splitting nodes, the decision tree identifies the optimal split to obtain the most information gain and minimize the output error[18]. Decision trees contain several nodes with different tasks. Each decision tree includes a root node, which consists of one or more samples from the original data set. The two other types of nodes are decision and leaf nodes[19]. Decision nodes divide the sampled data into two, while leaf nodes are terminal nodes that do not divide the data further, but provide the final prediction. In regression analysis, the outputs are numerical values. In the situation of multiple outputs in the terminal node, the average is returned as long as the expected shape of the output is a single value.

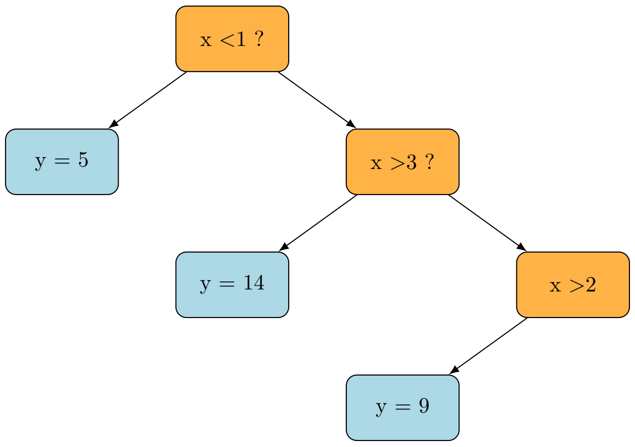


Figure 2.5: Regression tree illustration for a ficticious data set. Blue and orange corresponds to leaf/decision and terminal nodes, respectively.

Decision trees are widely utilized in machine learning due to their simple architecture[19]. Multiple DTs are often combined into ensembles to obtain better performance. One such ensemble algorithm is RF. RF utilizes multiple decision trees and returns the mean across all trees to obtain better generalization abilities compared to using solely one DT[13], and will be more comprehensively discussed in Section 2.2.3. On the other hand, GBDTs are constructed on a different basis, where the idea is to sequentially add more trees to the ensemble to train on the error of the whole ensemble so far[18]. By directly optimizing the next tree in the ensemble to maximize the negative error gradient, GBDTs can outperform RF, but risk adapting too well to training data, a term called *overfitting*, which is discussed in Section 2.2.2. Conversely, utilizing an insufficient number of iterations can induce *underfitting*, which occurs when the model has not reached its maximum potential to adapt to the training and validation data.

The main difference between the various GBDT algorithms is the employed tree and tree growing structure. While decision trees with LightGBM and CatBoost are by default built in a leaf-by-leaf fashion, the trees in XGBoost are built level-by-level by default. The difference between them is depicted in Figure 2.6, where the blue node represents the next node to build on. In the leaf-by-leaf tree growing structure, the tree is further developed from the node with the maxmimum error gradient[20;21]. Leaf-by-leaf allows for greater computational efficiency as well as being capabable of adapting greater to data, but are subsequently more susceptible to overfitting. On the other hand, level-by-level growth decreases the chance of overfitting as more regularization are applied by limiting the tree depth.

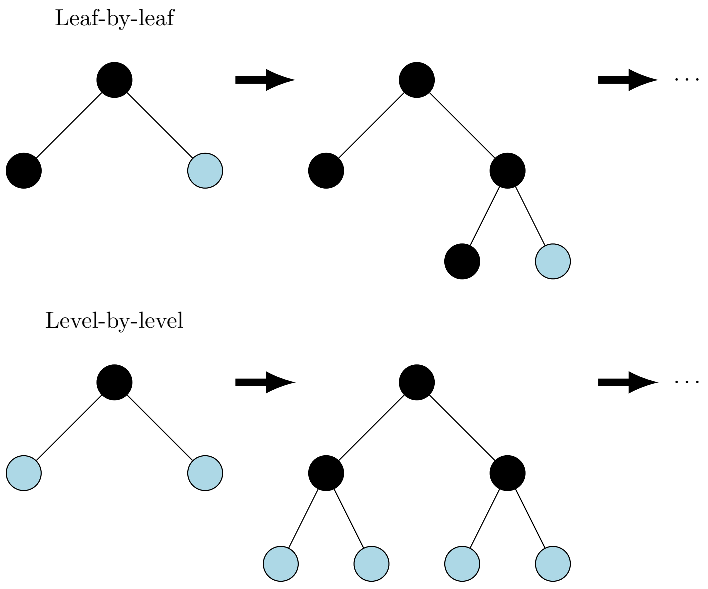


Figure 2.6: Leaf-by-leaf compared to level-by-level tree growing structure. The blue node represents the node to build on at the next iteration. While decision trees with LightGBM are built leaf-by-leaf, the trees in XGBoost are built level-by-level. Leaf-by-leaf allows for greater computational efficiency as well as being capable of adapting greater to data. On the other hand, level-by-level growth decreases the chance of overfitting as more regularization are applied by limiting the tree depth[20;21].

GBDTs have shown to be highly successful for a large variety of machine learning tasks, being highly adjustable to the specific problem[18]. A great strength towards RF is the plausibility to stop training early if no improvement is observed over a certain iteration window, and thus avoid overfitting. This phenomenon is called *early stopping* and is one of many widely utilized regularization techniques for iterative algorithms, included GBDTs and ANN. Another regularization technique to reduce likelihood of overfitting is the implementation of a penalty term added to the objective function, e.g. Mean Squared Error (MSE), Root Mean Squared Error (RMSE), to lower the *learning rate*, and thus improve the chance of finding the lowest loss with the drawback of increased training time. The learning rate determines how fast the model adapts to new information during training[22]. Commonly applied regularization techniques are *L1* and *L2* regularization where penalty terms to the loss function. In L2, extreme values are punished as the added penalty term is proportional to its magnitude, while in L1 sparse data (0 or no value at all) are favored as all other objectives are penalized equally[23]. Thus, applying both L1 and L2 in unison is a popular strategy to prevent overfitting. Contrary, utilizing L1 and L2 incorrectly by penalizing excessively can lead to high training time and lower likelihood of convergence, inducing underfitting. In combination with early stopping, this could indeed enhance training time to a significant degree.

2.2.2 Overfitting and underfitting

Overfitting and underfitting are unwanted phenomena that apply for all iterative machine learning algorithms[24;18]. Overfitting is recognized by a model that adapts well to training data, but generalizes poorly on unseen data. Contrary, underfitting is the opposite, where the algorithm has yet to adapt to training data, seen by a declining training and validation error. The algorithm learns from the training data, while a fraction of the data is delegated to work as validation data, which is unseen data the model is evaluated against to measure the performance. As discussed, a commnon way to tackle both underfitting and overfitting is through utilizing regularization techniques such as *early stopping* and added penalty terms to the loss function, in combination with a high number of iterations to avoid underfitting. These regularization techniques should be applied on the validation data[24]. In case of very noisy data, there is little to do to increase accuracy[24].

In Figure 2.7, a typical learning curve for an iterative machine learning algorithm is shown. Prior to optimal fit, the algorithm underfits to training data, which is observed by a declining validation error. When the optimal fit is reached, the validation error starts to increase and overfitting territory is reached. A plausible outcome of underfit, overfit and good fit for a regression problem is depicted in Figure 2.8a, 2.8b and 2.8b, respectively.

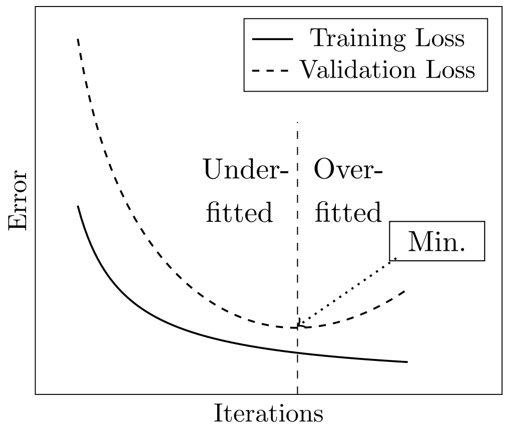
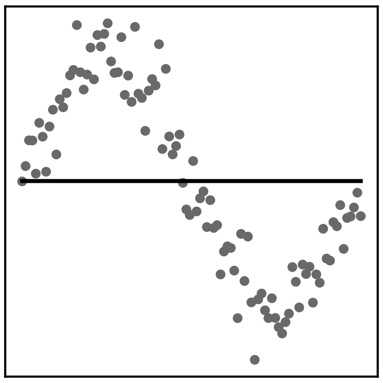
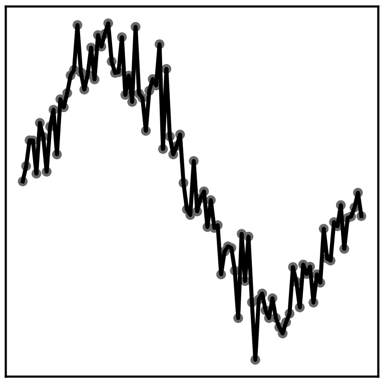


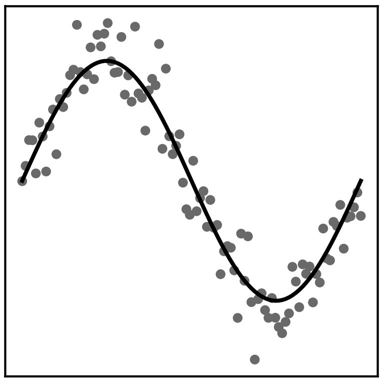
Figure 2.7: A quantitative illustration of underfitting and overfitting



(a) Underfit



(b) Overfit



(c) Good fit

Figure 2.8: Underfit, overfit, good fit

2.2.3 Random Forest

As discussed in Section 2.2.1, Random Forest (RF) is an ensemble learning algorithm with decision trees with random structures from pre-defined possible architectures as base learners. The combination of multiple decision trees enables the algorithm to achieve enhanced generalization performance by reducing the impact of individual decision trees' errors on the final prediction[13]. Contrary to GBDT algorithms, node splits are done by considering a random subset of features which drastically reduces likelihood of overfitting to training data. The error is expected to decrease rapidly by adding more trees for small number of trees, but tends to converge if many trees are already added.

The RF algorithm has become increasingly popular due to its success in addressing both classification and regression problems and its ability to handle noisy data[25;13]. It is efficient for both small and large-scale problems while maintaining statistical efficiency, and its architecture is easily adaptable. Furthermore, RF can handle a wide range of problems and has a simple architecture, resulting in fewer parameters to tune. Another strength of the algorithm is its ability to handle small datasets with high-dimensional feature spaces. In addition, it is well-suited to handle missing values. Another great feature about RF is that it is less susceptible to overfitting compared to iterative algorithms, as the variance between the trees grow for a large number of trees[25]. Training time does however become larger for a higher number of trees in the forest.

Despite the widespread use of RF, the algorithm has certain limitations. One of the major drawbacks is that each decision tree is learned independently, preventing the exploration of additional information across trees[26]. The approach of using independent trees may result in decent performance for many tasks, but it may not achieve top-tier accuracy like a GBDT algorithm can.

Table 2.1 provides the key hyperparameters of RF algorithm together with their respective definitions and default values obtained from `RandomForestRegressor` class in Scikit-learn[27]. A hyperparameter is defined as a parameter the developer has to assign prior to training[24]. Regarding hyperparameter tuning, empirical studies have indicated that pursuing enhances performance through hyperparameter tuning for RF may not be worth the time spent doing so[28]. However, scikit-learn `RandomForestRegressor` developers suggest that testing various values of number of trees (*n_estimators*) and maximum number of variables to consider for each split (*max_features*) should enhance the performance the most[29]. Additionally, applying many trees in the forest may be necessary to reliably determine feature (input variable) importance[30].

Table 2.1: Important hyperparameters in RF[30]. Hyperparameters are collected from the `RandomForestRegressor` class in scikit-learn[27], where description, importance and default values are also gathered.

Parameter	Description	Importance	Default
n_estimators	Number of trees in the forest.	High values increase training time and likelihood of overfitting. Opposite for small forests: low training time, but more likely to underfit.	100
max_features	Maximum number of features to consider when looking for the best node split.	Low values result in unique and less correlated trees, lower variance, but higher bias, and thus enhanced probability of underfitting to data. Few features also decrease training time.	1

2.2.4 eXtreme Gradient Boosting

eXtreme Gradient Boosting (XGBoost) is a GBDT algorithm that by default builds decision trees in a level-by-level fashion[31]. It is one of the most widely utilized algorithms in this category and has been found to provide high performance with minimal hyperparameter tuning requirements. The algorithm was initially developed by T. Chen and C. Guestrin to address scalability and sparse data set issues[32].

As XGBoost is a GBDT algorithm, overfitting may pose a challenge if excessive iterations are applied. To address this issue, XGBoost uilizes regularization techniques to limit the rate of learning[32]. Additionally, XGBoost may not be the best choice for high-dimensional feature data sets, as it can be computationally inefficient in those scenarios. With the development of new machine learning algorithms such as LightGBM (Section 2.2.5) and CatBoost (Section 2.2.6), XGBoost faces competition regarding training time. On mutliple machine learning tasks, LightGBM and CatBoost have shown equal or better accuracy with lower training time, computational power, and memory usage[16;21].

Regarding hyperparameter tuning, XGBoost developers acknowledge that optimal hyperparameters depends on the specific task[33]. The developers have multiple recommendations with regards to training time, overfitting and underfitting. With high and low training and testing data accuracy, respectively, there is a great probability that the model has overfitted to training data. They suggest two methods to combat this problem, (i) control the model complexity by adjusting *max_depth*, *min_child_weight* and *gamma* and (ii) to add randomness through changing *subsample*, *colsample_bytree* and *eta*. If underfitting is observed by not seeing any convergence, adjusting the parameters in opposite direction as to when overfitting is observed, is a good strategy. Regarding training time, XGBoost developers recommend setting tree method *tree_method* to *hist* or *gpu_hist*. *hist* is a tree method similar to LightGBM's leaf-by-leaf growth. Gpu_hist applies hist with GPU implementation. The parameters mentioned above are elaborated in Table 2.2, along with default values in the Python `xgboost` library.

Table 2.2: Important hyperparameters in XGBoost[34]. Default values are collected from XGBoost documentation[35].

Parameter	**Description**	**Importance**	**Default**
eta	Controls how fast the model learn by adding penalty term to the loss function.	Increase eta to prevent overfitting. However, a small laerning rate could enhance accuracy. Higher eta results in larger training time. Adjust iterations in the forest (num_round) reversely when changing eta.	0.3
max_depth	The maximum number of node levels in the tree	Enhanced depth will make the model more likely to overfit. Contrary, a very low depthmay induce overfitting	6
min_child_weight	A treshold value that determines whether or not a node should be split further. 1 implies purity.	Low value may induce overfitting as it the model could fit too well to data. Contrary, a high value increases likelihood of underfitting. Low value increases training time.	1
gamma	Minimum loss reduction for a leaf to be split.	Low gamma enhances overfitting probability. Conversely, high gamma make the model more conservative and prone to underfitting. Low value increases training time.	0

colsample_by_tree	Determines the fraction of features to train on.	Lower values induce more randomness and less likelihood of overfitting. Contrary, increases risk of underfitting.	1
num_round	Number of iterations/trees built	Underfitting is tackled by increasing iterations. Contrary, decrease iterations if overfitting is observed	100
subsample	Determines the fraction of randomly sampled data applied for training. Counts for each trees in the forest. Must be larger than 0. 1: All data used 0.01: 1% used.	Lower values enhances randomness and regularization and thus decrease likelihood of overfitting. Contrary, may leave out important data. High values have opposite effect. Lower values reduces training time.	1

2.2.5 Light Gradient Boosted Machine

Light Gradient Boosted Machine (LightGBM) is a GBDT algorithm using a leaf-wise tree structure, contrary to the level-wise growth in XGBoost[15]. LightGBM addresses the issue of scalability and efficiency for large data sets by introducing Gradient-based One-Side Sampling (GOSS) and Feature Bundling (EFB). LightGBM is proven to provide superior performance relative to XGBoost in terms of computational efficiency and memory usage, while sustaining same level of accuracy. GOSS works by filtering out data providing less information, and thus train on less and more important data. Consequently, training time is decreased. EFD bundles related features together to decrease the number of features without hurting the accuracy to a significant degree.

Regarding hyperparameter tuning, the developers of LightGBM (Microsoft Corporation) provide a set of recommendations[36]. Given LightGBM's low relative training time, it may be advantageous to prioritize tuning for improved accuracy and mitigating overfitting. To enhance accuracy, the developers suggest tuning *max_bin*, *learning_rate*, *num_leaves* or *dart*. For tackling overfitting, many of these parameters can be adjusted in the opposite direction. Table 2.3 depicts these critical hyperparameters in LightGBM, along with their explanations and default values collected from `LGBMRegressor` class in `lightgbm` scikit-learn API[37].

Table 2.3: Important hyperparameters in LightGBM[36]. Default values are collected from LightGBM's documentation[38].

Parameter	Description	Importance	Default
max_bin (this is not directly interpretable in Scikit-learn)	Categorizes feature values into bins or intervals based on their value. Higher bin value means shorter interval in the bin.	Higher maximum bin can enhance accuracy with the drawback of larger training time. max_bin is not directly accessable in Scikit-learn LGBMRegressor. Increase num_leaves instead.	N/A
num_ leaves	Max number of leaves in one tree.	Large and small values may induce overfitting and underfitting, respectively.	31
min_ data_ in_leaf (min_child_ samples in Scikit-learn)	min_child_ samples is the minimum number of data needed in a leaf.	Small and large values may result in overfitting and underfitting, respectively.	20
learning_ rate	See "eta" in XGBoost.	See "eta" in XGBoost.	0.1
boosting_ type("dart")	Trains solely on a random subset of the existing tree.	Increases randomization, and have thus proven to enhance accuracy.	N/A

2.2.6 Categorical Boosting

Categorical Boosting (CatBoost) is a GBDT machine learning algorithm that has demonstrated superior performance to other gradient boosting algorithms such as XGBoost and LightGBM in various machine learning tasks[16]. Prokhorenkova et al. (founders of CatBoost) demonstrated this in their paper comparing CatBoost, LightGBM, and XGBoost on several known machine learning tasks[16]. CatBoost significantly outperformed LightGBM and XGBoost in all cases, while LightGBM and XGBoost performed relatively similarly. For some tasks, XGBoost and LightGBM were beaten by a logarithmic error 21% higher than CatBoost.

CatBoost employs symmetric binary decision trees with leaf-wise growth by default[16]. Symmetric and asymmetric trees are compared in Figure 2.9. As the splitting criteria are the same at the same level in symmetric trees, they tend to be more balanced and less likely to overfit, as well as requiring less computational power. CatBoost has gained popularity due to their enhanced capability of handling categorical data, while still providing great performance in predicting numerical outputs.

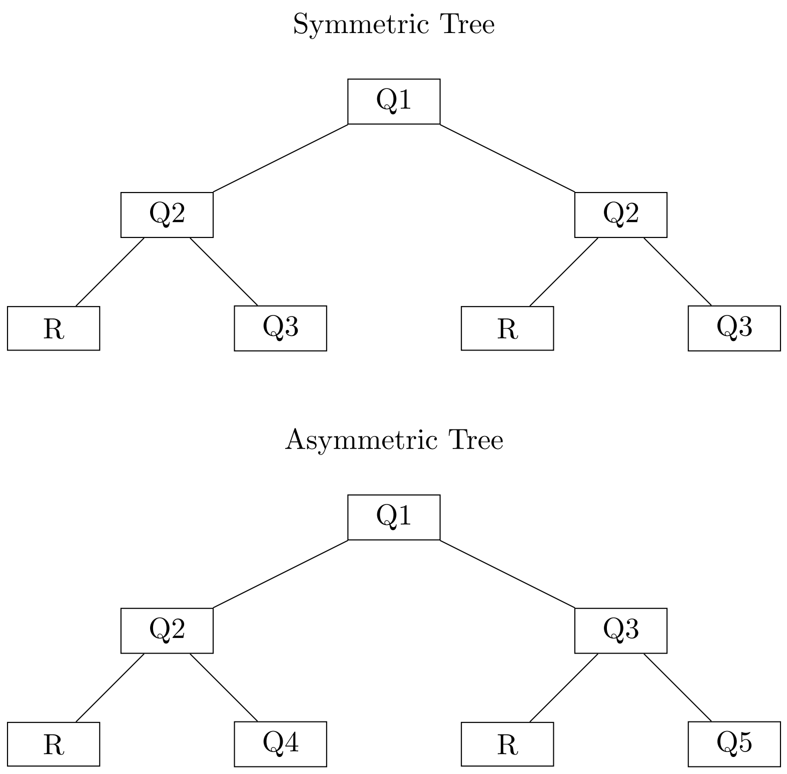


Figure 2.9: Comparison of symmetric (upper) and asymmetric (lower) decision trees. *Q* and *R* represent *question* and *residual*, respectively, where the question is the optimized splitting criteron. In the symmetric tree, the node splitting criteria is equal across the whole tree level. This is not the case or an asymettric approach, where all nodes can exhibit distinct splitting criteria.

Regarding hyperparameter tuning, CatBoost developers provide recommendations on how to optimize your model for the specific task[39]. Initially, they suggest analyzing the validation data for any obvious signs of underfitting or overfitting. To tackle the underfitting and overfitting simultaneously, they recommend implementing early stopping combined with a high number of iterations. In Table 2.4, the developers highlight the most important parameters to tune. CatBoost developers also suggest testing with multiple tree structures, e.g. asymmetric trees.

Table 2.4: Recommended parameters to tune are collected from CatBoost documentation[39]. Default values are collected from lightgbm library[40].

Parameter	**Description**	**Importance**	**Default**
n_estimators	See "num_round" in XGBoost.	See "num_round" in XGBoost.	1000

learning_ rate	See ”eta” in XGBoost.	See ”eta” in XGBoost.	Assigned automatically depending on num trees. 0.3 when 1000 iterations. Decreases for each iteration.
l2_leaf_reg	Coefficient in the L2 regularization.	Higher value discourages the model to learn too complex relationships and thus prevent overfitting. Contrary, increases training time as the information gain from each tree becomes smaller.	3
random_ strength	Amount of randomness when scoring node splits. A random score with mean 0 and decreasing variance for each iteration is added to the score. A value ”1” does however not mean 1 is added, just that randomness increases.	Higher value prevents overfitting, but increases training time as the model will converge slower.	1

border_ count	The number of splits for numerical features, i. e. put that amount of data into separate bins.	Higher value enhances overfitting probability as leaf nodes become more pure. However, higher accuracy can be reached. A higher value will enhance training time.	CPU: 254 GPU: 32/128

2.2.7 Artificial Neural Network

Artificial Neural Networks (ANNs) are class of supervised machine learning algorithms that attempts to optimize neuron-to-neuron connection parameters to minimize prediction error by modelling the human brain[24]. ANNs are well-suited for complex machine learning tasks with non-linear relationships and have been highly successful on platforms like Kaggle[41]. ANNs have a more complex architecture compared to decision tree algorithms. In this work, feed-forward structure (no cycle learning) is discussed. ANNs with *dense* layers with a *sequential* structure is considered, which refers to a stack of layers (sequential) where all neurons in layer i are connected to all neurons in layer $i+1$ (dense) with each neuron receiving and returning one tensor.

ANNs consist of a series of three layers: an input layer (IL), one or multiple hidden layers (HL), and an output layer (OL)[42]. The IL is where training data is provided. The HLs are responsible for solving non-linearities in the data, and the number of HLs required depends on the complexity of the problem being solved. Finally, the OL provides the output and prediction of the target variable(s). It is important to adapt the design of the network to the specific problem, and the optimal number of HLs is difficult to interpret, and should be explored through hyperparameter tuning. However, a common thumb rule is having a number of hidden layers greater than 2[43].

ANNs are iterative algorithms that aim to decrease the prediction error for each iteration through the network based on common loss functions such as MSE and MAE[44]. The optimization process involves finding optimal neuron-to-neuron connection parameters, where each neuron in the previous layer is connected to every neuron in the next layer, and a neuron's output is a prediction based on the given input variable(s)[24]. The parameters that are optimized are known as weights (w) and biases (b)[42]. Weights w are scalar values that determine the influence of each variable on the output. The bias b is a constant adjustment value unique to each layer and is used to shift the neurons toward activity or inactivity. The values of w and b are initially assigned random values or guessed qualitatively by the developer and then adjusted through training during each iteration. The output from the previous neuron is multiplied with w associating those two neurons, with a unique b added to it. As neurons accepts array-like objects, this calculation is executed element-wise.

Figure 2.10 shows an ANN architecture designed for electrochemistry applications, where the input variables are pH and potential E. Each neuron in the IL contains n data points from a unique variable, which are fed as array-like objects. The weights (w) in the IL are assigned a value of 0.5 by the ANN, while biases are represented by b_1 and b_2. Weights and biases are updated after the completion of one cycle, known as an epoch, where all *batches* have been processed once. A batch is a fraction of the training data. Common utilized batch sizes are > 16 with intervals of 16[45]. The batch size is a parameter assigned initially by the developer and remains constant throughout the training process.

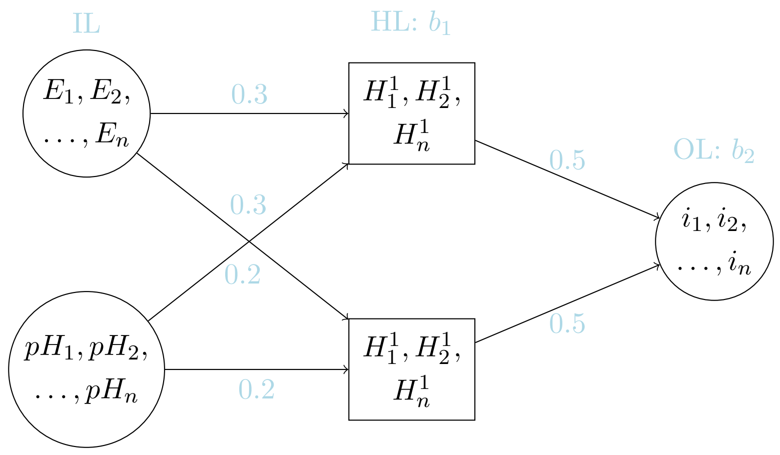


Figure 2.10: A schematic representation of the architecture of an Artificial Neural Network (ANN) with potential (E) and pH as input features. In the IL, the data associated with each variable are placed into their respective variable neuron. The activation function is then applied to the weighted sum of the neuron outputs, producing the output in the HLs. Initially, weights are assigned either randomly or qualitatively. Subsequently, the weights are adjusted with each batch using an optimizer of choice. The activation function processes the output generated by the hidden layers to produce the current density output in the output layer. This output is subsequently assessed against a set of validation data based on a selected loss function. Training continues as long as an optional stopping criteria is not met.

The input data to the two neurons in the HL in Figure 2.10 can be described mathematically through Eq. 2.14 and 2.15[46]. H^1 and H^2 are a value H in neuron 1 and 2 with x being a value between 0 and n, where n represents the length of the array provided in each neuron in the IL.. i represents the node under consideration in layer j. For any value x in the first neuron in the HL:

$$H^1_{\mathrm{x}} = \sum_{i=1}^{\mathrm{i=n}} \left[X^{\mathrm{i}}_{\mathrm{j=0}} \cdot w(X^{\mathrm{i}}_{\mathrm{j=0}}, H^1_{\mathrm{j=1}}) + b_{\mathrm{j=1}} + (Y^{\mathrm{i}}_{\mathrm{j=0}} \cdot w(Y^{\mathrm{i}}_{\mathrm{j=0}}, H^1_{\mathrm{j=1}}) + b_{\mathrm{j=1}} \right] , \qquad (2.14)$$

and for any value x in the second neuron in the HL:

$$H^2_{\mathrm{x}} = \sum_{i=1}^{\mathrm{i=n}} \left[X^{\mathrm{i}}_{\mathrm{j=0}} \cdot w(X^{\mathrm{i}}_{\mathrm{j=0}}, H^1_{\mathrm{j=1}}) + b_{\mathrm{j=1}} + (Y^{\mathrm{i}}_{\mathrm{j=0}} \cdot w(Y^{\mathrm{i}}_{\mathrm{j=0}}, H^1_{\mathrm{j=1}}) + b_{\mathrm{j=1}}\right]. \tag{2.15}$$

X and Y in this context represent potential (E) and pH. In order to improve training performance, features should be scaled (independently) as they can vary by orders of magnitude[24]. This can e.g. be done by min-max-scaling, where the features are scaled to a number between e.g. 0 and 1.

As previously mentioned, the bias b plays a role in shifting neurons towards activity or inactivity. Whether a neuron is considered active or inactive is determined by the *activation function*, which is also called a *transferring* function as it transforms the input of a neuron to an output[46]. Multiple activation functions can be applied depending on layer, however it is most common to apply the same activation across all neurons in one layer. The scalar obtained from the weighted sum of input data is passed through this activation function, which transforms it into values the ANN can process. A popular activation function is the Rectified Linear Unit (ReLU) because of its enhanced speed and accuracy[24]. Unlike other activation functions such as the sigmoid and tanh function, ReLU sustains a constant gradient, eliminating the problem of vanishing gradients[24;47]. The mathematical expression for the ReLU activation function is shown in Eq. 2.16:

$$f(x) = \max(0, x), \tag{2.16}$$

where x represents each neuron's output (before fed to the activation function).

Eq. 2.16 implies that the ReLU activation function returns only positive values. Hence, the bias can be used to make the neuron inactive (as 0 is returned) by being a sufficiently negative value.

Furthermore, the weights and bias are optimized to minimize the error using the *optimizer*, which finds the optimal values by adjusting the weights and bias iteratively with a defined learning rate. The choice of optimizer is important and problem-specific, but Adaptive Moment Estimation (ADAM) remains a popular choice due to its superior performance in many cases[48]. As for the GBDT algorithms, a low learning rate could result in getting trapped in local error minima, while a high value could lead to the optimizer failing to find the global minimum. These phenomena are demonstrated in Figure 2.12. In Figure 2.11, a good learning rate is recognized by a lower error gradient, but where the error ultimately converges to a lower value for a large number of epochs. Utilization of a lower learning rate will however indeed increase training time as the learning process is slower.

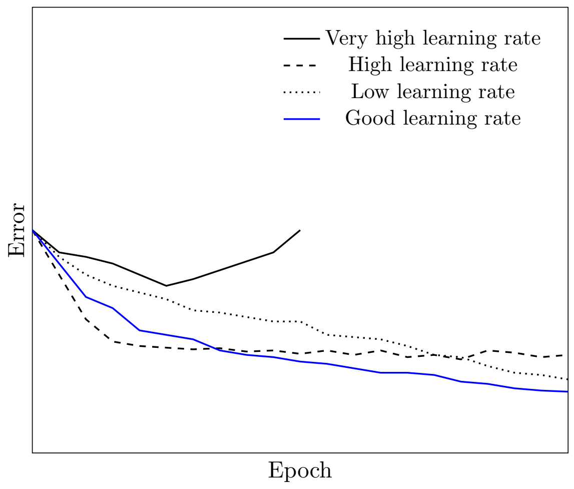


Figure 2.11: Comparison of different learning rates and their impact on the error

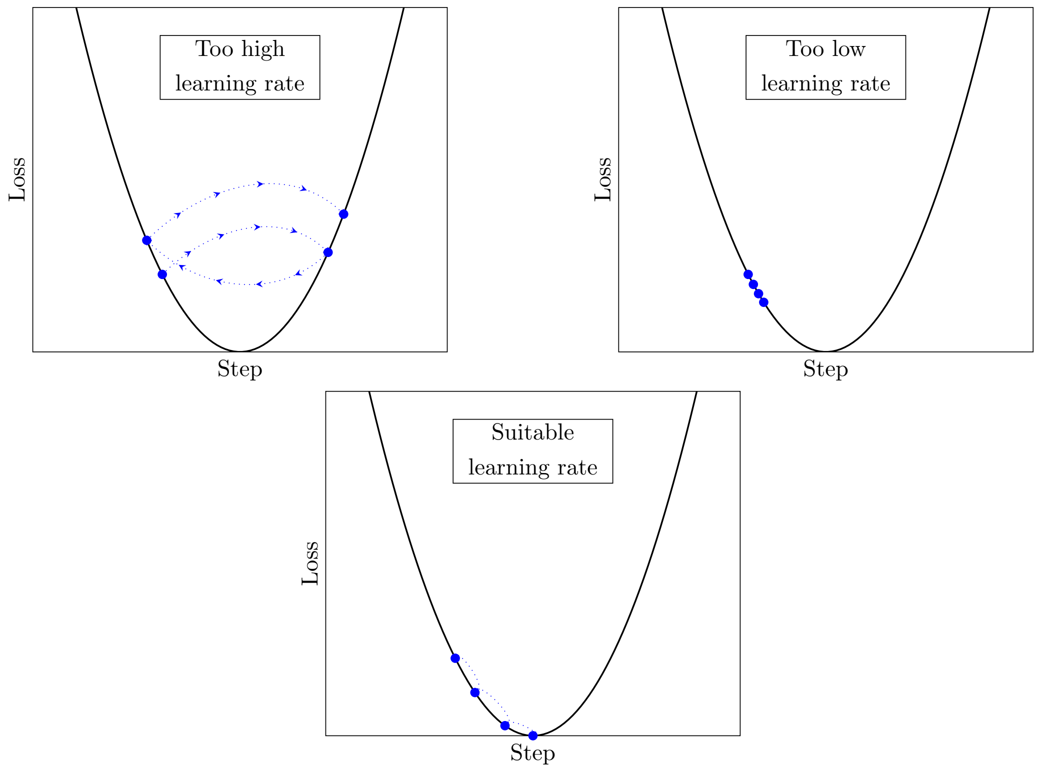


Figure 2.12: Too high (upper left), too low (upper right) and suitable (bottom) learning rate for an iterative machine learning algorithm

After each epoch, the model performs one prediction based on what it has learnt so far. This prediction is compared with unseen data based on the loss function. The unseen data is known as validation data. Prior to training, all data is divided into training and validation, with common percentage splits of 70/30, 80/20 and 90/10 (train/test), where the best split depends on the nature of the underlying data and the amount of data available[49]. The fraction of validation data plays a significant role, as an excessively high or low proportion can lead to underfitting or overfitting, respectively[50].

ANNs are highly complex and require the specification of many parameters. Selecting appropriate parameters is as discussed a challenging task, and thus parameter tuning techniques like Random Search (RS) and Grid search (GS) are applied. These methods will be discussed more comprehensively in Section 2.2.8.

2.2.8 Hyperparameter tuning

When training any machine learning algorithm, selecting appropriate model parameters before training is crucial to obtain optimal performance[51]. For certain problems, the hyperparameter combination can be the difference between a horrible and great model. For instance, in a structural design problem executed by X. Du et al., an Artificial Neural Network (ANN) achieved a 45.2% reduction in Root Mean Square Error (RMSE) following hyperparameter optimization[51].

Determining optimal hyperparameters for a machine learning algorithm is a challenging task and often requires a trial-and-error approach. Doing this manually can be time-consuming and counter productive. Thus, developers often use optimization methods, such as Random Search (RS) and Grid Search (GS). Both methods test a set of hyper-parameter combiations and return the combination providing the lower error[52]. GS and RS do however have differing approaches. GS tests all combinations within a predefined search space, while RS randomly samples and tests hyperparameter combinations. Each method has its own set of advantages and disadvantages. For instance, if the objective is to comprehensively explore all combinations in the search space, GS could be a more beneficial approach. On the other hand, if the search space is large, RS may offer a more computationally efficient solution due to its random sampling strategy.

Developers of machine learning algorithms provide optimized default values for their algorithms. For many tasks, using these can provide great results[16]. However, doing a detailed hyperparameter search is likely to enhance the predictability and should be executed if the trade-off between accuracy and training is acceptable. For GBDTs, a good way in the event where tuning is not affordable, utilizing regularization techniques such as early stopping is beneficial, as outlined in Section 2.2.1. For RF however, this is not suitable. Furthermore, neural net libraries such as `Keras` require the developer to determine all hyperparameters pre-training, as no default values are provided. Hence, hyperparameter tuning when using ANNs is highly important[51]. The necessity of tuning RF and the GBDTs algorithms discussed is attempted clarified in the following sections. However, it is important to keep in mind that all machine learning tasks should be treated independently. It is not possible to provide a definitive answer regarding the necessity of tuning. One potential approach is to initially apply default values and subsequently fine-tune the model.

2.3 Statistics for non-distributed data

In many instances, the data set does not follow a certain distribution or known function. In these cases, constructing trend lines can be of interest. One such method is by applying a *Simple Moving Average* (SMA), which is constructed by calculating an average for a predetermined window of the total data set. The SMA mathematical formula is given by Eq 2.17[53]:

$$\mathrm{SMA}_n = \frac{1}{n} \sum_{i=k-n+1}^{k} f(x_i), \tag{2.17}$$

where n is the fixed number of data points in the window, i equals the data point under consideration, and k is the relative position of the window being under consideration. A benefitial effect of the SMA is capturing local trends in the data, as well as getting a better understanding of likely future outcomes. The SMA weighs all data points equally.

3 Experimental

3.1 Electrochemical experiments

2 x 2 pieces of extruded AA6060 + 0.0043 wt% nickel were cut from billets provided by Norsk Hydro ASA. The acidic and alkaline electrolytes were formed by using hydrochloric acid and sodium hydroxide together with necessary sodium chloride to obtain 0.1 M chloride solutions. The pH range under consideration was pH between 2.0 and 12.0 with steps of 0.2, resulting in 51 experiments. No electrolyte was reused in order to prevent contamination. Conversely, due to sample limitations, it was necessary to apply the same AA6060 sample multiple times as long as it did not exhibit a substantial degree of pitting.

Platina (Pt) and Standard Calomel Electrode (SCE) were used as counter and reference electrode, respectively. The cell setup is depicted in Figure 3.1. No salt bridge was applied as 0.1 M chloride content was assumed to provide satisfactory conductivity. The experiment sequence was initiated with a measurement of the corrosion potential E_{corr}, lasting one hour or when it reached stability. The sequence was proceeded by cyclic voltammetry (CV) with a lower and upper scan limit of -0.35 and 0.35 V vs E_{corr}, respectively. A sampling period of 0.1 seconds was applied to obtain an appropriate amount of training data. Empirical data obtained from laboratory can be found in Appendix F (E_{corr}) and E (CV).

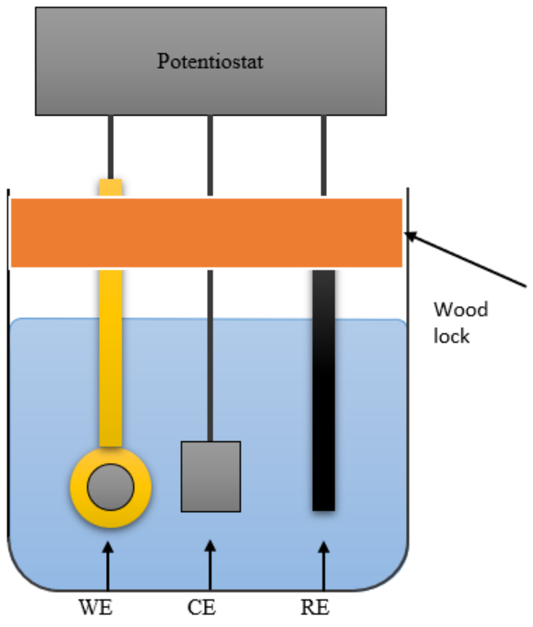


Figure 3.1: Electrochemical cell setup with Pt as Counter Electrode (CE), Standard calomel electrode (SCE) as Reference Electrode (RE) and AA6060 + 0.0043 wt% as Working Elecrtode (WE)[5].

3.2 Machine Learning in Python

The total amount of available data for training was 707913 data points. Training and testing data was organized according to Table 3.1, having pH and potential (E) as input features, and logarithmic absolute current density ($\log_{10}(|i|)$) as output. The full training data set can be found in training_data.csv on Github main branch[1].

The applied classes or libraries for creating instances of models were `sklearn.ensemble.RandomForestRegressor`[27] (Random Forest), `catboost.CatBoostRegressor`[16] (CatBoost), `xgboost.XGBRegressor`[54] (XGBoost), `lightgbm.LGBMRegressor`[37] (LightGBM) and `Keras` (running on top of `TensorFlow`, for ANN[55]).

Data was filtered prior to training. Data from t = 0 seconds until the reverse cathodic scan began (potential minima) were removed for a smoother output. The filtering process can be found in *src/filter_raw_data.py* on Github main branch[1]. In addition, obvious noise from the potentiostat were removed. At the shift in 10-based logarithm, small rapid shifts over a short period were observed and deleted from the data set. The result of the filtering process can be observed in Appendix E.

Post filtering, data was separated into training and testing with a 80/20% split. The same split was applied for all algorithms such that they would train and validate on the same data. No further preprocessing step was applied in the case of DT algorithms. On the other hand, normalization using MinMaxScaler in `scikit-learn` was employed as a pre-processing step prior to training of the ANN. A feature range of [0, 1] was utilized as all output values ($\log_{10}(|i|)$ had equal sign. Code is found in *src/data_preprocessing.py* on main branch on Github[1].

Furthermore, hyperparameter tuning was employed on several algorithms. Hyperparameter tuning was not applied to CatBoost or XGBoost, but it was implemented on LightGBM due to its faster training times per iteration. Solely one GBDT algorithm was tuned as they have comparable architectures. Instead, early stopping with 10000 iterations were employed in order to terminate training before overfitting to training data. For RF, early stopping was not an option due the different training architecture. Hence, manual hyperparameter tuning according to the suggestions of the`scikit-Learn` developers, given in Section 2.2.3, was employed. This included testing different amount of trees in the forest (*n_estimators*) and the maximum number of features to consider in each split (*max_features*). For the ANN, `RandomSearch` class in `Keras` was applied[2]. For LightGBM, `GridSearchCV` in `scikit-learn` was utilized[3].

The ANN hyperparameters were tuned successively by narrowing down the search space for each tuning attempt. The applied hyperparameters for all models can be found in

[1]Github repository : `https://github.com/matsssk/AA6060_ML/tree/main`

[2]`keras_tuner.RandomSearch`: `https://keras.io/api/keras_tuner/tuners/random/`

[3]`sklearn.model_selection.GridSearchCV`:`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

Appendix A, along with the hyperparameter tuning results of RF, LightGBM and ANN. Tuning related data can be found on Github[1]. Code for training models are given in *src/train_models.py* on Github[1].

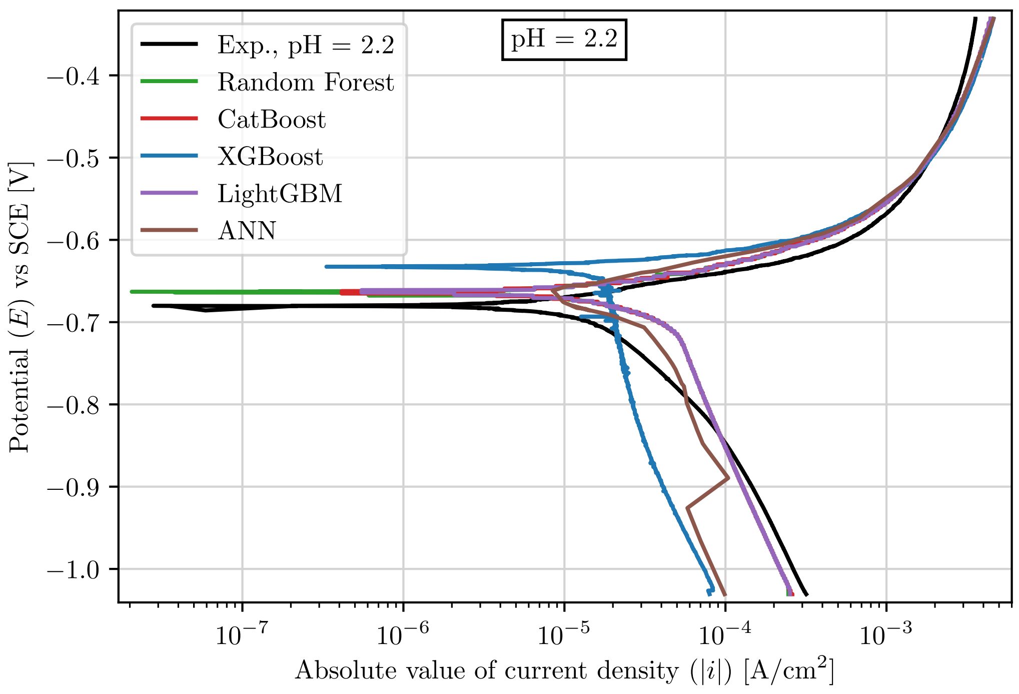
Table 3.1: Training data sorted into labeled tabular data

CV	**Input**		**Output**
	E [V]	**pH**	**log(|i|)**
CV 1	E_1^1	2.0	$\log(\|i_1^1\|)$
	E_2^1	2.0	$\log(\|i_2^1\|)$
	$\vdots$	$\vdots$	$\vdots$
	E_n^1	2.0	$\log(\|i_n^1\|)$
CV 2	E_1^2	2.2	$\log(\|i_1^2\|)$
	E_2^2	2.2	$\log(\|i_2^2\|)$
	$\vdots$	$\vdots$	$\vdots$
	E_n^2	2.2	$\log(\|i_n^2\|)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
CV 51	E_1^{51}	12.0	$\log(\|i_1^{51}\|)$
	E_2^{51}	12.0	$\log(\|i_1^{51}\|)$
	$\vdots$	$\vdots$	$\vdots$
	E_n^{51}	12.0	$\log(\|i_1^{51}\|)$

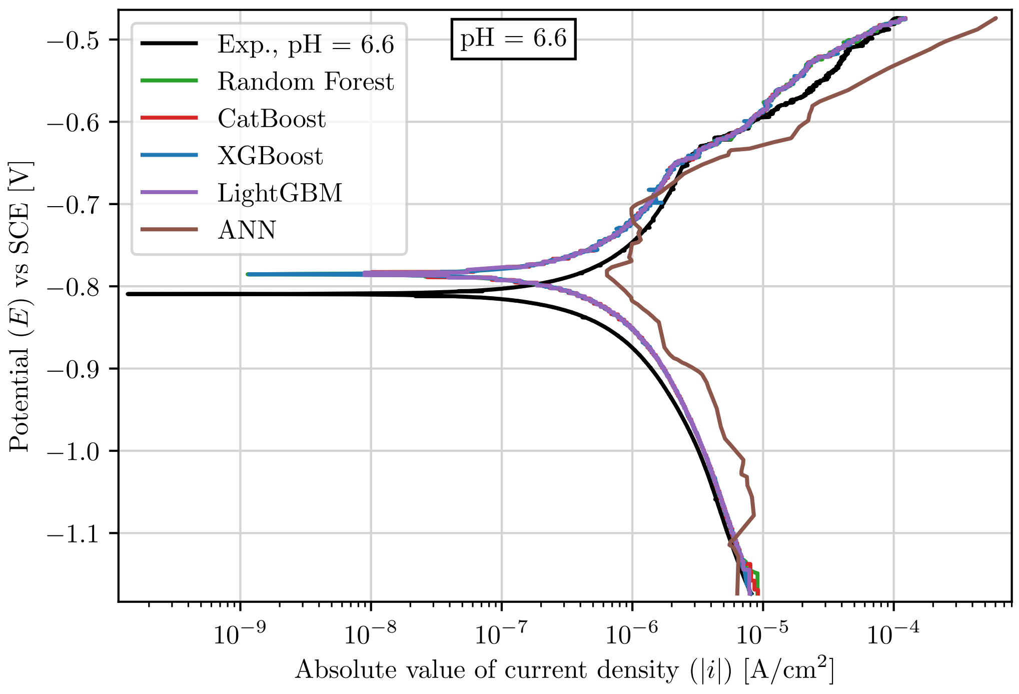
4 Results

4.1 Polarization curve predictions

Figure 4.1 illustrates the comparison between experimental data (Exp.) and the predicted polarization curves, as a function of both potential E and pH, generated by all algorithms for each set of test data at pH levels of 2.2, 6.6, 7.4, and 11.6. In Table 4.1, a comparison is made for each individual model against the corresponding empirical data. The values representing the minimum and maximum Mean Absolute Percentage Error (MAPE) at each pH level are distinctly highlighted in green and red, respectively. The algorithms demonstrated a generally robust capability for adapting to the training data, with the average of the Mean Absolute Percentage Error (MAPE) of the target output being approximately 7%.

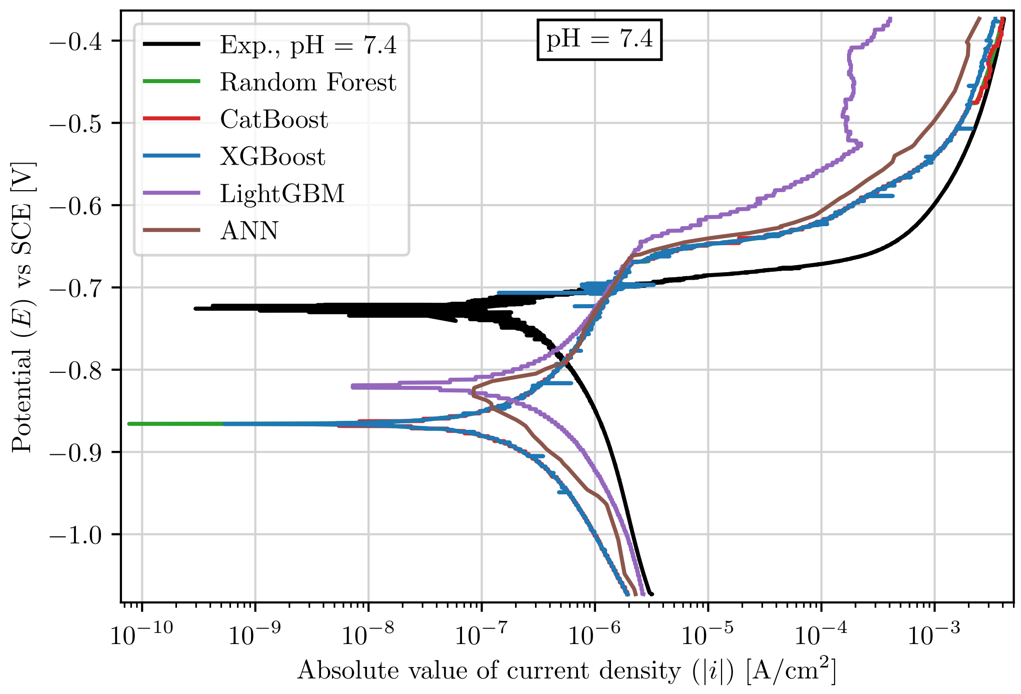


(a) pH = 2.2

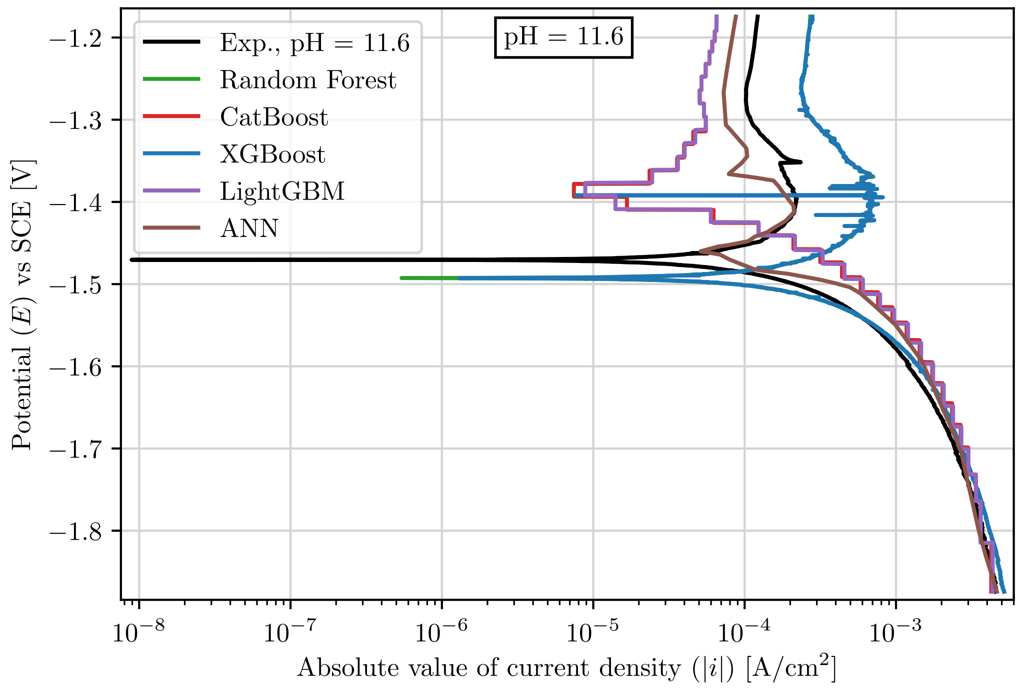


(b) pH = 6.6

Figure 4.1: Experimental data compared with the predicted polarization curves, with *Exp.* short for Experimental (Continued on next page).



(c) pH = 7.4



(d) pH = 11.6

Figure 4.1: (Continued from previous page) Experimental data compared with the predicted polarization curves, with *Exp.* short for *Experimental*.

Table 4.1: The prediction error of $\log_{10}(|i|)$ for each algorithm, quantified in both Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE), is reported. The lowest and highest average MAPE values across models and pH levels are denoted in red and green, respectively.

	RF	**Cat-Boost**	**XG-Boost**	**Light-GBM**	**ANN**	**Mean**
pH	**MAPE (RMSE) [$\log_{10}(\|i\|)$]**	**MAPE (RMSE) [$\log_{10}(\|i\|)$]**	**MAPE (RMSE) [$\log_{10}(\|i\|)$]**	**MAPE (RMSE) [$\log_{10}(\|i\|)$]**	**MAPE (RMSE) [$\log_{10}(\|i\|)$]**	**MAPE (RMSE) [$\log_{10}(\|i\|)$]**
2.2	3.46 (0.248)	3.45 (0.243)	8.46 (0.438)	3.46 (0.241)	5.54 (0.282)	4.87 (0.290)
6.6	2.48 (0.258)	2.46 (0.257)	2.41 (0.257)	2.43 (0.257)	5.55 (0.391)	3.07 (0.284)
7.4	11.57 0.713)	11.54 (0.710)	11.96 (0.712)	24.61 (1.03)	14.21 (0.722)	14.78 (0.778)
11.6	6.72 (0.371)	8.18 (0.465)	6.71 (0.370)	8.18 (0.464)	3.16 (0.187)	6.59 (0.371)
Mean	6.06 (0.398)	6.41 (0.419)	7.39 (0.444)	9.67 (0.498)	7.12 (0.395)	7.33 (0.431)

4.2 Individual model comparison with empirical data

In Figures 4.2, 4.3, 4.4, 4.5, 4.6, each model, created with optimized hyperparameters, is individually compared with the empirical data. Each figure includes the estimated Tafel slope obtained through linear regression, accompanied by their respective 99% confidence intervals (CI).

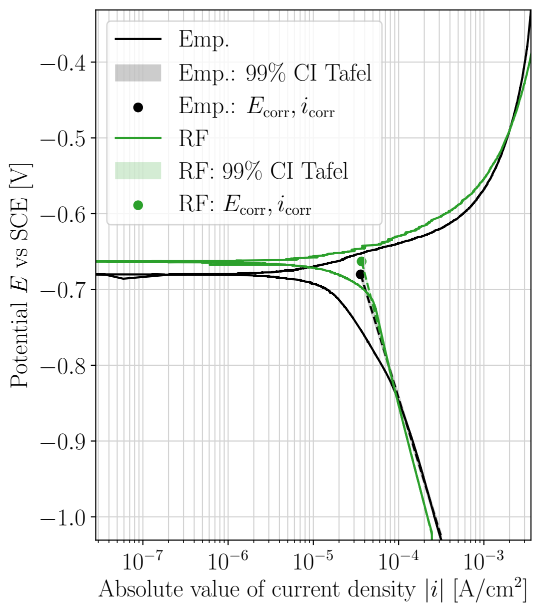
For the purpose of calculating cathodic Tafel slopes at pH 2.2 and 7.4, potentials greater than -1.05 V vs SCE were used due to a small, observable shift in slope around this potential, possibly due to HER. At pH 6.6, the cathodic Tafel slope was estimated using potentials greater than -1.1 V vs SCE, because a slope change was detected at potentials more negative than -1.1 V vs SCE. However, at pH 11.6, the same behaviour in slope change was not observed as the cathodic current was considerably large, showing signs of approaching the limiting current density i_{lim}.

In addition, only potentials lower than -0.15 V below E_{corr} were used for the Tafel line to ensure that the readings were sufficiently below E_{corr}. It is important to note that the narrow linear regions may suggest that a valid Tafel region is not definitively observed as ideally, the Tafel region should be calculated over at least one decade. To achieve this, the lower potential limit would have to be more negative than -0.35 V with respect to the corrosion potential E_{corr}. Then, on the other hand, the Tafel slope could be susceptible to change.

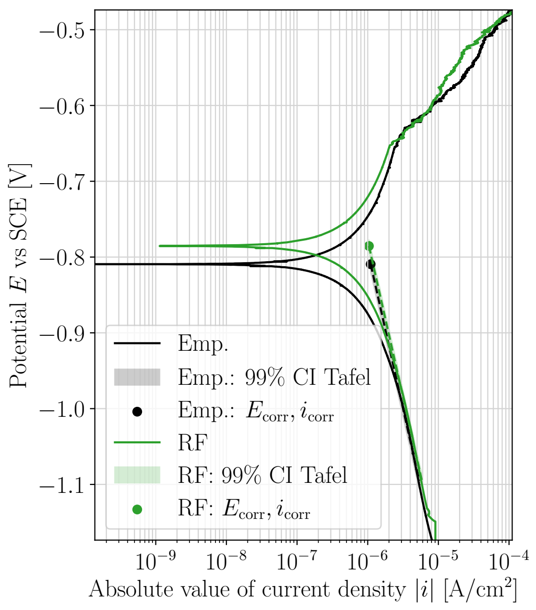
A normality test was performed on the residuals from the Tafel line to ascertain the validity of implementing a CI. Normality turned out to be a reasonable assumption, verified in the normality test in Figure C2a along with its standardization to obtain Figure C2b in Appendix C. For this normality test, the residuals were assumed independent on the Tafel slope, which was necessary to combine them into one single normality test. Details with utilized hyperparameters for each specific model, along with the associated outcomes from the tuning process (if tuning was executed), can be found in Appendix A.

In Table 4.2, 4.3, 4.4 and 4.5, the estimated corrosion potential E_{corr}, cathodic Tafel slope b_{c} and corrosion current density i_{corr} with their respective 99% CIs are given. The sign of the Percentage Error (PE) indicates for simplicity a lower (more negative) and higher (more positive) value compared to the empirical data. Additionally, take note that the PE of E_{corr} would indeed change depending on the applied reference electrode (RE). Still, a PE is provided for easier distinction between the empirical and predicted E_{corr}. The mean MAPE across all pH is given in Table 4.6. As the anodic reaction was unsuitable for measuring i_{corr}, the intersect between E_{corr} and the cathodic Tafel line was utilized instead. Furthermore, in Table 4.3, the measured pitting potentials E_{pit} are provided as pitting was observed at pH 6.6.

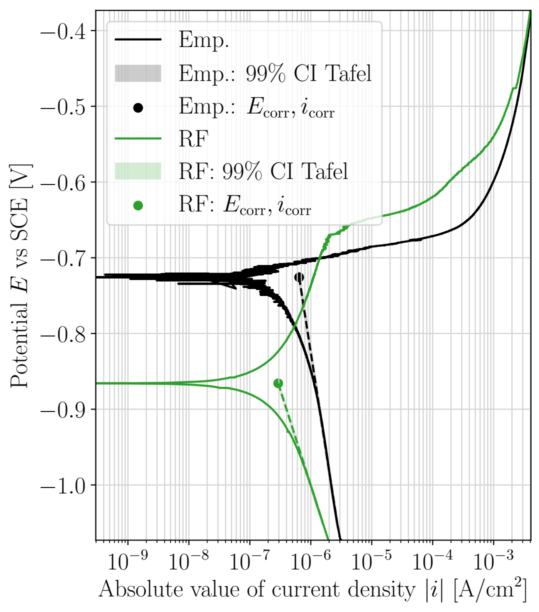
The linear regression used to calculate the cathodic Tafel slopes demonstrated considerable reliability, as evidenced by the narrow range of the 99% confidence interval. This resulted in a relatively high level of certainty when measuring the corrosion current density i_{corr}. However, the validity of the ANN Tafel slopes depicted in Figure 4.6a, 4.6b, 4.6c can be questioned. These cathodic branches showed untypical behavior, with rapid shifts in current that were not observed in the empirical data.



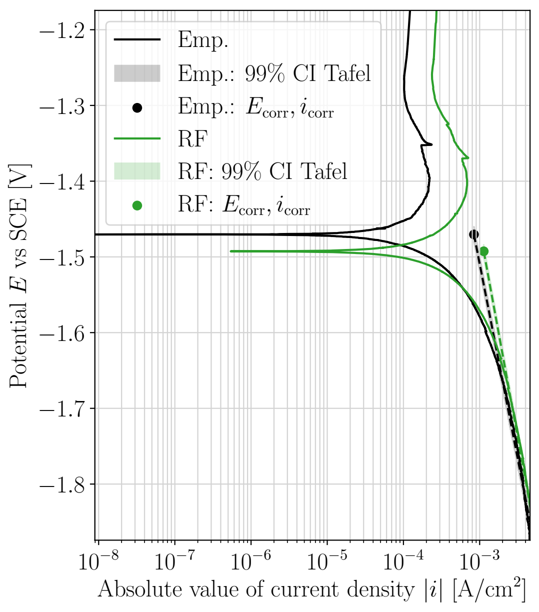
(a) RF, pH = 2.2



(b) RF, pH = 6.6

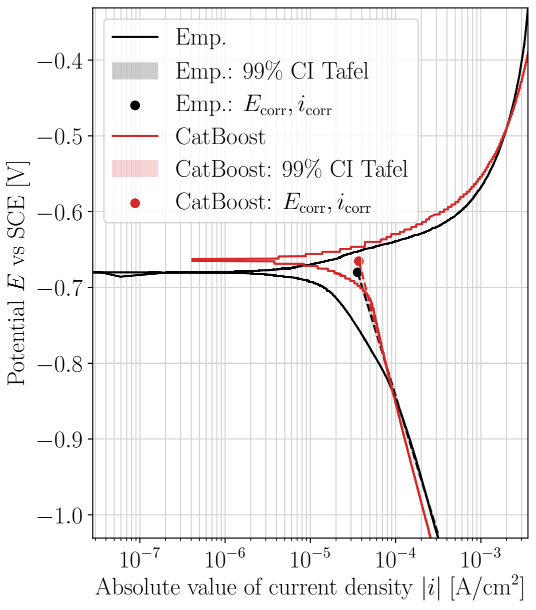


(c) RF, pH = 7.4

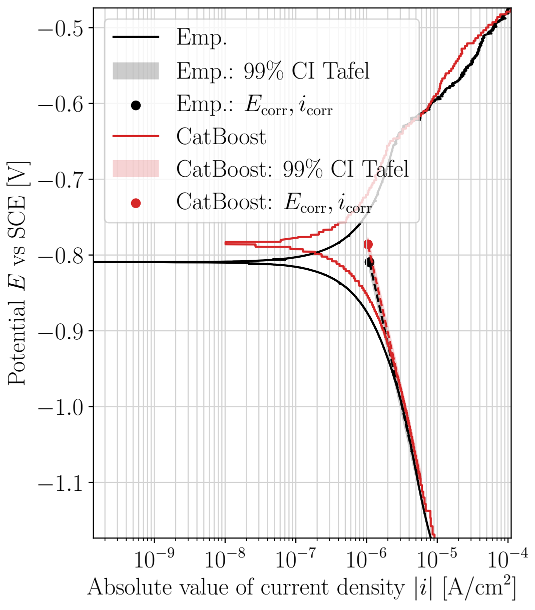


(d) RF, pH = 11.6

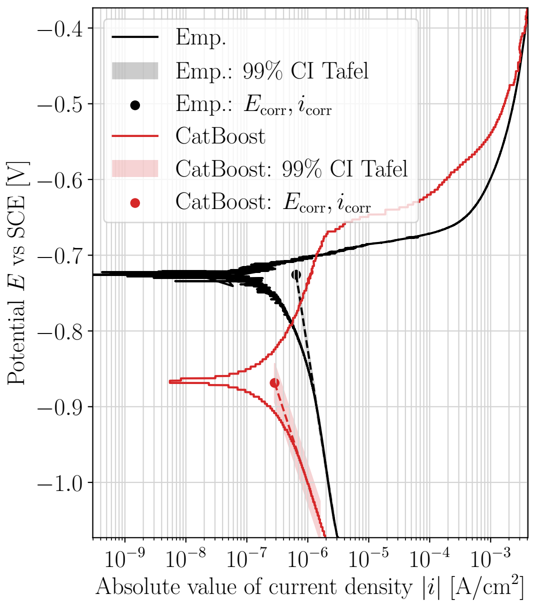
Figure 4.2: The RF results are compared to the empirical (Emp.) data for all test data. Included are the estimated Tafel line with a 99% confidence interval (CI), the corrosion potential E_{corr}, and the corrosion current density i_{corr}.



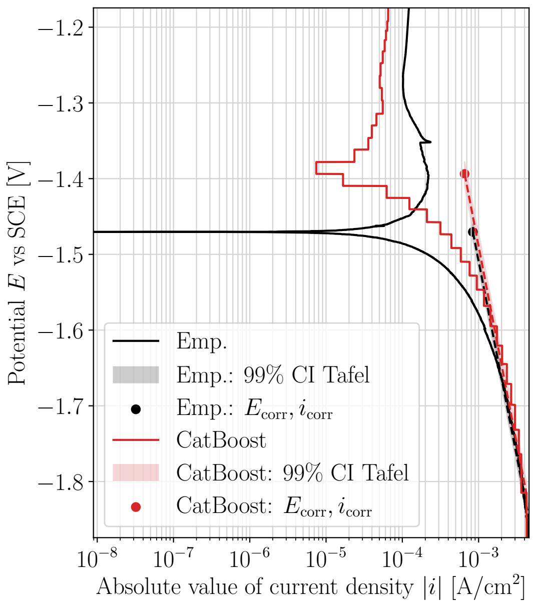
(a) CatBoost, pH = 2.2



(b) CatBoost, pH = 6.6

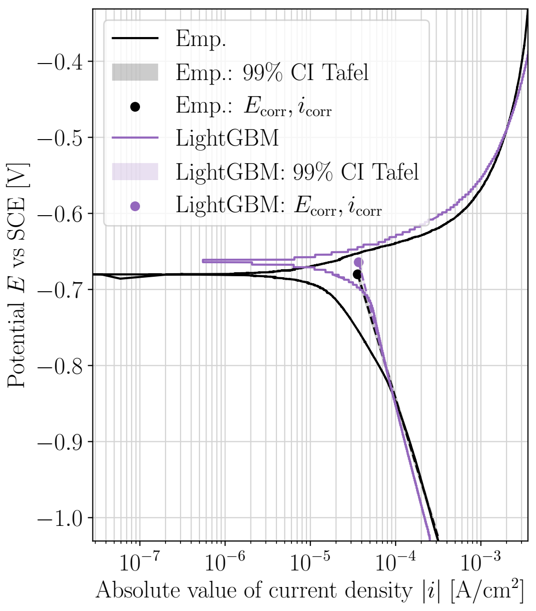


(c) CatBoost, pH = 7.4

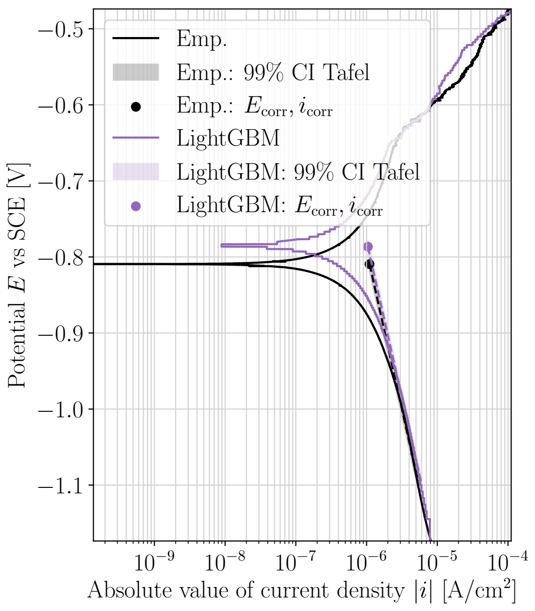


(d) CatBoost, pH = 11.6

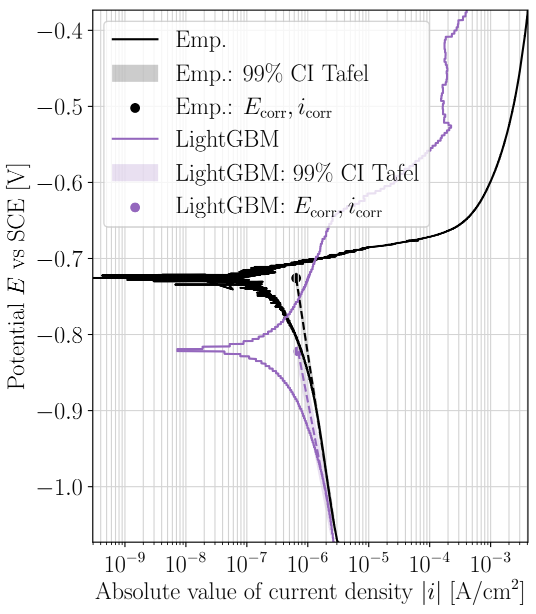
Figure 4.3: The CatBoost results are compared to the empirical (Emp.) data for all test data. Included are the estimated Tafel line with a 99% confidence interval (CI), the corrosion potential E_{corr}, and the corrosion current density i_{corr}.



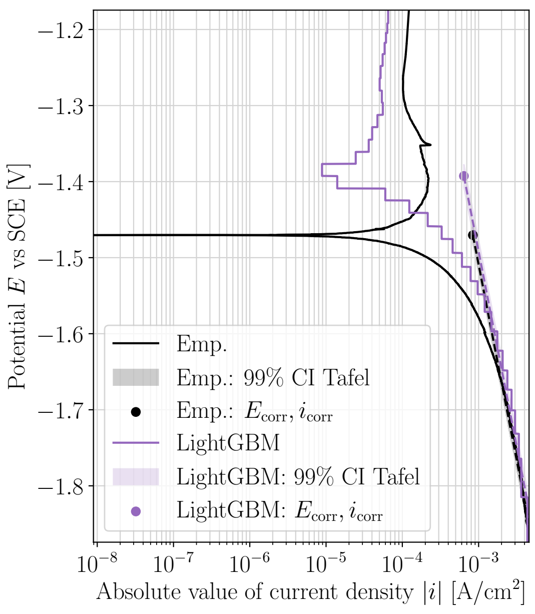
(a) LightGBM, pH = 2.2



(b) LightGBM, pH = 6.6

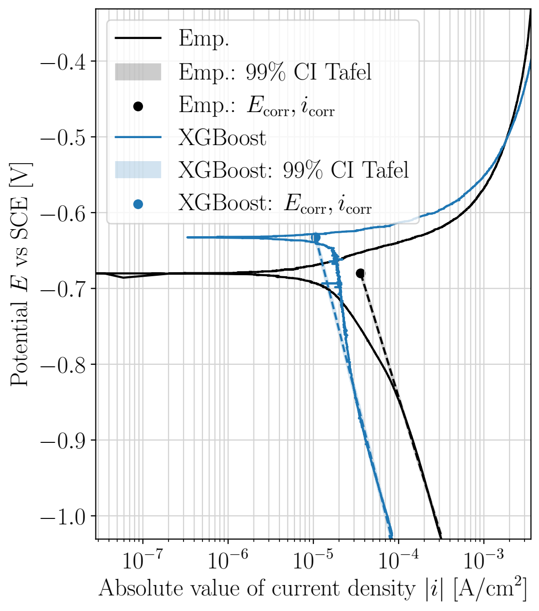


(c) LightGBM, pH = 7.4

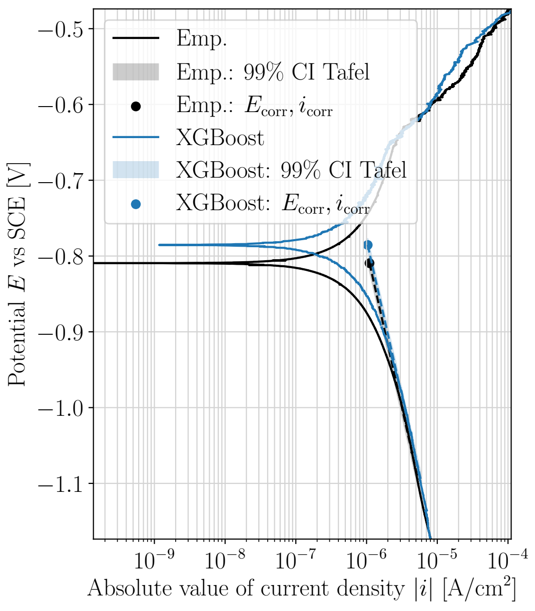


(d) LightGBM, pH = 11.6

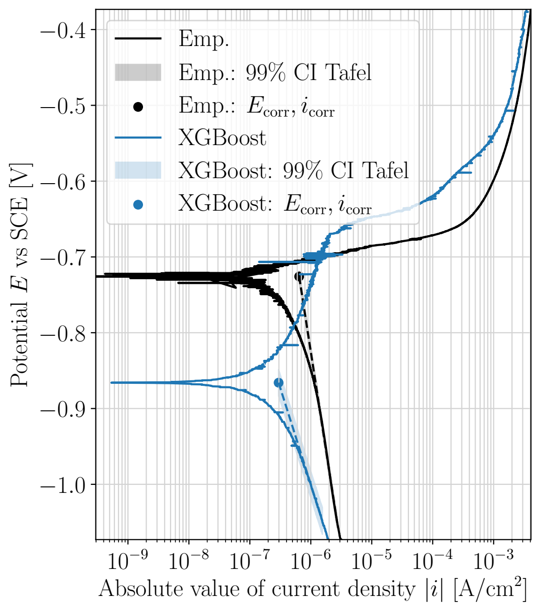
Figure 4.4: The LightGBM results are compared to the empirical (Emp.) data for all test data. Included are the estimated Tafel line with a 99% confidence interval (CI), the corrosion potential E_{corr}, and the corrosion current density i_{corr}.



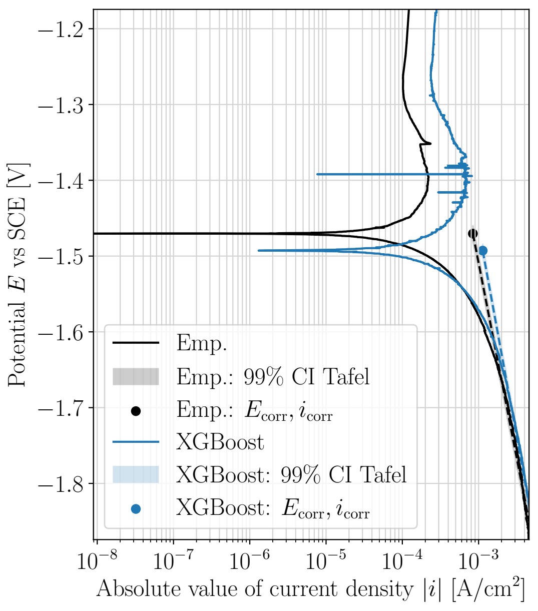
(a) XGBoost, pH = 2.2



(b) XGBoost, pH = 6.6

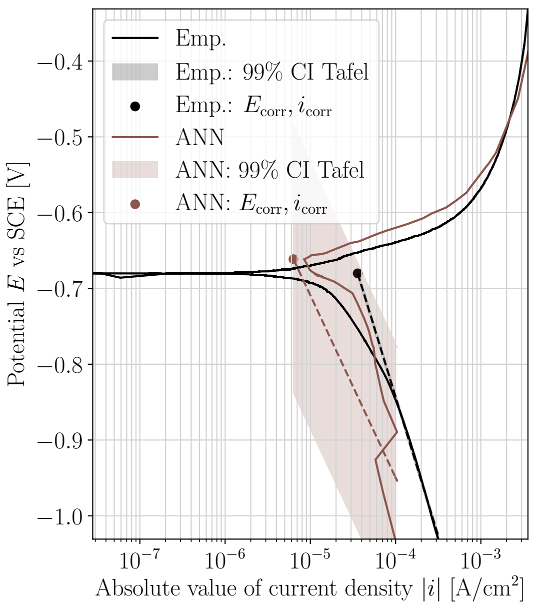


(c) XGBoost, pH = 7.4

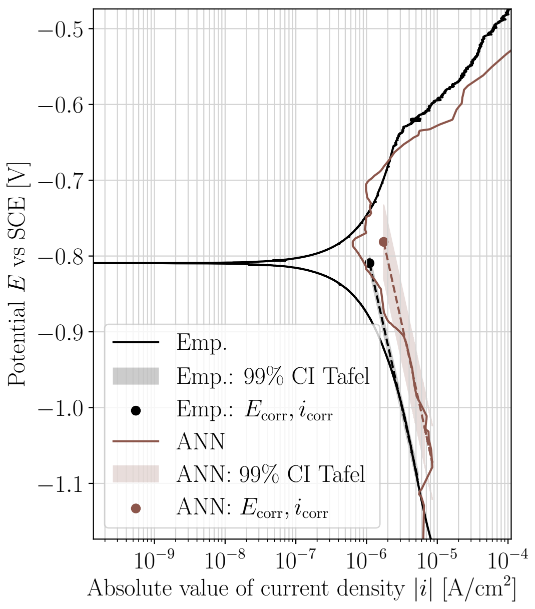


(d) XGBoost, pH = 11.6

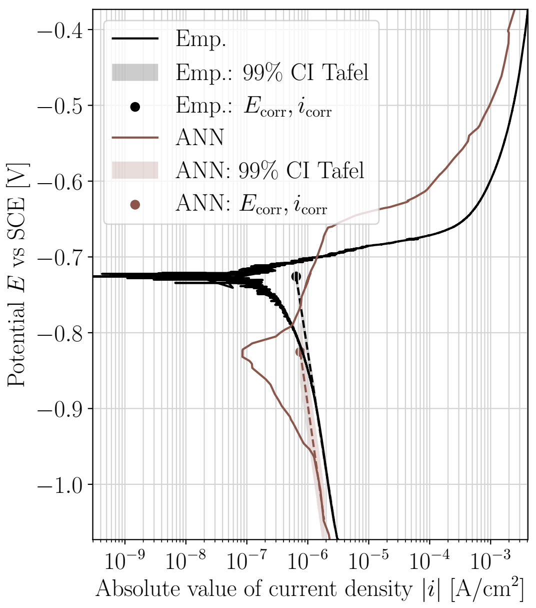
Figure 4.5: The XGBoost results are compared to the empirical (Emp.) data for all test data. Included are the estimated Tafel line with a 99% confidence interval (CI), the corrosion potential E_{corr}, and the corrosion current density i_{corr}.



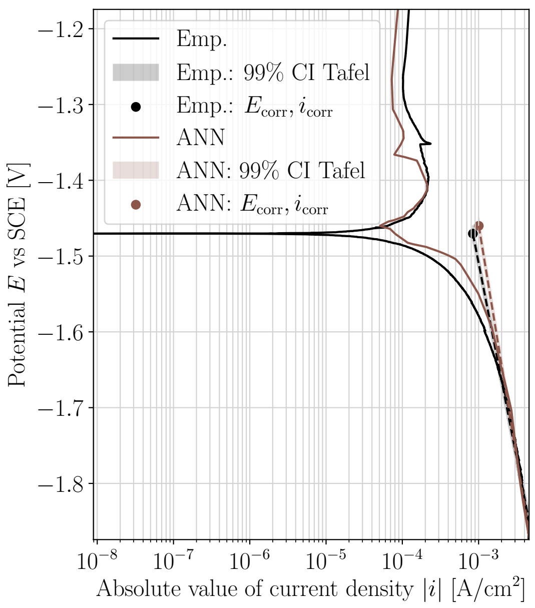
(a) ANN, pH = 2.2



(b) ANN, pH = 6.6



(c) ANN, pH = 7.4



(d) ANN, pH = 11.6

Figure 4.6: The ANN results are compared to the empirical (Emp.) data for all test data. Included are the estimated Tafel line with a 99% confidence interval (CI), the corrosion potential E_{corr}, and the corrosion current density i_{corr}.

Table 4.2: Polarization curve features at pH 2.2

	E_{corr} [mV]	Tafel slope b_c with 99% CI [mV/dec]		i_{corr} with 99% CI [µA/cm^2]	
		$[\bar{b}_c - z\bar{\sigma}, \bar{b}_c + z\bar{\sigma}]$	**Pred**	$[\bar{i} - z\bar{\sigma}, \bar{i} + z\bar{\sigma}]$	**Pred**
RF	-663.0	[-437.3, -436.7]	-437.0	[36.86, 36.79]	36.83
CatBoost	-665.4	[-435.2, -434.6]	-434.9	[37.12, 37.05]	37.09
XGBoost	-632.6	[-446.7, -443.9]	-445.3	[10.69, 10.6]	10.65
LightGBM	-664.2	[-435.7, -435.0]	-435.4	[36.93, 36.86]	36.89
ANN	-661.4	[-274.9, -206.1]	-240.5	[8.641, 4.182]	6.332
Mean ML	≈ -661	N/A	≈ -399	N/A	≈ 26
Exp. data	≈ -680	[-367.5, -365.8]	≈ -367	[36.04, 35.78]	≈ 36
PE [%]	≈ 3	N/A	≈ -9	N/A	≈ -29

Table 4.3: Polarization curve features at pH 6.6

	E_{corr} [mV]	$\mathbf{E_{pit}}$ [mV]	Tafel slope b_c with 99% CI [mV/dec]		i_{corr} with 99% CI [µA/cm^2]	
			$[\bar{b}_c - z\bar{\sigma}, \bar{b}_c + z\bar{\sigma}]$	**Pred**	$[\bar{i} - z\bar{\sigma}, \bar{i} + z\bar{\sigma}]$	**Pred**
RF	-785.4	≈ -650	[-414.1, -411.7]	-412.9	[1.045, 1.037]	1.041
CatBoost	-785.9	≈ -650	[414.6, -412.0]	-413.3	[1.050, 1.042]	1.046
XGBoost	-785.4	≈ -650	[-413.5, -410.9]	-412.2	[1.043, 1.034]	1.038
LightGBM	-786.5	≈ -650	[-413.5, -410.8]	-412.1	[1.049, 1.041]	1.045
ANN	-781.3	N/A	[-431.3, -415.7]	-423.5	[1.788, 1.706]	1.747
Mean ML	≈ -789	≈ -650	N/A	≈ -415	N/A	≈ 1.2
Exp. data	≈ -809	≈ -650	[-417.8, -414.7]	≈ -416	[1.106, 1.096]	≈ 1.1
PE [%]	≈ 3	0	N/A	0.3	N/A	≈ 8

Table 4.4: Polarization curve features at pH 7.4

	E_{corr} [mV]	Tafel slope b_c with 99% CI [mV/dec]		i_{corr} with 99% CI [µA/cm^2]	
		$[\bar{b_c} - z\bar{\sigma}, \bar{b_c} + z\bar{\sigma}]$	Pred	$[\bar{i} - z\bar{\sigma}, \bar{i} + z\bar{\sigma}]$	Pred
RF	-865.8	[-253.1, -252.0]	-252.5	[0.2933, 0.2914]	0.2924
CatBoost	-868.6	[-248.1, -240.6]	-244.4	[0.2924, 0.2788]	0.2856
XGBoost	-865.9	[-256.8, -251.7]	-254.3	[0.3009, 0.292]	0.2965
LightGBM	-822.0	[-415.4, -409.7]	-412.5	[0.698, 0.688]	0.6939
ANN	-825.2	[-570.5, -559.5]	-565.0	[0.757, 0.7458]	0.7514
Mean ML	≈ -829	N/A	≈ -346	N/A	≈ 0.46
Exp. data	≈ -726	[-521.4, -519.3]	≈ -520	[0.6460, 0.6433]	≈ 0.65
PE [%]	≈ -14	N/A	≈ 34	N/A	≈ -28

Table 4.5: Polarization curve features at pH 11.6

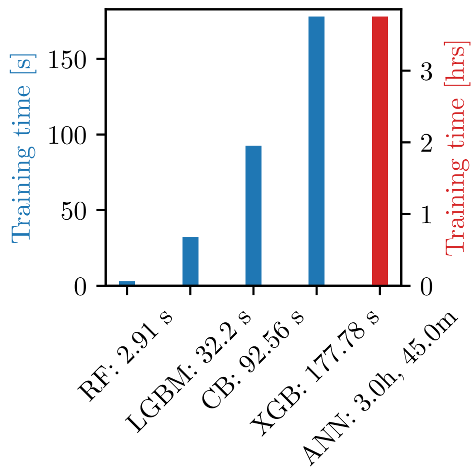
	E_{corr} [mV]	Tafel slope b_c with 99% CI [mV/dec]		i_{corr} with 99% CI [mA/cm^2]	
		$[\bar{b_c} - z\bar{\sigma}, \bar{b_c} + z\bar{\sigma}]$	Pred	$[\bar{i} - z\bar{\sigma}, \bar{i} + z\bar{\sigma}]$	Pred
RF	-1492.6	[-553.0, -548.0]	-550.5	[1.137, 1.126]	1.131
CatBoost	-1393.8	[-525.5, -516.7]	-521.1	[0.6637, 0.6485]	0.6561
XGBoost	-1492.8	[-553.0, -548.1]	-550.6	[1.138, 1.126]	1.132
LightGBM	-1392.6	[-517.7, -509.4]	-513.6	[0.6485, 0.6339]	0.6413
ANN	-1460.1	[-616.7, -610.9]	-613.8	[1.010, 1.000]	1.005
Mean ML	≈ -1450	N/A	≈ -550	N/A	≈ 0.91
Exp. data	≈ -1470	[-518.1, -512.2]	≈ -515	[0.8474, 0.8357]	≈ 0.84
PE [%]	≈ 1	N/A	≈ -7	N/A	≈ 9

Table 4.6: The average MAPE across all test data (pH) of the estimated polarization curve features

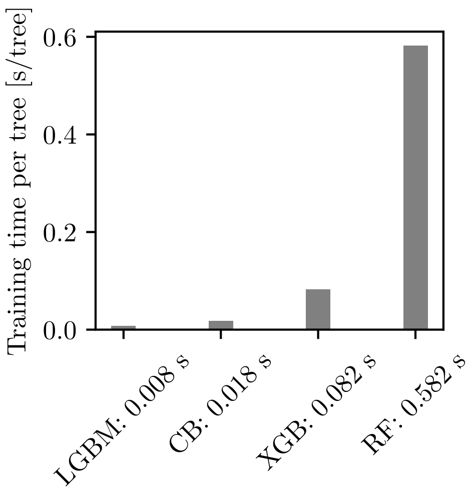
	E_{corr} vs SCE	b_c	i_{corr}
Mean MAPE [%]	5	13	19

4.3 Training time

Figure 4.7a shows the total training time of each optimized model. In Figure 4.7b, training times per iteration prior to termination for the DT algorithms is depicted. RF demonstrated fastest overall training time, while LightGBM and CatBoost proved superior performance in terms of time per iteration/tree.



(a) Training time for all optimized models with the DT algorithms and ANN marked in blue (primary y-axis) and red (secondary y-axis), respectively.



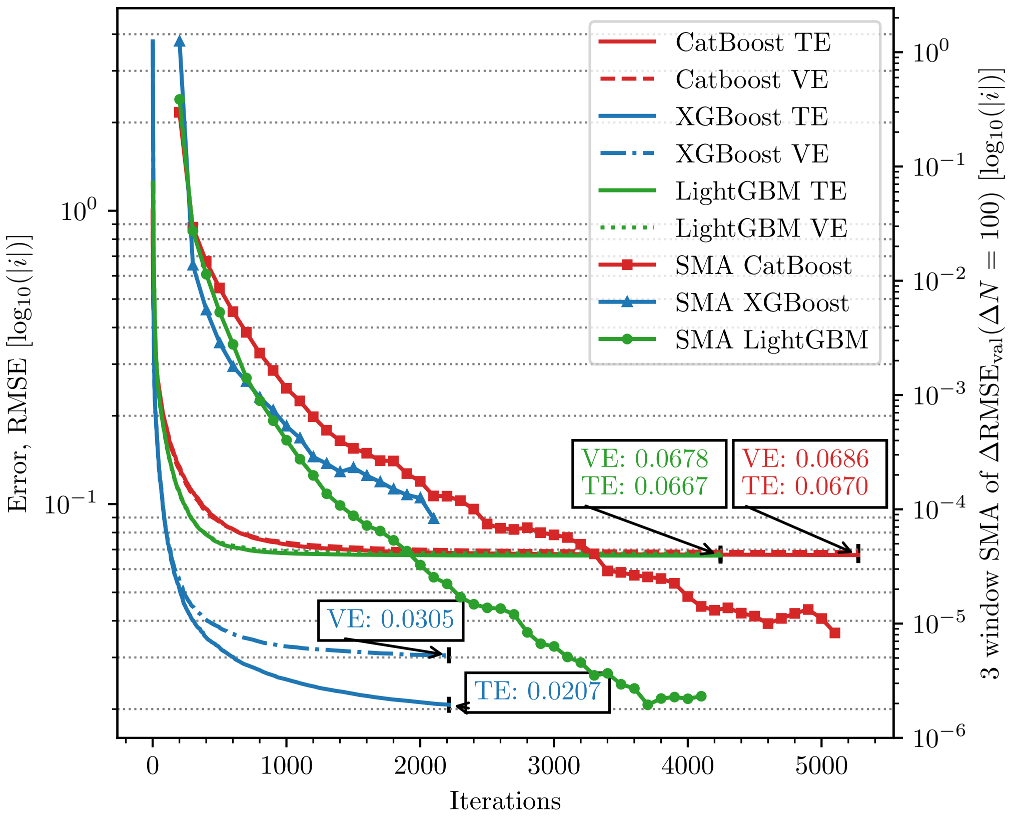
(b) Training time per tree for DTs

Figure 4.7: Training times for all models (left, 4.7a) and training time per iteration/tree for the GBDT algorithms (right, 4.7b).

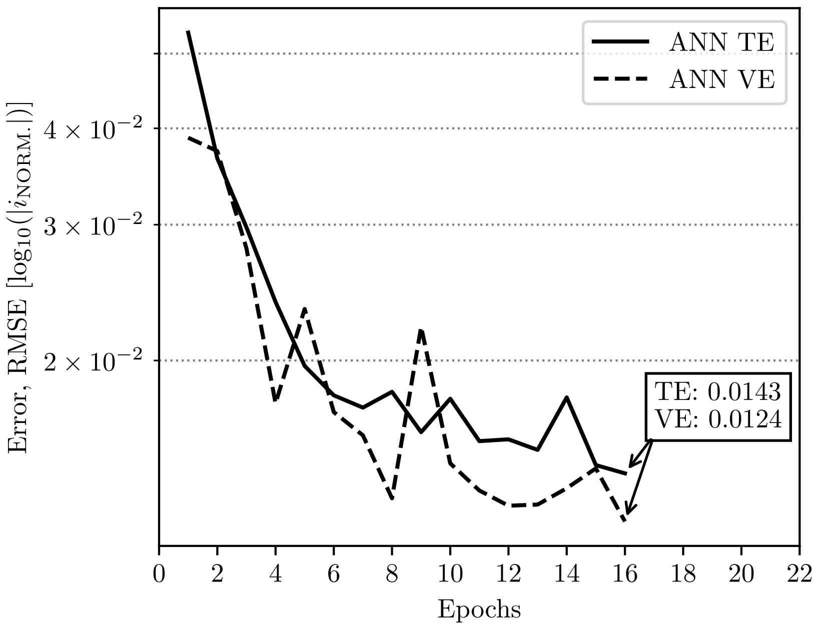
4.4 Learning curves

Figure 4.8a displays the learning curves of the GBDT algorithms with vertical black bars marking the iteration where early stopping was triggered. The Training Error (TE) and Validation Error (VE) are given in RMSE, as RMSE was set as the evalution metric for all algorithms. Together with the learning curves, the 3 window SMA of the validation error gradient over 100 iterations for each GBDT algorithm is provided to display the general trend. It can be observed that the error decreased substantially for smaller number of iterations, while converging to a lower limit for higher iterations, with XGBoost showing enhanced ability to adapt to training data.

In Figure 4.8b, the learning curve of the optimal ANN model is presented. The error is given in RMSE of the predicted normalized logarithmic current density. At the 16th epoch, training was terminated as a new global error minima was not achieved over the early stopping criteria of 4. As seen from the figure, the validation error (VE) did however find a new error low at the epoch of termination, which indicates that training was stopped prematurely.



(a) Training and validation error measured in RMSE for GBDT algorithms (primary y-axis). The 3 window SMA of the error gradient over 100 iterations is also provided (secondary y-axis). TE and VE are short for Training Error and Validation Error, respectively.



(b) Training and validation error for optimal ANN. TL and VL are short for Training Error and Validation Error, respectively.

Figure 4.8: Learning curves GBDTs and ANN

4.5 Error on known data compared to unseen data

Figure 4.9 presents the ratio of the average error across all pH levels for LightGBM, CatBoost, and XGBoost to the validation error (VE) at the point of training termination. The large contrast between errors on known and unknown data suggests presence of unexpected behavior within the training and validation sets. Solely based on Figure 4.9, one might infer overfitting to the training data as a possible explanation. However, overfitting is deemed less probable. This hypothesis is discussed further in Section 5.3.

Data from RF and ANN is not included in Figure 4.9 as they were not accessible. RF does not utilize validation data, and thus the comparison with the GBDT algorithms would indeed be incorrect. In the case of ANN, the target output was the normalized value of the logarithmic current density. Consequently, the error at termination in relation to the non-normalized value is not available. This would require training on the non-normalized value, which would not provide an accurate error conversion, as training on non-normalized data is expected to enhance the training and validation error.

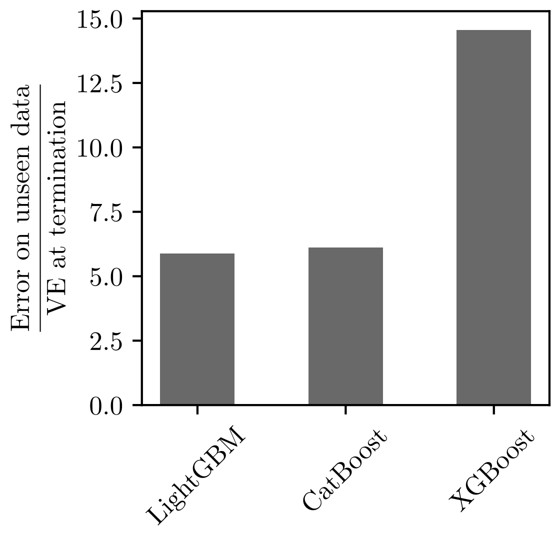


Figure 4.9: The average error across all pH for LightGBM, CatBoost and XGBoost divided by the validation error (VE) at termination of training. The fraction is dimentionless, as both errors are measured in RMSE.

4.6 Analysis of empirical data

To achieve a comprehensive understanding of the machine learning results, a thorough analysis of the training data is executed. Consequently, exploring the consistencies and inconsistencies in both E_{corr} and empirical polarization curves is essential.

4.6.1 Corrosion potential measurements

Typically, the corrosion potential E_{corr} behaved relatively stagnant during the last third of the measurement period, as depicted in Figure D2. This observation is in line with findings by B. Zaid et. al. considering AA6061[4], where the OCP in acidic, neutral and alkaline media reached stability after an approximate duration of one hour. Due to the stagnant behaviour, a normality test was assessed to investigate the assumption of normally distributed residuals from the mean. The results are shown in Figure C1a and C1b. The results imply that the assumption of normally distributed residuals from the mean is reasonable. Thus, it was concluded that E_{corr} on a general basis reached stability before polarization was initialized.

However, relying solely on the E_{corr} residuals to describe the trends is inadequate. This is because certain measurements exhibit clear deviations from normality, observed by distinct change in potential in either direction. Such trends can e.g. be observed in Figure F3, F7, F11, and F12 for pH values of 4.2, 7.2, 10.4, and 10.8, respectively. These polarization curves show clear deviation from normality by exhibiting clear positive or negative E_{corr} gradients during the last third of the measurement period. Thus, potential inaccuracies in the measured E_{corr} is plausible as a stable state was not reached.

Figure D1a show that E_{corr} fluctuated especially to a large extent in the passive region, possibly due to meta-stable pitting induced by chloride attacks, depassivating the surface. The fluctuating characteristic of E_{corr} had a noticeable impact on the prediction error of E_{corr} at pH = 7.4, as outlined in Figure 4.1c. At this pH, the mean prediction error of E_{corr} was approximately 14%. A likely explanation for this error is the unstable E_{corr} seen in Figure F7 and F8 at pH 7.2, 7.4 and 7.6. At t = 3600 seconds, the E_{corr} recorded at pH 7.4 exhibited a significantly higher value compared to pH 7.2 and pH 7.6. As a result, the initial E_{corr} during polarization showed high variation at this pH range. This is observed in Figure E14 and E15. Justifying these variations is difficult, however a case could be made that these samples possibly demonstrated different surface conditions, a consequence of either or both re-usage of samples or scratches obtained through e.g. physical transport. On the other hand, in acidic media at pH < 3.6, as depicted in Figure F1 and F2, the E_{corr} measurements displayed a relatively stagnant behavior within the range of -0.65 to -0.70. Consequently, the average error of E_{corr} at pH 2.2 was only approximately 3%.

In certain cases, E_{corr} underwent large changes during cathodic polarization, as outlined in Figure D1c and Table A8. at pH 4.4, the value of ΔE_{corr} was observed to be -0.197 V. As depicted in Figure D1c, this was however untypical behaviour. These observations are expected to have a negative impact on the predictive capability. Therefore, further examination of the empirical polarization curves is necessary.

4.6.2 Polarization curves

As discussed, this work attempted to predict corrosion potential E_{corr}, cathodic Tafel slopes b_{c}, corrosion current density i_{corr}, and pitting potential E_{pit}. For simplicity, the E_{corr} at time = 0 seconds in the polarization is denoted $E_{\text{corr}_{\text{t0}}}$, while E_{corr} post cathodic scan is abbreviated $E_{\text{corr}_{\text{th}}}$.

The findings presented in Appendix E indicate that the Butler-Volmer (BV) equations do not accurately describe reality. The BV equation considering mass transport (Eq. 2.11), with the theoretical polarization curve in Figure 2.4, demonstrates in some instances a reasonable depiction of the kinetics, particularly observed at pH 4.0 (E6), where the cathodic current density is limited by insufficient oxygen transport to the electrode surface, resulting in an infinitely negative cathodic Tafel slope b_{c}. On the other hand, BV_{diff} fails to adequately represent the cathodic branch in numerous cases where a diffusive current limitation is absent. Furthermore, BV_{diff} obviously falls short in determining the pitting potential and regions where meta-stable pitting may occur. Consequently, these observations emphasize the utility of machine learning to estimate important features of polarization curves, such as E_{corr}, b_{c}, and E_{pit}.

Empirical values of E_{corr} aligned well with theoretical expectations ($\approx$ -0.75 V vs SCE), as seen in Table 4.2, 4.3, 4.4 and 4.5. In Figure D1, E_{corr} is depicted as a function of pH, both prior to polarization (Figure D1a) and post cathodic scan (Figure D1b). The measured $\text{E}_{\text{corr}_{\text{t0}}}$ is generally in line with $\text{E}_{\text{corr}_{\text{th}}}$, with an average difference of only 7 mV. However, a notable deviation from this trend is observed at pH = 4.4, where the difference between $\text{E}_{\text{corr}_{\text{t0}}}$ and $\text{E}_{\text{corr}_{\text{th}}}$ is nearly -0.2 V, as outlined in Figure D1c and Table A8. Justifying the larger ΔE_{corr} of -0.2 V with certainty is challenging. However, a plausible explanation is impurities on the aluminium alloy surface. Metallic ions such as magnesium could be dissolved and thus act anodic towards aluminium, increasing E_{corr} of aluminium. Additionally, the pH in the solution is changed throughout cathodic polarization as both ORR and HER will change the pH, as seen in the electrochemical reactions in Section 2.1.2. Another interesting observation with respect to ΔE_{corr} in Figure D1c is the delta as a function of pH. The delta was observed significantly larger in strong alkaline media at pH 10-11. However, it is further discovered that the percentage change is relatively independent on pH. Thus, the E_{corr} delta measured in percentage is not generally changing with pH, local deltas are indeed observered, e.g. at pH 4.4 as discussed above.

In strong alkaline media, E_{corr} was substantially decreased. In Table A8, it can be discerned that E_{corr} was $\approx$ -1.5 V vs SCE at pH 12, surpassing the E_{corr} of -1.4 V vs SCE in 3.5 wt% ($\approx$ 0.6 mol/L) chloride[4]. The slightly higher E_{corr} of -1.5 V vs SCE obtained in this work is difficult to clarify, as E_{corr} was suggested relatively independent of chloride content. The more negative E_{corr} of -1.5 V vs SCE does however indicate that the extruded AA6060 aluminium alloy apploy in this work is more likely to corrode in strong alkaline media compared to the AA6061 alloy investigated in the work by B. Zaid. et al.[4].

The large negative Tafel slopes in Tables 4.2 - 4.5 deviate significantly from theoretically expected ORR Tafel slopes of -60 to -120. The estimated Tafel slope of up to -520.3 mV/decade at pH = 7.4 indicates unexpected surface behaviour as neither of the empirical polarization curves at test pHs (2.2., 6.6, 7.4 and 11.6) exhibit clear limiting currents. Conversely, at e.g. pH 4.0 (Figure E6) a limiting current i_{lim} is detected initially for potentials closer to E_{corr} at the cathodic branch. Furthermore, the cathodic Tafel slopes were seemingly largely influenced by the electrode surface condition, which plausibly was fairly inconsistent across the samples.

Moreover, HER is not explicitly found in test data at neither pH 2.2, 6.6, nor 7.4 (Figure E1, E12 and E14). This is because ORR is more dominant at higher potentials for negative overpotentials. Nevertheless, there are minor indications of HER at pH 6.6 and pH 7.4 at -1.1 and -1.05 V vs SCE, as the Tafel slopes become less steep, as outlined in Section 4.2. At pH 11.6 (Figure E25), distinguishing between ORR and HER is obviously challenging, as both reactions are thermodynamically feasible. In general, the empirically observed high cathodic Tafel slopes verify the importance of utilizing machine learning, as analytically interpreting Tafel slopes proves to be a challenging task due to complex kinetics of ORR and HER.

In this work, the corrosion current density i_{corr} was empirically estimated by use of the corrosion potential E_{corr} and the cathodic Tafel line, as previously discussed at the beginning of this section. The anodic branch typically suffered from pitting in acidic and neutral environments and was therefor not utilized to find i_{corr}. Additionally, in highly alkaline environments (pH $\geq$ 10.4), an anodic limiting current was observed. A limiting current may have occurred as a result of enhanced concentration of hydroxide ions (through ORR and HER in alkaline media, Eq.2.4 and Eq. 2.6), causing oxidated aluminum to react with these hydroxide ions to form aluminium hydroxide. Anodic current limitations are also observed at pH 11.6 (Figure E25), which also exhibit untypical behaviour by shifts to lower currents. The reason for this is difficult to interpret. However, a plausible explanation could be the dealloying of aluminium induced by the high pH. Consequently, the current observed may indeed represent the passive current. A similar shift in anodic current density to obtain the passive current was empirically found by B. Zaid et al. in their study of AA6061[4].

Tables 4.2 to 4.5 provide empirically estimated values for the corrosion current density i_{corr}. The observed results align well with theoretical expectations[4]. It is observed that in strong alkaline environments, the corrosion current densities increase significantly (approximately 0.84 mA/cm^2 at pH 11.6) compared to both acidic and neutral conditions. Furthermore, in strong acidic electrolytes (pH = 2.2), the corrosion current density is considerably higher than in neutral media (pH = 6.6 and pH = 7.4). The empirically estimated i_{corr} of 36 µA/cm^2 at Ph 2.2 is approximately 36 times larger than at pH = 6.6 and 56 times larger than at pH = 7.4, respectively. The elevated corrosion current densities in acidic and alkaline media are consequences of the larger cathodic currents. The current densities at negative overpotentials align well with theoretical expectations [4], where the highest current densities were observed in strong alkaline media, surpassing 1 mA/cm^2. The enhanced current density could be justified by increased HER and ORR reaction rates caused by the lower E_{corr}. Additionally, in strong alkaline electrolytes, the depassivated surface area is expected to increase due to enhanced dissolution rate of aluminium through Eq. 2.2, leaving a larger available electrode surface area unprotected. Thus, increasing the surface concentration of oxidized species in Eq. 2.8 and consequently the net current density.

The pitting potential E_{pit}, is most visible in test data at pH = 6.6 (Figure E12) at ≈ -0.65 V vs SCE. E_{pit} is not clearly visible at pH = 2.2, as shown in Figure E1, since E_{corr} is ≈ -0.68 V vs SCE, only 60 mV lower than E_{pit}. At pH = 7.4, E_{pit} is not visible as E_{corr} was ≈ -0.726 V vs SCE, as obtained from Table A8. At this potential, aluminium oxidation is likely the main anodic reaction.

The average E_{pit}, as found in Figure D1d, was determined by plotting E_{pit} against pH, if E_{pit} was clearly visible. Aligning with theory, E_{pit} did not exhibit any clear trend with pH and stayed relatively constant with an average of $E_{\mathrm{pit,\ avg}}$ = -0.66 V vs SCE, as depicted in Figure D1d. The outlier at -0.7 V vs SCE at pH = 5.2 could be explained by distinct surface conditions such as cracks and segregation, propagating pitting. The average of -0.66 V vs SCE is higher than the measured ≈ -0.75 V in a 3.5 wt% (0.6 mol/L) chloride solution and aligns well with expectation that E_{pit} should decrease for larger chloride content [4].

5 Discussion

5.1 Introductory remarks

Generally, the machine learning algorithms performed similarly with an average MAPE of approximately 7% of the logarithmic current density across all pH. RF and LightGBM was the top and worst performer with MAPE of 6.06% and 9.67%, respectively, as seen in Table 4.1. The fact that the GBDT algorithms did not outperform RF contradicts with theory, and can be explained by the noisy data set as a significant larger error was observed on unseen data, as seen in Figure 4.9. This is further explained in the coming paragraphs. The highest prediction error was observed at pH 7.4, as the polarization curves around this pH varied significantly. The corrosion potential E_{corr}, cathodic Tafel slope b_{c} and corrosion current density i_{corr} were estimated with an average MAPE across all pH of 5.23%, 12.3% and 18.2%, respectively, as found in Table 4.6. The estimated cathodic Tafel slopes b_{c} and corrosion current densities i_{corr} exhibit small regression uncertainty, as seen in the narrow 99% CIs. In addition to accurately predict the magnitudes of E_{corr}, b_{c} and i_{corr}, the algorithms were highly capable of capturing important pH dependent transitions in the polarization curves, such as the pitting potential E_{pit}. In addition, XGBoost showed promising potential of locating meta-stable pitting.

The fastest algorithm was RF, using solely 2.91 seconds to train, as seen in Figure 4.7a. This can be attributed to the fact that only five trees were utilized in the forest, as seen in Table A5. LightGBM was the most efficient algorithm per tree/iteration, using only 0.008 seconds, as depicted in Figure 4.7b. Moreover, ANN was the slowest algorithm using 3 hours and 45 minutes to train in total. A lot of effort was put into hyperparameter tuning of ANN with the result of being the average performer across all algorithms. The applicability of ANN, for what was proven a data set with a large degree of unknown behaviour, is therefor questioned as the trade-off between time and accuracy was poor relative to the other algorithms. A brief assessment of the proposed algorithms to employ moving forward is provided in the conclusive remarks in Section 6.

It was observed that the algorithms typically adapted well to training data, but struggled to predict unseen data with same accuracy, as depicted in Figure 4.9. As early stopping and built-in L1 and L2 regularization were utilized for the iterative algorithms, and because only 5 iterations were applied in RF, overfitting to training data was not considered a likely explanation for any algorithm. A more probable explanation is non-optimal model architectures in combination with large amounts of unknown behaviour in the data set. The observed unknown behaviour may be attributed to the limited use of pH as the sole input feature. In this work, characteristics such as cracks or segregation of certain alloy elements are not accounted for. Additionally, the re-usage of samples across experiments has indeed caused changed surface structure, which is not accounted for in the data set. It was however observed that pH did indeed influence the predicted results, as the average pH feature importance (for GBDTs) was 36% (potential E: 64%), as shown in Figure B2.

The feature importances in the ANN was not obtained as `Keras` does not provide feature importance data.

Section 4.6 provided a comprehensive analysis of the empirical data. In the following paragraphs, the feature importance of E and pH, the outcomes of performed hyperparameter tuning, and the obtained learning curves is discussed. Additionally, individual algorithms are examined at each pH.

5.2 Feature importances

Examination of Figure B2 revealed that pH was, on average, less important (36%) during the training process compared to potential (64%). The purpose of calculating the feature importance is however to get a more comprehensive understanding of how important pH is compared to the potential E for the observed current density. Interestingly, RF, with its lower number of trees, predicted the lowest importance for pH, whereas the GBDT algorithms estimated a higher average pH importance. As explained in section 2.2.3, a larger number of iterations is typically required to accurately determine feature importances. Thus, it is reasonable to assign greater weight to the feature importances of the GBDT algorithms presented in Figure B2. However, Figure B1 challenges this theory, as the feature importances calculated by RF did not undergo significant changes with the number of trees in the forest. The feature importances of ANN was not provided as `Keras` does not support this feature.

It is assumed that the importance of pH and E may vary significantly to those estimated by the DT algorithms due to the different model architectures. In a research conducted by A. S. Dyer et. al[56], the feature importances in XGBoost and ANN was examined for a task of predicting the remaining useful life of offshore operating platforms. The study revealed that a particular parameter held over 80% importance in XGBoost's prediction model, while its significance dropped to less than 30% within the ANN framework. A similar result was obtained by G. K.F. Tso et al.[57] on electricity energy consumption, where high contrast between the feature importances of ANN and DT were detected. Therefore, it is reasonable to anticipate contrast between feature importance of pH and potential E in this work. However, it is concluded that pH did indeed influence the optimal node splitting criteria, and thus, the prediction of the logarithmic current density $\log_{10}(|i|)$.

5.3 Hyperparameter tuning and learning curves

Generally, the effect of hyperparameter tuning in this work is questioned. CatBoost had the second lowest MAPE of 6.41% without any tuning. That being said, the default learning rate of 0.3 was substituted with 0.35 as a learning rate of 0.3 made the model incapable of converging over a period of 10000 iterations. This may have affected the prediction ability of CatBoost negatively. A comprehensive research conducted by C.

Bentéjac et al.[58] revealed that tuning CatBoost hyperparameters had minimal effect on the result of various classification tasks. This is attributed to the change in learning rate throughout the iterative training process. Thus, the optimal value is expected to be close to the default. Hence, the applied value of 0.35 compared to default (0.3) in this work may have lead to unwanted higher error. It is however noted that the study by C. Bentéjac et al. considered decision trees for classification. Nevertheless, the architecture is similar to regression trees[59]. Thus, it is assumed that the results conducted by C. Bentéjac et al. can be utilized for CatBoost regression as well.

In contrast to CatBoost, the RF, LightGBM, and ANN models each underwent varying levels of hyperparameter tuning. The hyperparameter tuning of ANN was indeed the most time consuming tuning process, requiring days to tune. Several rounds of tuning were applied by filtering down the search space after each tuning session. The complete search spaces with the optimized hyperparameters are found in Table A1. Additionally, the complete csv files containing the validation loss of each hyperparameter combination for each trial can be found in *tuning_results_ANN* on Github[1]. In the penultimate tuning round, all individual hyperparameter's effect on the training error was thoroughly studied by plotting the error as a function of hyperparameter combinations with only one distinguishing hyperparameter. From these results, as depicted in Figure A1, it was concluded that more neurons and HLs were preferred, as the error decreased for larger values. The fact that more neurons and HLs were preferred indicates a complex data set. Additionally, the results indicated that both a decrease in batch size and the utilization of MSE as loss function were beneficial. However, this result was not employed into further tuning, as the number of samples utilized to obtain these plot were assumed insufficient to make any conclusions on the effect of batch size and loss function.

After the penultimate tuning session, smaller neurons (50 and 100) were removed and 1200 neurons was added in the search space. Additionally, 4 HL was removed while 9 was added. Unfortunately, this did not improve the results in the ultimate tuning session relative to the penultimate, seen in Table A2, where the validation loss at the penultimate round (N -1) was lower than the last tuning session (N), RMSE of $\log_{10}|i|$ at 0.010337 compared to 0.01149, respectively. Thus, the hyperparameter combination of the best combination in the penultimate tuning experiment was utilized for training.

The ANN learning curve can be found in Figure 4.8b. The validation loss of 0.0124 represents the RMSE of the normalized absolute value of the logarithmic current density. The learning curve does not indicate obvious signs of underfitting nor overfitting. This assumption is however contradicted in Figure 4.6, where the ANN model seems to underfit to data by exhibiting untypical polarization curve shapes. It is likely that the found hyperparameter combination through tuning was indeed not the optimal combination, or that the early stopping criteria of 4 iterations were too low, resulting in a pre-mature termination of training.

From the ANN learning curve, it is further detected that the validation error is smaller than the training error. This is untypical behaviour, as the model learns from the training set, and should thus perform better on training data. This implies large variations in data set and insufficient description of the training data by the input features, potential E and pH[60]. This assumption is reinforced by the results from the learning curve, where the validation error demonstrates larger fluctuations. Another observation is that the validation loss of approximately 0.0124 is substantially higher than the validation loss obtained through tuning (≈ 0.0103). This is surprising, as both these models utilized the same hyperparameter combination. A shift in error from the tuning phase to the training phase could generally be explained by different employed training and validation splits, as this would change the data the model trains and validates on. However, this was not the circumstance in this study, as the same split was utilized for both tuning and final training.

A probable explanation for the observed error difference could be that the weights and biases, assigned prior to and during training, varied between the tuning and final training processes. Consequently, the two training processes may have terminated at a different epoch. Such detailed tuning history is unfortunately not available information in this work. From the learning curve of ANN, it can be observed that this is not an unlikely scenario as the validation loss fluctuated significantly and may have been terminated prematurely due to the small applied early stopping criteria of 4. Furthermore, if this is true, the ANN has underfit to data and has room to learn if a larger early stopping criteria is applied. This could probably be advantageous given the fluctuations in validation error indicating much unknown behaviour (treated as randomness) in the data set.

The hyperparameter tuning processes for LightGBM and RF were significantly more efficient compared to the ANN, in terms of execution time. Specifically, LightGBM and RF required only hours and minutes, respectively, while the ANN tuning process was performed over several days. The RF was tuned manually by testing only a few combinations of various number of trees in the forest (*n_estimators*) and maximum features (*max_features*) to consider at each split, as previously discussed in section 2.2.3. The search space and complete result of the process can be found in Appendix A.3. Interestingly, the training error did not typically decrease for larger iterations, contradicting with theory. This may be explained by large amounts of unknown behaviour (randomness) in traning data, as discussed above. The optimal *max_features* was determined to be the default value 1, indicating that using all available features yielded the best results. This assumption is supported by C. Bentéjac et al. who also identified that the lowest errors were obtained by applying default values in RF[58]. This aligns with the presented theory in section 2.2.3 that RF shows minimal added accuracy post tuning. The optimal value of 1 (using both pH and E) supports the assumption of a complex data set with large unknown behaviour. As the data set already contained a large degree of randomness, further added randomness by utilizing only one feature (*max_features* = 0.3) was not favored.

Furthermore, LightGBM was tuned as this model proved the lowest training time per iteration, aligning well with the expectations of computational efficiency. The goal of tuning LightGBM in this work was to find the slowest learner in the attempt of locating the global error minimum. However, this was unsuccessful, as the hyperparameter combination working as the slowest learner showed a higher error relative to default values. The complete search space and tuning process can be found in Table 2.3 and A3.

As recommended by the LightGBM developers and discussed in Section 2.2.5, *dart*, a large number of leaves (*num_leaves*), and a low learning rate (0.001) was the hyperparameters under consideration. The slowest learner was the model employing *dart*, *num_leaves* = 61 and with a learning rate of 0.001. As observed in Figure A2, this LightGBM model was unsuccessful as the RMSE was greater than 0.1 for more than 400 iterations, after a total of $\approx$ 71000 iterations, before manual termination was executed. Training had to be terminated manually as *GridSearchCV* in `Scikit-learn` does not provide this feature. The model was additionally way slower compared to default values, which was terminated by the early stopping criterion after 4245 iterations. Utilizing *dart* together with 61 maximum leaves in combination with a learning rate of 0.001 seems to have lead to excessively regularization limiting the model to adapt to training data. The model does however show a large degree of generalization on the validation data set, proved by the small delta between training and validation error.

Upon examination of the GBDT learning curves in Figure 4.8a, a noticeable trend is observed. XGBoost significantly outperformed both CatBoost and LightGBM in terms of minimized training and validation errors. Conversely, when evaluating the performance on unseen data, CatBoost demonstrated lowest RMSE of the GBDT algorithms (6.41%), as shown in Table 4.1. The generalization ability of CatBoost aligns well with expectations provided in Section 2.2.6.

By considering XGBoost as an example, the validation loss at termination of training was 0.0305, as depicted in Figure 4.8a. Contrary, the average RMSE across all pH was approximately 0.444, by employing the RMSE of XGBoost in Table 4.1. Thus, the RMSE on unseen data is 14.56 times higher than the validation RMSE obtained through training, as depicted in Figure 4.9. This indicates decent ability to adapt to the training set, while struggling with generalizing on unseen data. Furthermore, from the learning curves of the GBDT algorithms (Figure 4.8a) and ANN (Figure 4.8b), it is not detected any clear signs of neither underfitting nor overfitting, which indicates that the proper hyperparameter combinations are yet to be found. It is detected that all GBDT algorithms have relatively small loss gradients for > 2000 iterations. It is thus possible that the error minima are not reached, suggesting that decreasing learning rate and adding more complexity/greediness to the trees could be beneficial. By utilizing more leaves in the trees, while decreasing the maximum number of numerical values in the terminal nodes, the algorithms may have adapted slightly better to training data, though require higher training time.

Furthermore, the predicted polarization curves and lack of smoothness by ANN indicate underfitting to training data. As 7 HLs with 900 neurons in total were applied, outperforming both 8 and 9 HLs combined larger number of neurons (Table A2), underfitting is suprising. To tackle this problem, the number of epochs before early stopping is triggered could be increased. From the learning curve in Figure 4.8b, the validation error exhibits large spikes and was finally stopped at 16 epochs. Increasing the patience parameter to e.g. 6 could provide lower training and validation error, but on the other hand increase the chances of overfitting to data.

5.4 Evaluation of polarization curve at pH 2.2

At pH = 2.2, the average MAPE was approximately 5%. It is observed that XGBoost and ANN exhibit higher error compared to the other algorithms. While all algorithms provide comparable anodic branchs as the empirical output, they typically struggle to a greater extent with accurately predicting the cathodic region of the polarization curve. This observation is particularly noticeable for XGBoost, which estimates a current density below 10^{-4} A/cm^2 at the potential minimum, whereas the empirical current density exceeds $3{\cdot}10^{-4}$ A/cm^2 at the same potential.

The high XGBoost error is difficult to justify as XGBoost provides the second lowest average error across all test data, as seen in Table 4.1. However, upon examination of the empirical data for pHs below and above pH 2.2, it could be argued that the algorithms weighted training data differently during training. XGBoost, which exhibit a distinctive polarization curve shape relative to the other algorithms (Figure 4.1a), seems to have weighted pH 2.4 heavier during training, while the other algorithms except ANN plausibly assigned more importance to pH 2.0. The underlying cause of this is difficult to clarify. However, it is possible that XGBoost utilizing pH 2.4 over pH 2.0 is simply a result of random chance, considering the nature of decision trees. On the other hand, XGBoost exhibits a distinguishing learning curve, while the shapes of LightGBM and CatBoost are comparable. Therefore, the contrast between XGBoost and LightGBM and CatBoost can plausibly be explained by the different tree structures. The level-wise growth is only found in XGBoost, while both CatBoost and LightGBM utilize a leaf-wise growth strategy.

A distinct polarization curve of XGBoost compared to CatBoost and LightGBM is also observed at pH 11.6 (Figure 4.1d), which indicates that the contrasts in tree growing structure have indeed affected the results at certain pH. The level- and leaf-wise methods make the algorithms choose samples differently. All nodes are split in the level-wise structure, while in the leaf-wise structure only the node with the most negative error gradient are built on further[21]. Thus, the employed samples for training will also vary significantly, which is supported in the different feature importance outputs in Figure B2. The feature importances are explained more comprehensively in Section 5.2. Another interesting observation is that RF predicted a similar polarization curve as both CatBoost and LightGBM. RF, which is solely an averaging ensemble DT algorithm, splits by randomly selecting features and therefor without searching for optimal splits. Thus, it presents a compelling observation that the algorithms most susceptible to overfitting (CatBoost and LightGBM) are indeed those that predict the same polarization curve shape as RF, which is notably less prone to overfitting. The combination of generalization and accuracy from these algorithms imply that RF, CatBoost and LightGBM are indeed preferable in strong acidic media. Furthermore, ANN indicates signs of underfitting as the cathodic branch exhibits unknown behaviour not seen in training data. This will be further elaborated at the end of this section, as underfitting is observed across all pH for ANN.

In terms of predicting E_{corr}, i_{corr} and b_c at pH 2.2, it is observed that CatBoost yields the most accurate values for the corrosion potential E_{corr} and b_{c}. On the other hand, RF exhibits the lowest error for the corrosion current density i_{corr}. The average E_{corr} value of -661 mV deviates by approximately 3% from the empirically determined E_{corr} value of -680 mV. It is worth noting that XGBoost predicts the most positive E_{corr}, which may be attributed to the higher E_{corr} observed at pH 2.4 compared to pH 2.0, as shown in Table A8.

Despite XGBoost underperforming relative to the other algorithms, the cathodic Tafel slope of approximately -445 mV/decade is only 21% more negative compared to the empirically determined cathodic Tafel slope of approximately -367 mV/decade. As depicted in Figure 4.5a, this is due to utilizing the Tafel slope at lower potentials instead of the diffusion limited current region at higher potentials. Furthermore, the mean approxmated cathodic Tafel slope b_{c} across all models was was -399 mV/decade, 9% higher than the empirical b_{c} of -367 mV/decade. By examining Figure 4.6a, it becomes evident that the ANN poorly estimates the cathodic Tafel slope b_{c}. This can be justified by the distinctive change in current at approximately -0.9 V vs SCE. Moreover, the validity of the ANN cathdic Tafel slope is questioned, as the 99% CI indicates large regression uncertainty.

Due to XGBoost's significantly lower prediction of the cathodic current, the predicted corrosion current density i_{corr} is consequently much lower than the empirical value of 36 µA/cm^2. In contrast, RF performs reasonably well in estimating the empirical i_{corr} with a MAPE of only 2.6%.

5.5 Evaluation of polarization curve at pH 6.6

At pH 6.6, the average MAPE was approximately 3%, representing the lowest error across all pH by significant margin. The performance of the ANN was inferior to that of the DT algorithms, which showed comparable performance. XGBoost provided the lowest error with MAPE of 2.4%. At pH 6.6, the pitting potential E_{pit}, at $\approx$ -650 mV, was visible and predicted to detail by all algorithms. This is however not overly surprising as E_{pit} stayed relatively constant over the pH range, shown in Figure D1d. The mentioned results is indeed relevant to practical applications, as neutral pH is commonly encountered.

From the comparable polarization curve shapes, seen in Figure 4.1b, it can be deduced that the different tree growing methods have not affected the optimal splitting criteria during training. Clearly, the polarization curve of pH 6.4 was weighted heavier during training, as the corrosion potential E_{corr} at pH 6.4 (-785 mV, Table A8) is indeed similar to the mean predicted E_{corr} of -789 mV (Table 4.3). The predicted E_{corr} does however differ from the empirical determined E_{corr} of -809 mV. The error in E_{corr} can be attributed to the larger spread in E_{corr} in the passive region, as observed from Figure D1b.

Furthermore, the average predicted cathodic Tafel slope b_{c} at a pH of 6.6 was -415 mV/decade, which is only a marginal 0.3% more positive than the empirically derived b_{c}. Yet, one could contend that the mean Tafel slope should be calculated by excluding the prediction made by the ANN, as no clear Tafel region is located due to the untypical shift in current density at approximately -1.1 V vs SCE, increasing the upper and lower band of the 99% CI. Excluding ANN b_{c} would however unfortunately result in a larger discrepancy between the empirical b_{c} and the mean b_{c} predicted by the machine learning algorithms.

In Table A8, the Tafel slopes at pH 6.4 and 6.8 are given as -384 and -753 mV/decade. As the predicted Tafel slopes at pH 6.6 were indeed similar to that of pH 6.4, the idea that the algorithms utilized pH 6.4 to a greater extent is reinforced. Additionally, it can be shown in Table A8 that the cathodic Tafel slope of pH 6.8 was untypical over the pH range from pH 5.6 to 7.2, which made data samples of pH 6.8 unsuitable as node splitting criteria.

As a result of the precise prediction of the cathodic Tafel slope b_{c}, the estimated corrosion current density i_{corr} also exhibited a high level of accuracy. The mean predicted corrosion current density i_{corr} of 1.2 µA/cm^2, as seen in Table 4.3, was comparable to the empirical value 1.1 µA/cm^2, with a difference of solely 8%.

5.6 Evaluation of polarization curve at pH 7.4

The highest prediction error was observed at pH 7.4, characterized by an average MAPE of approximately 15%, as shown in Table 4.1. The higher error could by justified by the variation in E_{corr} at this pH range, as previously discussed in Section 4.6.1. As shown in Table A8, E_{corr} at pH 7.2 and 7.6 were larger than expected at approximately -820 mV and -866 mV vs SCE, respectively. The predicted E_{corr} was $\approx$ -829 mV vs SCE, and thus within the range of pH 7.2 and 7.6. Conversely, an elevated empirical E_{corr} of $\approx$ -726 mV vs SCE was observed at pH 7.4 due to the high fluctuations in E_{corr} in neutral media, as discussed in Section 4.6.2. The large spread in E_{corr} apparently caused uncertainty the algorithms were not able to predict and thus reinforces the assumption that utilizing solely pH as an input feature (in addition to potential E) is insufficient to accurately describe the polarization curves.

An interesting observation is that RF, CatBoost and XGBoost predict relatively comparable polarization curves at pH 7.4, while LightGBM and ANN exhibit unique shapes. XGBoost allegedly weighted pH 7.8 (Figure E15) to a higher degree compared to the other algorithms as XGBoost predicts meta-stable pitting below E_{pit} (Figure 4.5c). It is also observed that the empirical E_{corr} of $\approx$ -866 mV vs SCE at pH 7.6 (Table A8) is indeed similar to the predicted E_{corr} by XGBoost, CatBoost and RF. Additionally, as RF and CatBoost exhibit relatively similar polarization curve shapes, it is likely that these two algorithms also utilized pH 7.8 more frequently during training, however to a lower degree as no meta-stable pitting is observed in Figure 4.2c (RF) and Figure 4.3c (CatBoost).

An explanation to why solely XGBoost predicted meta-stable pitting is difficult to interpret, but it can be observed that pitting was a typical phenomena in empirical data in the passive region. Consequently, XGBoost provided enhanced generalization at this pH range by predicting meta-stable pitting, while determining E_{corr} by use of pH values close to 7.4. However, it is observed in the empirical data that meta-stable pitting is clearly less visible at pH 7.4 compared to e.g. pH 7.8. CatBoost, with its leaf-wise structure and approximately 3000 additional training iterations compared to XGBoost does not predict meta-stable pitting. This outcome may be attributed to CatBoost's larger tree depth, which emphasizes specific details to a greater extent but potentially overlooks the typical trends if the tree is grown too deep. Surprisingly, Random Forest (RF) also fails to predict meta-stable pitting despite the expectation that utilizing only five trees would lead to better generalization. It appears that RF underfits the data by not capturing the typical behavior of meta-stable pitting. It is however possible that the RF model treated meta-stable pitting as noise.

The learning curves depicted in Figure 4.8a demonstrate that, overall, XGBoost performs better on both training and validation data. It is worth considering that the fact that XGBoost predicts meta-stable pitting while the other decision tree algorithms do not, may be purely a result of random variations during the data sampling processes. On the other hand, the ability to predict meta-stable pitting with some precision is important for many applications in corrosion science, which favors XGBoost.

Moreover, LightGBM and ANN demonstrate better accuracy in predicting E_{corr} compared to the other algorithms. LightGBM and ANN predict E_{corr} of approximately -822 and -825 mV vs SCE, respectively, whereas the empirical value was -726 mV vs SCE, as displayed in 4.4. On the other hand, LightGBM inaccurately predicts a shift in current at approximately -0.54 V vs SCE, a feature seen at pH 7.0 (Figure E13). Consequently, LightGBM had the highest MAPE of 24.51%. The cause of this is difficult to clarify with certainty. Nevertheless, it is possible that the weakness of LightGBM's EFD and GOSS are exploited at this pH range due to the broad variety of empirical polarization curve shapes at neutral pH. Notably, the anodic current density at pH 7.0 (Figure E13) is significantly lower than that of pH 6.8 (Figure E13), pH 7.6 and 7.8 (Figure E15). Due to the high contrast, the GOSS feature of LightGBM may have utilized data samples of pH 7.0 more frequently as adapting to pH 7.0 plausibly increased the loss gradient. This can be visualized in Figure 4.4c by the substantially lower anodic current of LightGBM compared to empirical data and the other algorithms, plausibly a consequence of the lower anodic current at pH 7.0 relative to e.g pH 6.8 and 7.6. Additionally, and another possible explanation, is the removal of necessary information by the EFD feature of LightGBM.

Furthermore, at pH 7.4, the average cathodic Tafel slope of -346 mV/decade was 34% lower than the measured empirical Tafel slope of -520 mV/decade. The large error is on the other hand not too surprising due to the big variety of cathodic behaviour at neutral pH. ANN and LightGBM provided the best predictions with approximately -565 and -413 mV/decade, respectively. Yet, the cathodic branch of ANN did not show a clear Tafel line and the potential range applied for the linear regression were relatively small as a consequence of the big shift in gradient at $\approx$ -0.97 V vs SCE in combination with the lower limit to avoid HER kinetics. LightGBM predicted the cathodic Tafel slope more accurately than the other DT algorithms due to the plausibly higher weighting of pH 7.0 (Figure E13), which exhibited a Tafel slope of approximately -439 mV/decade, as found in Table A8. The Tafel slopes of RF, CatBoost and XGBoost were indeed comparable to the empirical Tafel slope of -253 mV/decade at pH 7.6 seen in Table A8, which reinforces the assumption made previously that these tree algorithms weighted data at pH 7.6 in a more significant manner.

The large general errors in E_{corr} and cathodic Tafel slope $|b_{\text{c}}|$ induced an overall i_{corr} prediction error of -28%. Interestingly, LightGBM had the definite largest MAPE overall, but had the lowest error in both E_{corr} and i_{corr} as it was largely penalized by utilizing anodic features of pH 7.0 (Figure E13).

5.7 Evaluation of polarization curve at pH 11.6

From Table 4.1, it is recognized that the average MAPE at pH 11.6 was approximately 7%, with ANN outperformning the other algorithms with a MAPE of 3%. An interesting observation in Figure 4.1d is that while CatBoost and LightGBM provide very similar polarization curves at this pH, so does XGBoost and CatBoost. The polarization curve predicted by ANN exhibit a distinct shape.

Interestingly, it appears that CatBoost and LightGBM have incorporated the empirical data from pH 11.4 (Figure E24) into their predictions to a larger extent, as their predicted polarization curves exhibit numerous similarities with the empirical curve at pH 11.4. Conversely, XGBoost and RF seem to have given more weight to the polarization curve at pH 11.8 (Figure E25). An interesting observation can be made regarding the step-wise shape present in the polarization curves of CatBoost and LightGBM, which contrasts with the smoother curves generated by the other algorithms. Such step-wise behaviour appears as a sign of underfitting. However, none of the other predicted polarization curves by LightGBM nor CatBoost are exhibiting similar shape. Thus, it is not clear what causes the step-wise appearance. It is not observed any enhanced complexity of the polarization curves in alkaline media, so the oddly shaped polarization curves by LightGBM and CatBoost at pH 11.6 must be a cause of the tree architecture, if not solely a consequence of the random data sampling where alkaline training data were sampled less frequently. However, error caused by the random sampling is unlikely as LightGBM and CatBoost predict comparable polarization curve shapes. The anodic branches predicted by LightGBM and CatBoost at pH 11.6 exhibit similar behaviour as observed in pH 11.4 (Figure E24, which demonstrates unique appearance seen by the rapid shifts in current for larger potentials. Thus, it appears that CatBoost and LightGBM managed to predict the overall trend, albeit with lack of smoothness - a typical sign of underfitting. From the learning curve in Figure 4.8a, it is however observed that neither LightGBM nor CatBoost demonstrate significant more learning potential from 1000 iterations and out, as the validation loss gradients represent declining trends. Consequently, it appears that the optimal hyperparameter combinations are yet to be found, or that XGBoost is a more suitable algorithm, solely concluded by observing the learning curves. This assumption is however not supported by the errors demonstrated on test data, as depicted in Table 4.1.

Furthermore, regarding the polarization curve features at pH 11.6, the empirical E_{corr} at pH 11.6 was -1470 mV vs SCE, as given in Table 4.5. LightGBM and CatBoost predicted similar corrosion potentials at -1393 and 1394 mV vs SCE, respectively. On the other hand, XGBoost, RF and ANN predicted much higher corrosion potentials, with only small deviations from E_{corr} of pH 11.8 (-1493 mV vs SCE). In total, the average prediction of E_{corr} (-1450 mV vs SCE) was 1% higher than the empirical E_{corr} of -1470 mV vs SCE. Furthermore, all algorithms predicted the cathodic branch reasonably well, resulting in an average Tafel slope of -550 mV/decade, only 7% lower than the empirical value of -515 mV/decade. Consequently, a low error in the estimated i_{corr} of 0.91 mA/cm^2 was observed, only 9% higher than the empirical i_{corr} of 0.84 mA/cm^2.

6 Conclusions and further work

Generally assessed, the algorithms proved similar predictive abilities with an average Mean Absolute Percentage Error (MAPE) of approximately 7% for the predicted logarithmic current density. Additionally, the corrosion potential, cathodic Tafel slope, and corrosion current density were estimated with average MAPE across all pH testing data of approximately 5%, 13% and 19%, respectively. These results illustrate the algorithms' capabilities to capture the pH dependency of pivotal polarization curve features on AA6060 + 0.0043 wt% nickel. In addition, the algorithms show potential to pinpoint important transitions, for instance the occurrence of meta-stable pitting, while also accurately locating the pitting potential. The results emphasize the substantial advantage of utilizing machine learning as a time-saving alternative for identifying critical pH dependent features, traditionally achieved through lengthy testing procedures.

It is further recognized that the use of Artificial Neural Networks (ANN) may not necessarily yield the desired payoff, given its time-intensive tuning process, while only obtaining average results compared to the other algorithms. The fastest algorithm, the Random Forest (RF), was also the top performer. This was assumed a consequence of using only pH (in addition to electrochemical potential) as input feature and thus neglecting the surface condition. Overlooking features such as cracks or segregation resulted in a data set with a high degree of unaccounted behaviour. Consequently, the algorithms struggled with adapting to the training data, favoring use of the Random Forest algorithm as it provides great ability to handle noisy data sets. The restrictive use of pH as the sole input feature was further evidenced by large contrast between the errors on known data compared to those on unseen data, which differed by several multiples. For future applications, implementing surface condition, prior to and post polarization, into the feature space may prove advantageous. These data can for instance be obtained through a scanning electron microscope, and further convert the grey scales pixels to integers between e.g. 0 and 1. By incorporating surface features into the training data set, the effect of re-usage of aluminum samples across multiple experiments would have been comprehended.

Furthermore, there are several adjustable parameters like the sampling period and scan rate during polarization that could have been increased to enhance the smoothness of the polarization curves. Nonetheless, while this approach could streamline the data collection process, it could have reduced the predictive abilities of features such as meta-stable pitting and the pitting potential. Like with any other machine learning task, given sufficient time, it could be beneficial to add more empirical data for training. Instead of using the 0.2 interval employed in this work, more detailed polarization curves could be predicted by having intervals of 0.1. Furthermore, if features like the corrosion current density are of particular interest, it could be beneficial to exclude the anodic curve from the training data. This would shift the focus towards the cathodic branch. However, this would also require the utilization of the corrosion potential and not the anodic curve for calculating the corrosion current density.

Further improvement of the predictive capabilities may be achievable by expanding the search spaces in the hyperparameter tuning processes, while employing tuning of all iterative algorithms. However, this will indeed be time consuming and the trade-off between accuracy and training time must be considered. The obtained learning curves, displaying the training and validation at each iteration, imply that all iterative algorithms had room for improvement, as the error converged to a value shortly after training was initiated. Applying smaller learning rates and increase the complexity of the trees by e.g. allowing for more nodes may be beneficial to enhance the likelihood of finding the error minima.

Ultimately, it may prove beneficial to concentrate on tuning just one Gradient Boosted Decision Tree (GBDT) algorithm, given their overall comparable performance, with no observed distinction between the leaf- and level-wise tree growing structures. Categorical Boosting (CatBoost) emerges as a natural choice for further development, as it exhibited the lowest average error across all pH for the GBDT algorithms in combination with being the second fastest algorithm per iteration. Furthermore, considering Random Forest (RF) may also prove beneficial, considering its enhanced performance compared to the other algorithms, both in terms of precision and training time. On the other hand, by expanding the input feature dimension to limit unknown behaviour in the data set, CatBoost or another GBDT algorithm discussed in this work could potentially exhibit improved performance given their ability to correct for errors across trees in the ensemble.

References

[1] J. G. Kaufman et al., *Mechanical Engineers' Handbook 4th edition*, vol. 1 of *Materials and Engineering Mechanics.* Wiley, 2015. Ch. 3: aluminium alloys, ISBN: 978-1-118-11282-3.

[2] C. Exley, "Human exposure to aluminium," *The Royal Society of Chemistry 2013*, vol. 15, pp. 1807–1816, 2013. DOI: https://doi.org/10.1039/C3EM00374D.

[3] Z. S. Smialowska, "Pitting corrosion of aluminum," *Corrosion Science*, vol. 41, no. 9, pp. 1743–1767, 1999. DOI: https://doi.org/10.1016/S0010-938X(99)00012-8.

[4] B. Zaid et al., "Effects of ph and chloride concentration on pitting corrosion of aa6061 aluminum alloy," *Corrosion Science*, vol. 50, no. 7, pp. 1841–1847, 2008. DOI: https://doi.org/10.1016/j.corsci.2008.03.006.

[5] M. S. Skui, "The use of artificial neural networks and random forest to predict polarization curves of aa6060 + nickel alloys in various electrolyte environments," 2022. Project work by this author, p. 15.

[6] M. Stratmann and G. Frankel, *Corrosion and oxide films*, vol. 4 of *Encyclopedia of electrochemistry.* Wiley-VCH, 2003. Ch. 1.3 and 1.4, ISBN: 3-527-30396-0.

[7] T. Shinagawa et al., "Insight on tafel slopes from a microkinetic analysis of aqueous electrocatalysis for energy conversion," *Sci Rep*, vol. 5, 2015. DOI: https://doi.org/10.1038/srep13801.

[8] A. T. Marshall and L. Vaisson-Béthune, "Avoid the quasi-equilibrium assumption when evaluating the electrocatalytic oxygen evolution reaction mechanism by tafel slope analysis," *Electrochemistry Communications*, vol. 61, pp. 23–26, 2015. DOI: https://doi.org/10.1016/j.elecom.2015.09.019.

[9] B. V. Tilak and C. P. Chen, "Generalized analytical expressions for tafel slope, reaction order and a.c. impedance for the hydrogen evolution reaction (her): mechanism of her on platinum in alkaline media," *Journal of Applied Electrochemistry*, vol. 23, p. 631–640, 1992. DOI: https://doi.org/10.1007/BF00721955.

[10] M. F. Li et al., "ph effect on oxygen reduction reaction at pt(111) electrode," *Electrochimica Acta*, vol. 110, pp. 780–789, 2013. DOI: https://doi.org/10.1016/j.electacta.2013.04.096.

[11] K. Aggarwal et al., "Has the future started? the current growth of artificial intelligence, machine learning, and deep learning," *Iraqi Journal for ComputerScienceand-Mathematics*, vol. 4, pp. 115–123, 2021. DOI: https://doi.org/10.52866/ijcsm.2022.01.01.013.

[12] E. Naqa et al., *What Is Machine Learning?*, pp. 3–11. Cham: Springer International Publishing, 2015. DOI: https://doi.org/10.1007/978-3-319-18305-3_1.

[13] L. Breimann, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, 2001. DOI: `https://doi.org/10.1023/A:1010933404324`.

[14] B. Sun et al. , “Shared mobility for transport and its environmental impact,” *Journal of Advanced Transportation*, vol. 2021, 2021. DOI: `https://doi.org/10.1155/2021/5559562`.

[15] G. Ke et al. , “Lightgbm: A highly efficient gradient boosting decision tree,” *Advances in neural information processing systems*, vol. 30, 2017.

[16] L. Prokhorenkova et al., “Catboost: unbiased boosting with categorical features,” 2017. DOI: `https://doi.org/10.48550/arXiv.1706.09516`.

[17] P. M. Atkinson and A. R. L. Tatnall , “Introduction neural networks in remote sensing,” *International Journal of Remote Sensing*, vol. 18, no. 4, 1997. DOI: `https://doi.org/10.1080/014311697218700`.

[18] A. Natekin and A. Knoll, “Gradient boosting machines, a tutorial,” *Frontiers in Neurorobotics*, vol. 7, 2013. DOI: `https://doi.org/10.3389/fnbot.2013.00021`.

[19] B. Charbuty and A. Abdulazeez, “Classification based on decision tree algorithm for machine learning,” *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, 2021. DOI: `https://doi.org/10.38094/jastt20165`.

[20] V. A. Dev and M. R. Eden, “Formation lithology classification using scalable gradient boosted decision trees,” *Computers and Chemical Engineering*, vol. 128, pp. 392–404, 2019. DOI: `https://doi.org/10.1016/j.compchemeng.2019.06.001`.

[21] H. Alshari et al., “Comparison of gradient boosting decision tree algorithms for cpu performance,” *Journal of Institue Of Science and Technology*, vol. 37, no. 1, pp. 157–168, 2021.

[22] C. Darken and J. Moody, “Note on learning rate schedules for stochastic optimization,” in *Advances in Neural Information Processing Systems* (R. Lippmann, J. Moody, and D. Touretzky, eds.), vol. 3, p. 837, Morgan-Kaufmann, 1990.

[23] K. Kouvaris et al., “How evolution learns to generalise: Principles of under-fitting, over-fitting and induction in the evolution of developmental organisation,” 2015. DOI: `https://doi.org/10.48550/arXiv.1508.06854`.

[24] C. C. Aggarwal, *Neural networks and deep learning.* Springer, 2018. DOI: `https://doi.org/10.1007/978-3-319-94463-0`.

[25] G. Biau and E. Scornet, “A random forest guided tour,” *TEST*, vol. 25, 2016. pp. 197-227, DOI: `https://doi.org/10.1007/s11749-016-0481-7`.

[26] S. Ren et al., “Global refinement of random forest,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[27] scikit-learn developers, "sklearn.ensemble.randomforestregressor." URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Accessed: 10.03.2023.

[28] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?," *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[29] scikit-learn developers, "Ensemble methods." URL: https://scikit-learn.org/stable/modules/ensemble.html#forest. Accessed: 09.05.2023.

[30] P. Probst et al., "Hyperparameters and tuning strategies for random forest," *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, vol. 9, no. 3, p. e1301, 2019. DOI: https://doi.org/10.1002/widm.1301.

[31] W. Dong et al., "Xgboost algorithm-based prediction of concrete electrical resistivity for structural health monitoring," *Automation in Construction*, vol. 114, p. 103155, 2020. DOI: https://doi.org/10.1016/j.autcon.2020.103155.

[32] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 785–794, Association for Computing Machinery, 2016. DOI: https://doi.org/10.1145/2939672.2939785.

[33] XGBoost developers, "Notes on parameter tuning." URL: https://xgboost.readthedocs.io/en/stable/tutorials/param_tuning.html. Accessed: 23.05.2023.

[34] T. Kavzoglu1 and A. Teke, "Advanced hyperparameter optimization for improved spatial prediction of shallow landslides using extreme gradient boosting (xgboost)," *Bull Eng Geol Environ*, vol. 81, 2022. DOI: https://doi.org/10.1007/s10064-022-02708-w.

[35] XGBoost developers, "Xgboost parameters." URL: https://xgboost.readthedocs.io/en/stable/parameter.html#parameters-for-tree-booster. Accessed: 10.03.2023.

[36] Microsoft Corporation, "Parameters tuning." URL: https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html. Accessed: 23.05.2023.

[37] Microsoft Corporation, "lightgbm.lgbmregressor." URL:https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html. Accessed: 10.03.2023.

[38] Microsoft Corporation, "Parameters." URL:https://lightgbm.readthedocs.io/en/latest/Parameters.html. Accessed: 10.03.2023.

[39] Yandex, "Parameter tuning." URL: `https://catboost.ai/en/docs/concepts/parameter-tuning`. 13.03.2023.

[40] Yandex, "Common parameters." URL: `https://catboost.ai/en/docs/references/training-parameters/common`. Accessed: 13.03.2023.

[41] A. Goldbloom, "What algorithms are most successful on kaggle?." URL: `https://www.kaggle.com/code/antgoldbloom/what-algorithms-are-most-successful-on-kaggle`. Accessed: 29.03.2023.

[42] A. J. et al., "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996. DOI: 10.1109/2.485891.

[43] S. Sharma et al., "Activation functions in neural networks," *International Journal of Engineering Applied Sciences and Technology,*, vol. 4, pp. 310–316, 2020. ISSN: 2455-2143.

[44] Q. W. et al., "A comprehensive survey of loss functions in machine learning," *Annals of Data Science*, vol. 9, p. 187–212, 2022. DOI: `https://doi.org/10.1007/s40745-020-00253-5`.

[45] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," *Information Technology and Management Science*, vol. 20, pp. 20–24, 2017. DOI: 10.1515/itms-2017-0003.

[46] C. G. Hounmenou et al., "A formalism of the general mathematical expression of multilayer perceptron neural networks," 2021. DOI: `https://doi.org/10.20944/preprints202105.0412.v1`.

[47] S. Kiliçarslan and M. Celik, "Rsigelu: A nonlinear activation function for deep neural networks," *Expert Systems with Applications*, vol. 174, 2021. DOI: `https://doi.org/10.1016/j.eswa.2021.114805`.

[48] R. M. Schmidt et al., "Descending through a crowded valley - benchmarking deep learning optimizers," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 9367–9376, PMLR, 2021. URL: `http://proceedings.mlr.press/v139/schmidt21a/schmidt21a.pdf`.

[49] Q. H. Nguyen et al., "Influence of data splitting on performance of machine learning models in prediction of shear strength of soil," *Mathematical Problems in Engineering*, vol. 2021, 2021. DOI: `https://doi.org/10.1155/2021/4832864`.

[50] W. Wu et al., "A method for comparing data splitting approaches for developing hydrological ann models," *International Congress on Environmental Modelling and Software*, vol. 6, p. 2, 2012.

[51] X. Du, H. Xu, and F. Zhu, "Understanding the effect of hyperparameter optimization

on machine learning models for structure design problems," *Computer-Aided Design*, vol. 135, 2021. DOI: https://doi.org/10.1016/j.cad.2021.103013.

[52] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," 2019. DOI: https://doi.org/10.48550/arXiv.1912.06059.

[53] C. A. Ellis and S. A. Parbery, "Is smarter better? a comparison of adaptive, and simple moving average trading strategies," *Research in International Business and Finance*, vol. 19, no. 3, pp. 399–411, 2005. DOI: https://doi.org/10.1016/j.ribaf.2004.12.009.

[54] XGBoost developers, "Xgboost parameters." URL: https://xgboost.readthedocs.io/en/stable/python/index.html. Accessed: 18.05.2023.

[55] Keras developers, "About keras." URL: https://keras.io/about/. Accessed: 18.05.2023.

[56] Alec S. Dyer et al., "Applied machine learning model comparison: Predicting offshore platform integrity with gradient boosting algorithms and neural networks," *Marine Structures*, vol. 83, 2022. DOI: https://doi.org/10.1016/j.marstruc.2021.103152.

[57] G. K. F. Tso et al., "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," *Energy*, vol. 32, no. 9, pp. 1761–1768, 2007. DOI: https://doi.org/10.1016/j.energy.2006.11.010.

[58] C. Bentéjac et al., "A comparative analysis of gradient boosting algorithms.," *Artif Intell Rev*, vol. 54, p. 1937–1967, 2021. DOI: https://doi.org/10.1007/s10462-020-09896-5.

[59] Wei-Yin Loh, "Classification and regression trees," *WIREs Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011. DOI: https://doi.org/10.1002/widm.8.

[60] S. Duffner et al., "An online backpropagation algorithm with validation error-based adaptive learning rate," in *Artificial Neural Networks – ICANN 2007* (J. M. de Sá, L. A. Alexandre, W. Duch, and D. Mandic, eds.), (Berlin, Heidelberg), pp. 249–258, Springer Berlin Heidelberg, 2007. DOI: https://doi.org/10.1007/978-3-540-74690-4_26.

Appendices

A Hyperparameter search space and applied hyperparameters

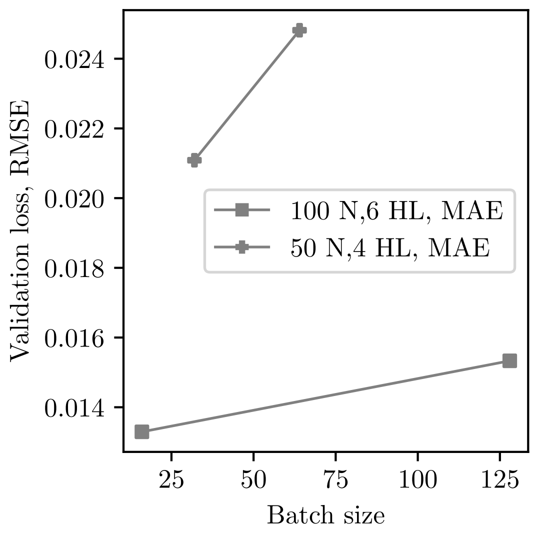
A.1 ANN

Table A1: Applied hyperparameters for ANN with search space in the random search

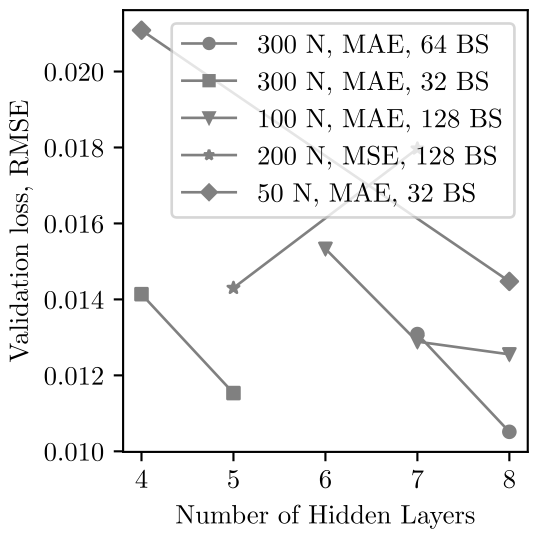
Param	Search space	Optimal
Epochs applied, terminated	N/A	80, 16
Early stopping (iterations)	N/A	4
Hidden layers	$\mathbb{Z} \in [2, 9]$	7
Neurons	[20, 50, 70, 100, 150, 200, 300, 400, 600, 900, 1200]	900
Optimizer	Adam	Adam
Learning rate [$\log_{10}$]	[-3, -2, -1]	0.001
L2 coefficient [$\log_{10}$]	[-4, -3, -2]	0
Batch size	[16, 32, 64, 128]	16
Activation function	[ReLU, sigmoid, tanh]	ReLU
Loss function	[MSE, MAE]	MAE
Train/val split [%]	N/A	80/20

Table A2: ANN tuning results from RandomSearch. *N* denotes the last tuning session

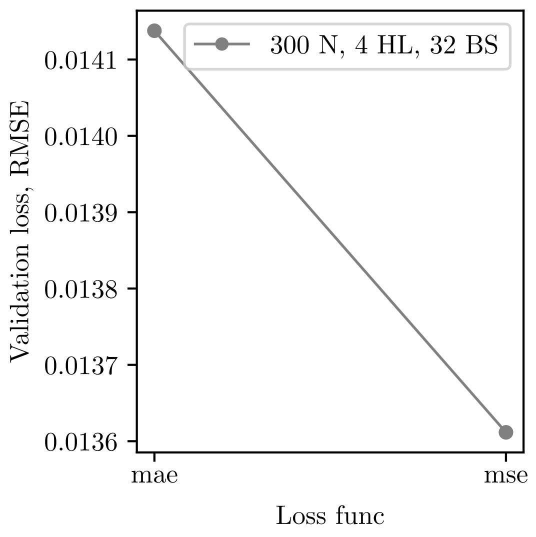
Validation error (RMSE)		**Neurons**		**HLs**		**Loss function**		**Batch size**	
N-1	**N**	**N-1**	**N**	**N-1**	**N**	**N-1**	**N**	**N-1**	**N**
0.010337	0.01149	900	900	7	6	MAE	MSE	16	32
0.010513	0.01180	300	300	8	7	MAE	MSE	64	32
0.011082	0.01190	50	900	6	8	MAE	MSE	16	32
0.011168	0.01224	400	900	6	5	MAE	MAE	64	32
0.011237	0.01234	900	1200	6	8	MAE	MSE	128	64
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.016791	0.01494	100	200	6	6	MSE	MSE	32	64
0.017971	0.01635	200	200	7	8	MSE	MSE	128	64
0.018367	0.01637	200	400	4	7	MSE	MSE	64	32
0.021087	0.01644	50	300	4	7	MAE	MSE	32	64
0.024818	0.01709	50	1200	4	5	MAE	MSE	64	64



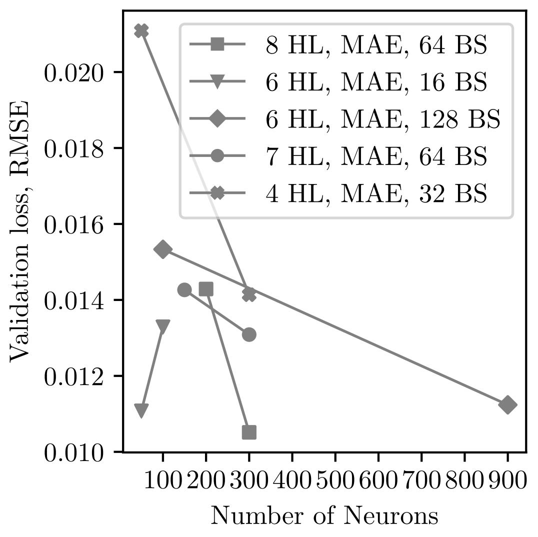
(a) Batch size



(b) Hidden layers



(c) Loss functions



(d) Neurons

Figure A1: The exploration of one independent hyperparameter's effect on the validation error/loss by variation in solely one hyperparameter. Data in the second last tuning exercise (N - 1 in Table A2) are used to construct the output. The data set can be found in src/tuning_result_ANN/df_results_8_HL.csv on Github[1].

[1]Githup repository: `https://github.com/matsssk/AA6060_ML/tree/main`

A.2 LightGBM hyperparameter tuning and applied hyperparameters

Table A3: LightGBM applied hyperparameters together with search space. Star (*) marks default value. See `lightgbm.LGBMRegressor` for the remaining model parameters[4].

Parameter	**Search space**	**Applied**
n_estimators	1000	10^5
Early stopping criteria (iterations)	N/A	50
num_leaves	[21, 31, 41, 51, 61]	31*
learning_rate	[0.001, 0.01, 0.1, 0.5]	0.3*
boosting_type	[gbdt, dart]	gbdt*

Table A4: LightGBM tuning results. The three slowest and fasted learners are included as the purpose was to outline the most likely hyperparameter combination to find the local error minimum. None of these hyperparameter combinations were applied in the final LightGBM model.

num_ leaves	**learning_ rate**	**boosting_ type**	**Mean_test_score (closer to 0 : less error)**
Three slowest learners			
61	0.001	dart	-3.1021
51	0.001	dart	-3.0997
31	0.001	dart	-3.0986
Three fastest learners			
21	0.1	gbdt	-0.7567
21	0.5	dart	-0.7535
21	0.1	dart	-0.7514

[4]LGBMRegressor documentation: `https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html`

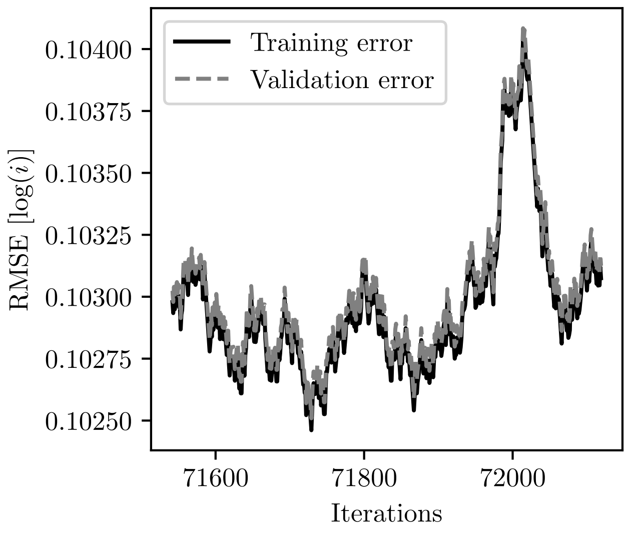


Figure A2: LightGBM training and validation loss before termination where the slowest learner was applied. *num_leaves* = 61, *learning_rate* = 0.001, *boosting_type* = *dart*. This hyperparameter combination was not applied as the error was too high compared to that of default values.

A.3 Random Forest hyperparameter tuning and applied hyperparameters

Table A5: Applied hyperparameters with search space for Random Forest. All the possible hyperparameter combinations were examined in a Grid Search-style manner. The remaning hyperparameters for `sklearn.ensemble.RandomForestRegressor` can be found in the documentation[5].

Parameter	**Search space**	**Optimal**
n_estimators	[1, 5, 10, 100, 200]	5
max_features	[0.3, 1.0]	1*

Table A6: RF tuning results

Trees	**Max features**	**Average RMSE**	**Feature importances**	
(n_estimators)	**(max_features)**	**across all test**	**Potential (E)**	**pH**
5	1.0	0.3975	0.8061	0.1939
10	1.0	0.3976	0.6460	0.3540
100	1.0	0.3977	0.7129	0.2871
200	1.0	0.3977	0.6488	0.3512
1	1.0	0.3979	0.6841	0.3159
1	0.3	0.4047	0.6479	0.3521
100	0.3	0.4158	0.7317	0.2683
200	0.3	0.4167	0.6477	0.3523
5	0.3	0.4201	0.7245	0.2755
10	0.3	0.4245	0.6479	0.3521

[5]RandomForestRegressor `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`

A.4 CatBoost and XGBoost applied hyperparameters

Table A7: CatBoost and XGBoost applied hyperparameters. Default values are marked with star (*). The remaining hyperparameters and their default values can be found in the documentations[5][6]

Parameter	**CatBoost applied**	**XGBoost applied**
Iterations	10^5	10^5
Learning rate	0.35	0.3*
Early stopping criteria (iterations)	50	50

[5]CatBoostRegressor default: `https://catboost.ai/en/docs/references/training-parameters/common`

[6]XGBoostRegressor default: `https://xgboost.readthedocs.io/en/stable/parameter.html`

B Feature importances

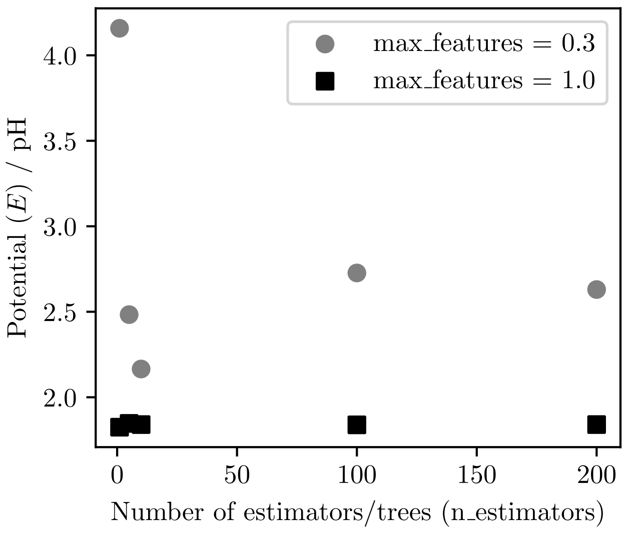


Figure B1: Feature importances in RF as a function of trees (n_estimators) depicted as a fraction potential (E) divided by pH.

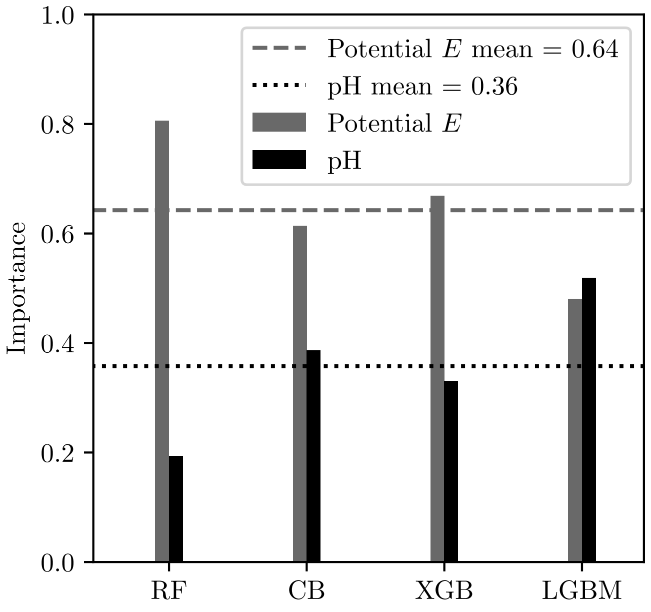
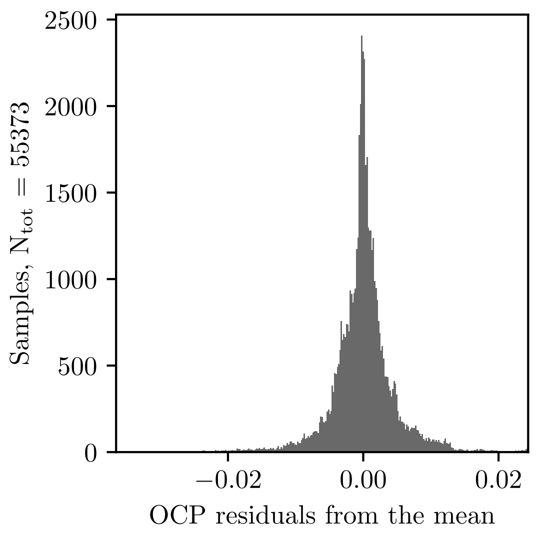
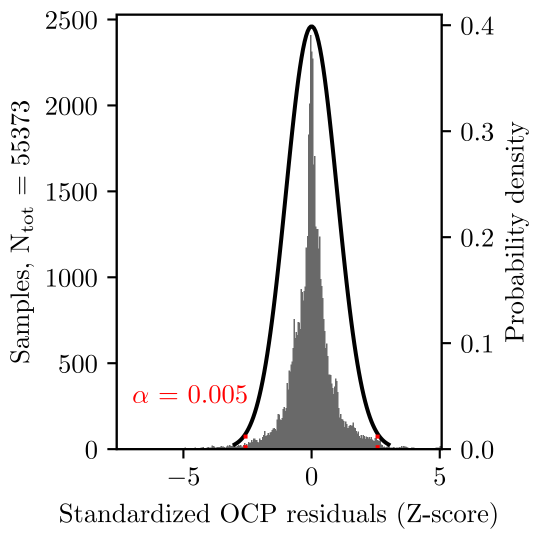


Figure B2: Feature importances estimated by optimal DT models.

C Normality tests

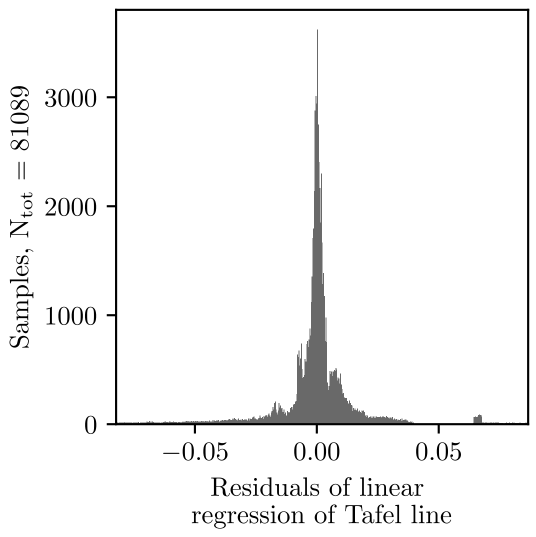


(a) Normality test OCPs

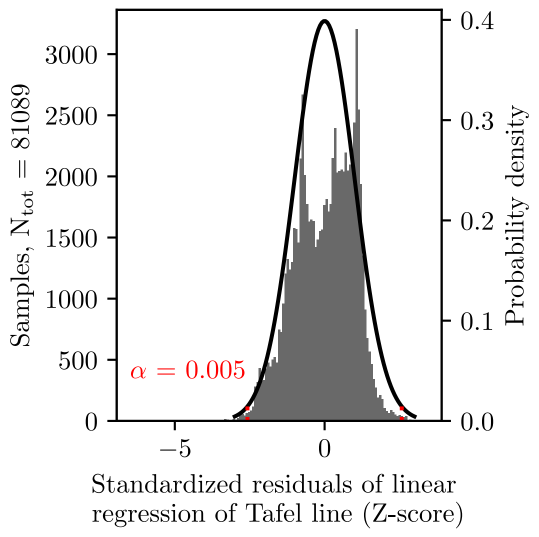


(b) Standardized corrosion potential E_{corr} residuals with the 99% quantiles in red.

Figure C1: Residuals from the mean for the last third of corrosion potential data for each E_{corr} measurement (left, C1a). The residuals were standardized to obtain C1b including the quantiles corresponding to a 99% CI $\alpha = 0.005$.



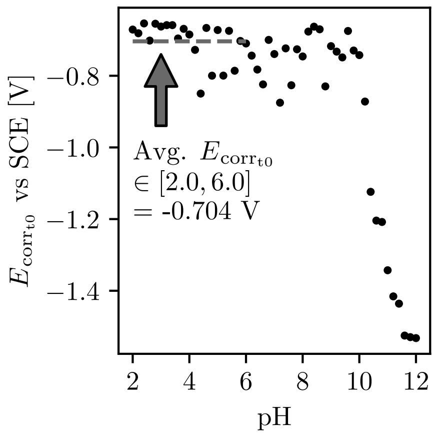
(a) Normality test of the residuals in the linear regression of cathodic Tafel line for all ML models plus experimental data



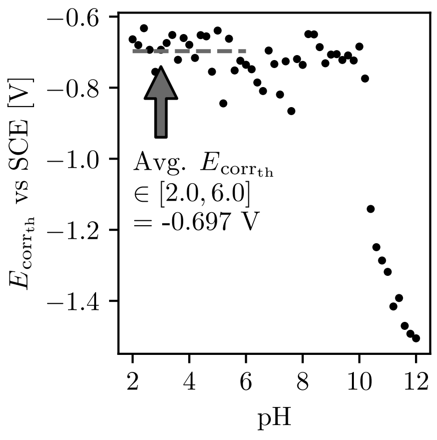
(b) Standardized residuals in the linear regression of cathodic Tafel line for all ML models plus experimental data. 99% quantile is included in red.

Figure C2: Residuals from the Tafel line obtained by linear regression. The residuals from all polarization curves were merged into one plot. Figure C2a depicts the standardized residuals with the quantiles corresponding to a 99% CI $\alpha = 0.005$.

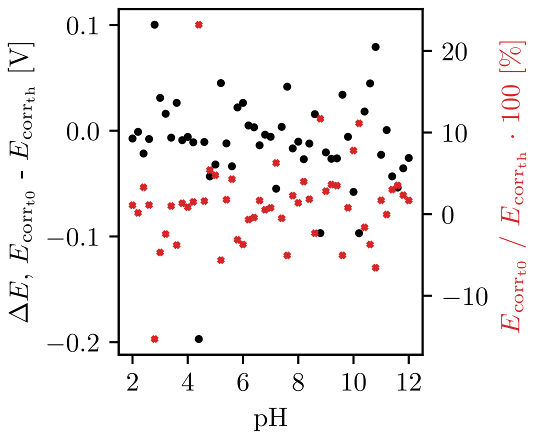
D Trends in corrosion potential and other features



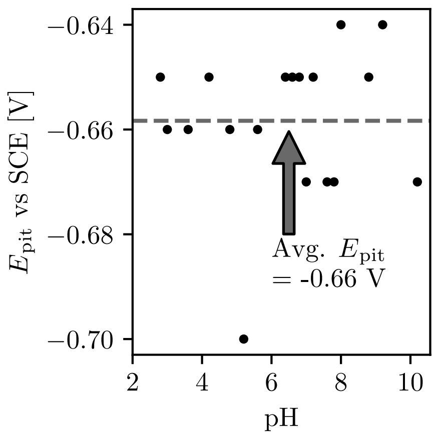
(a) Corrosion potential E_{corr} prior to polarization, denoted as $E_{\mathrm{corr_{t0}}}$



(b) Corrosion potential E_{corr} post cathodic scan, denoted $E_{\mathrm{corr_{th}}}$



(c) The change in corrosion potential E_{corr} during cathodic polarization



(d) The pitting potential E_{pit} plotted as a function of pH if clear pitting potentials are detected

Figure D1: Evaluation of corrosion potential E_{corr} prior to and post cathodic polarization

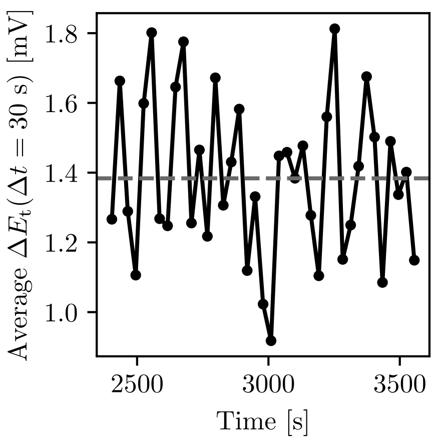


Figure D2: The average 30 seconds change in corrosion potential E_{corr}. The mean is given by the dashed gray horizontal line at ≈ 1.4 mV.

Table A8: Comparison of important polarization curve features at various pH. $E_{\mathrm{corr}_{\mathrm{t0}}}$ and $E_{\mathrm{corr}_{\mathrm{th}}}$ are E_{corr} observed prior to and post cathodic polarization.

pH	$E_{\mathbf{corr_{t0}}}$	$E_{\mathbf{corr_{th}}}$	Δ $E_{\mathbf{corr}}$	$E_{\mathbf{pit}}$	**Tafel slope** $b_{\mathbf{c}}$ **[mV/dec]**	**Lin.reg. R^2**
2.0	-0.671	-0.664	-0.007	N/A	-436	0.9999
2.2	-0.681	-0.680	-0.001	N/A	-367	0.9966
2.4	-0.654	-0.633	-0.022	N/A	-453	0.9917
2.6	-0.702	-0.693	-0.008	N/A	-504	0.9990
2.8	-0.655	-0.755	0.100	-0.65	-336	0.9963
3.0	-0.663	-0.693	0.031	-0.66	-377	0.9991
3.2	-0.659	-0.674	0.016	N/A	-439	0.9970
3.4	-0.659	-0.652	-0.007	N/A	-574	0.9988
3.6	-0.696	-0.722	0.026	-0.66	-436	0.9985
3.8	-0.669	-0.660	-0.009	N/A	-1113	0.9844
4.0	-0.685	-0.679	-0.006	N/A	-820	0.8942
4.2	-0.728	-0.716	-0.011	-0.65	-617	0.9988
4.4	-0.850	-0.652	-0.197	N/A	-645	0.9982
4.6	-0.667	-0.656	-0.011	N/A	-442	0.9985
4.8	-0.800	-0.755	-0.043	-0.66	-280	1.0000
5.0	-0.672	-0.639	-0.032	N/A	-525	0.9971
5.2	-0.800	-0.844	0.045	-0.7	-263	0.9987
5.4	-0.675	-0.662	-0.012	N/A	-710	0.9725
5.6	-0.786	-0.752	-0.034	-0.66	-440	0.9976
5.8	-0.703	-0.724	0.022	N/A	-459	0.9982
6.0	-0.710	-0.736	0.026	N/A	-420	0.9998
6.2	-0.744	-0.748	0.005	N/A	-416	0.9984
6.4	-0.783	-0.785	0.003	-0.65	-384	0.9968
6.6	-0.824	-0.809	-0.014	-0.65	-377	0.9955
6.8	-0.700	-0.695	-0.004	-0.65	-753	0.9833
7.0	-0.739	-0.733	-0.006	-0.67	-439	1.0000
7.2	-0.875	-0.819	-0.055	-0.65	-411	0.9922
7.4	-0.724	-0.726	0.004	N/A	-52	0.9979
7.6	-0.826	-0.866	0.042	-0.67	-253	0.9995
7.8	-0.726	-0.719	-0.017	-0.67	-626	0.9593
8.0	-0.746	-0.736	-0.010	-0.64	-417	0.9962
8.2	-0.677	-0.649	-0.027	N/A	-540	0.9985
8.4	-0.663	-0.650	-0.012	N/A	-429	0.9978
8.6	-0.670	-0.686	0.016	N/A	-514	0.9316
8.8	-0.830	-0.731	-0.097	-0.65	-397	0.9941
9.0	-0.717	-0.707	-0.020	N/A	-662	0.9950
9.2	-0.733	-0.706	-0.027	-0.64	-552	0.9982

9.4	-0.749	-0.722	-0.026	N/A	-437	1.0000
9.6	-0.675	-0.709	0.034	N/A	-449	0.9941
9.8	-0.730	-0.723	-0.006	N/A	-828	0.9993
10.0	-0.742	-0.684	-0.058	N/A	-484	0.9995
10.2	-0.872	-0.774	-0.097	-0.67	-320	0.9997
10.4	-1.124	-1.141	0.018	N/A	-165	0.9989
10.6	-1.204	-1.249	0.045	N/A	-173	0.9989
10.8	-1.208	-1.286	0.079	N/A	-185	0.9968
11.0	-1.343	-1.318	-0.023	N/A	-252	0.9567
11.2	-1.416	-1.416	0.001	N/A	-306	0.9727
11.4	-1.436	-1.392	-0.043	N/A	-429	0.9692
11.6	-1.525	-1.470	-0.054	N/A	-515	0.9758
11.8	-1.529	-1.493	-0.036	N/A	-555	0.9850
12.0	-1.532	-1.505	-0.026	N/A	-568	0.9771

E Empirical polarization curves

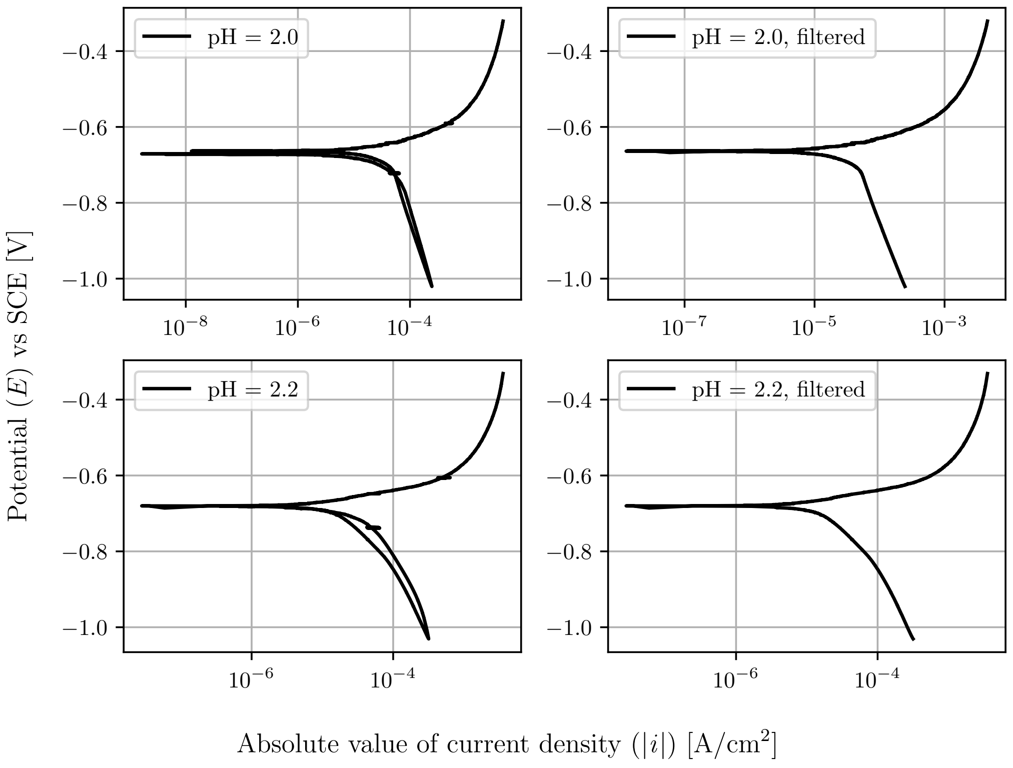
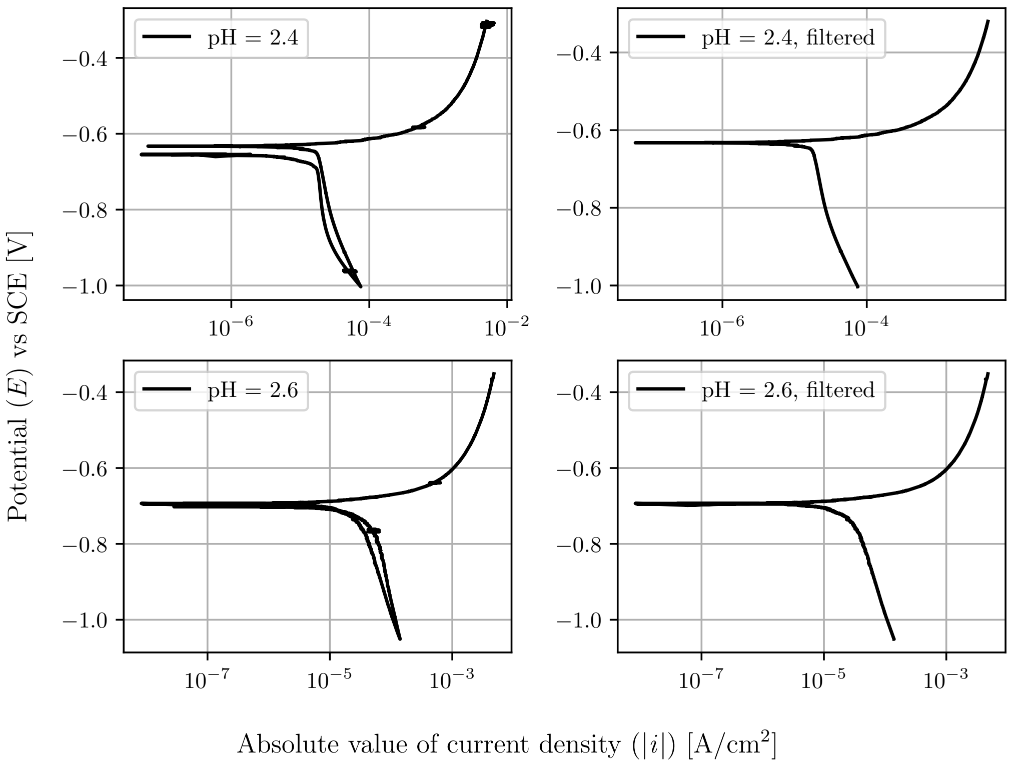
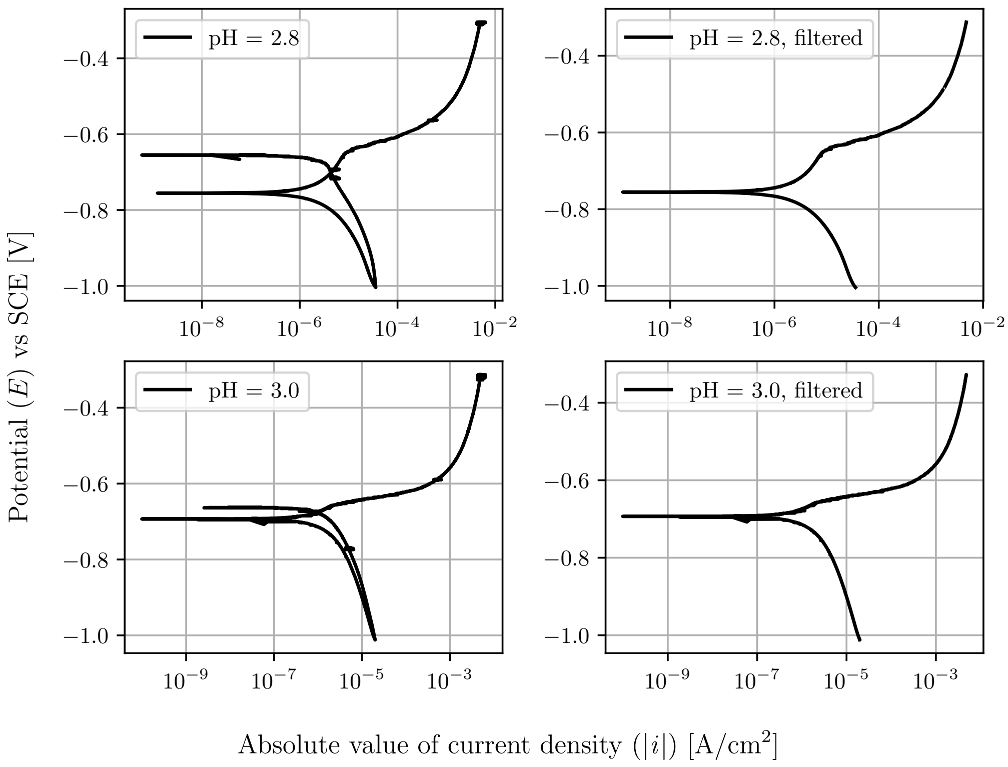


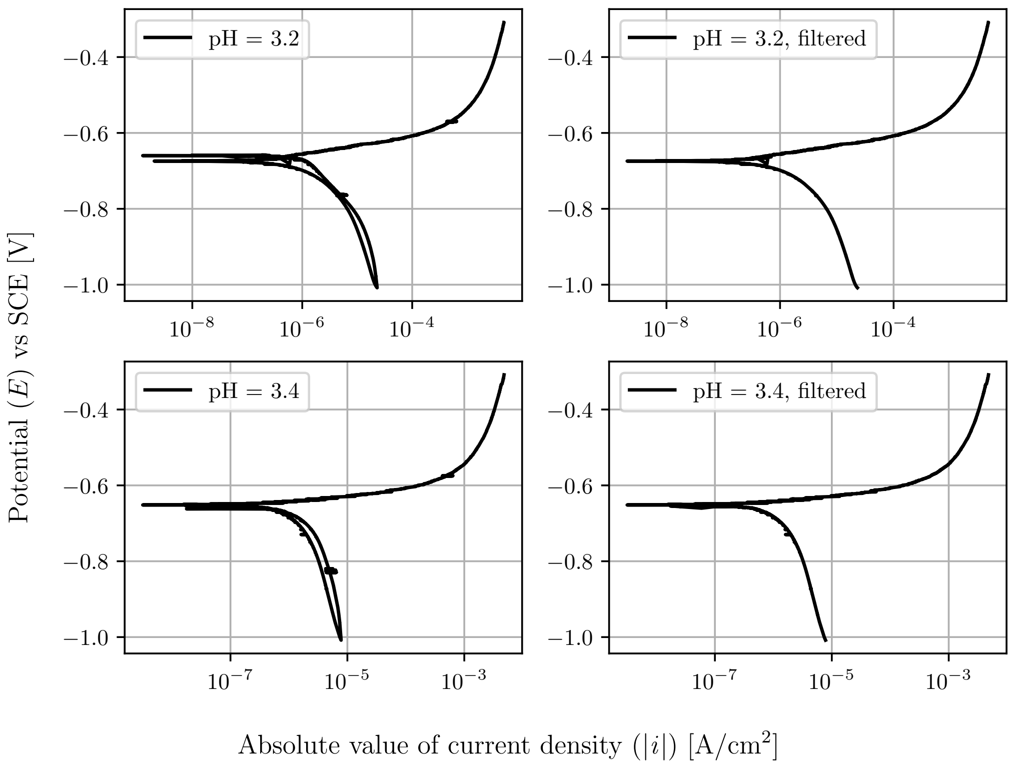
Figure E1: Empirical polarization curve #1.



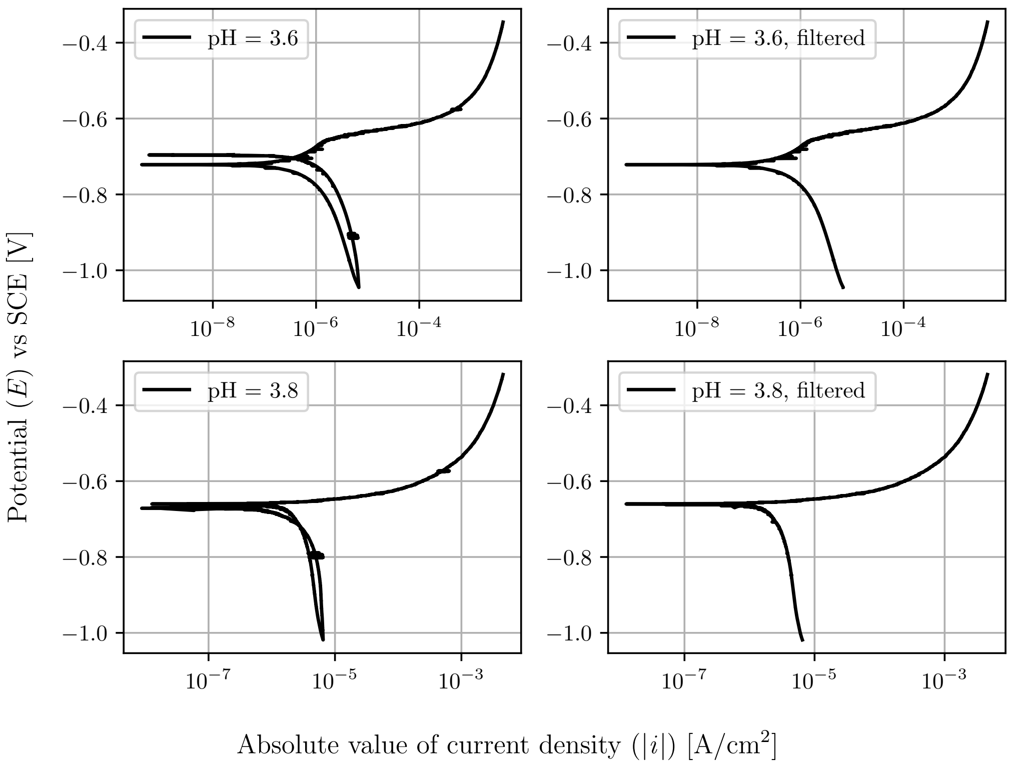
Figures E2: Empirical polarization curve #2.



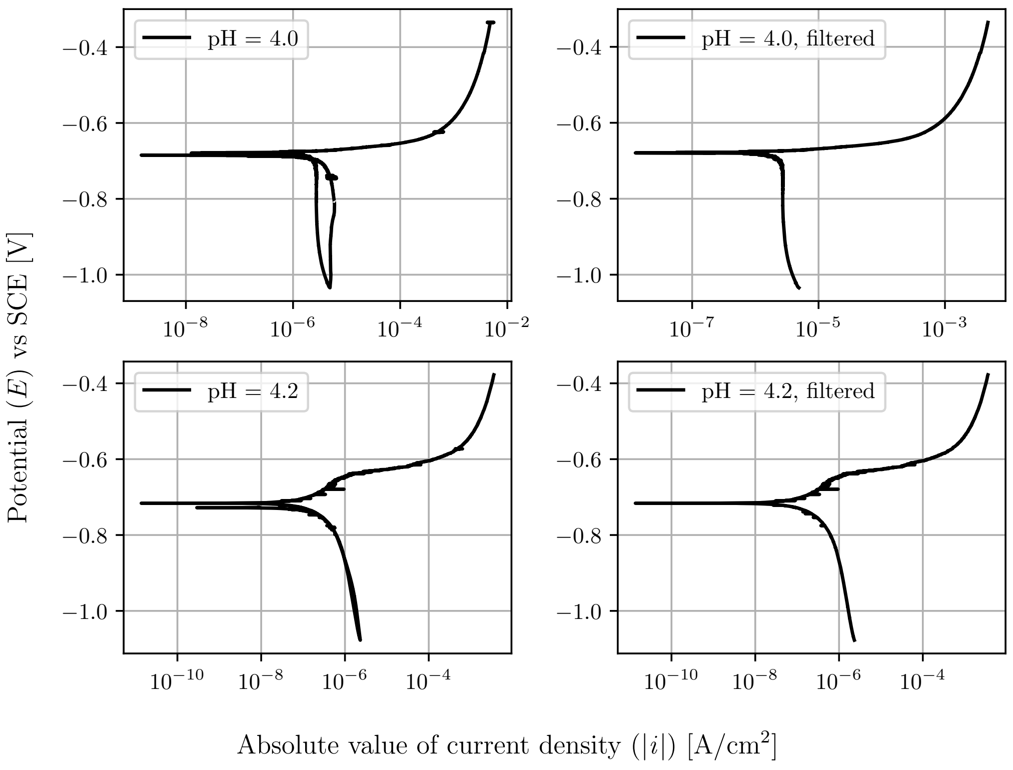
Figures E3: Empirical polarization curve #3.



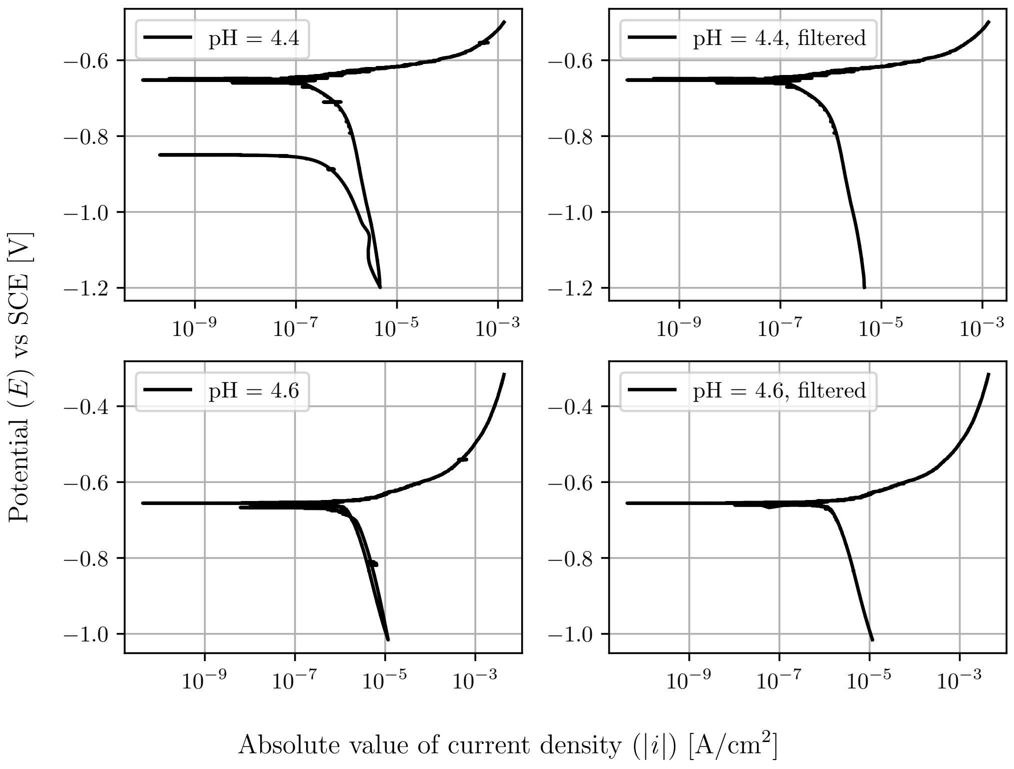
Figures E4: Empirical polarization curve #4.



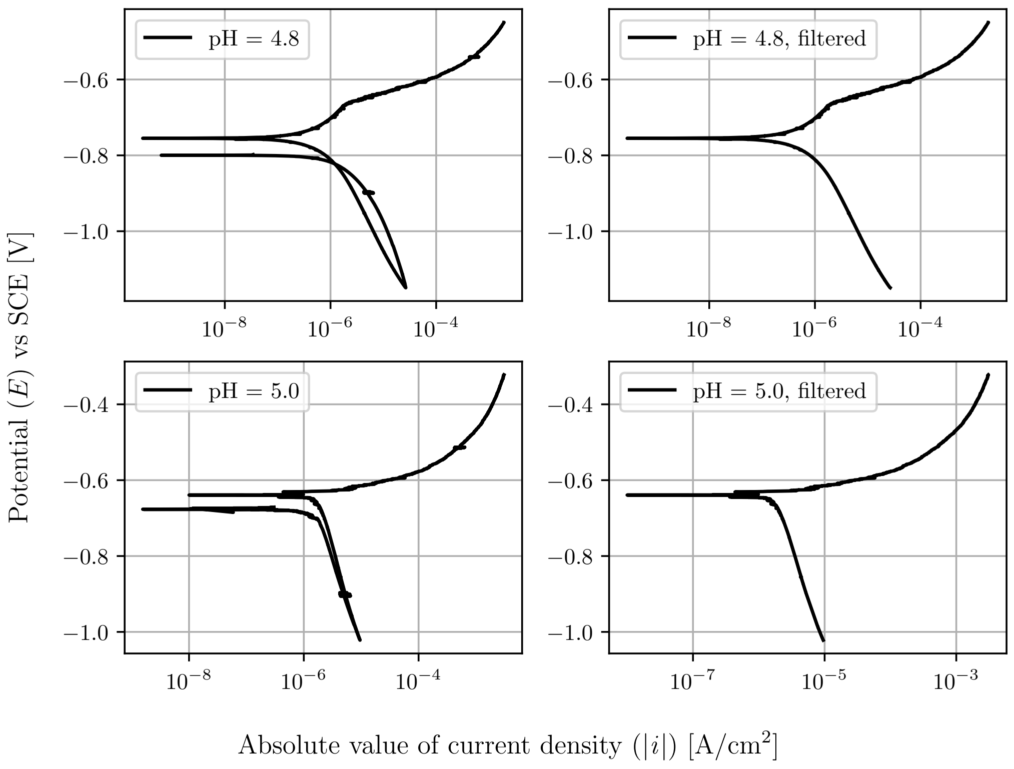
Figures E5: Empirical polarization curve #5.



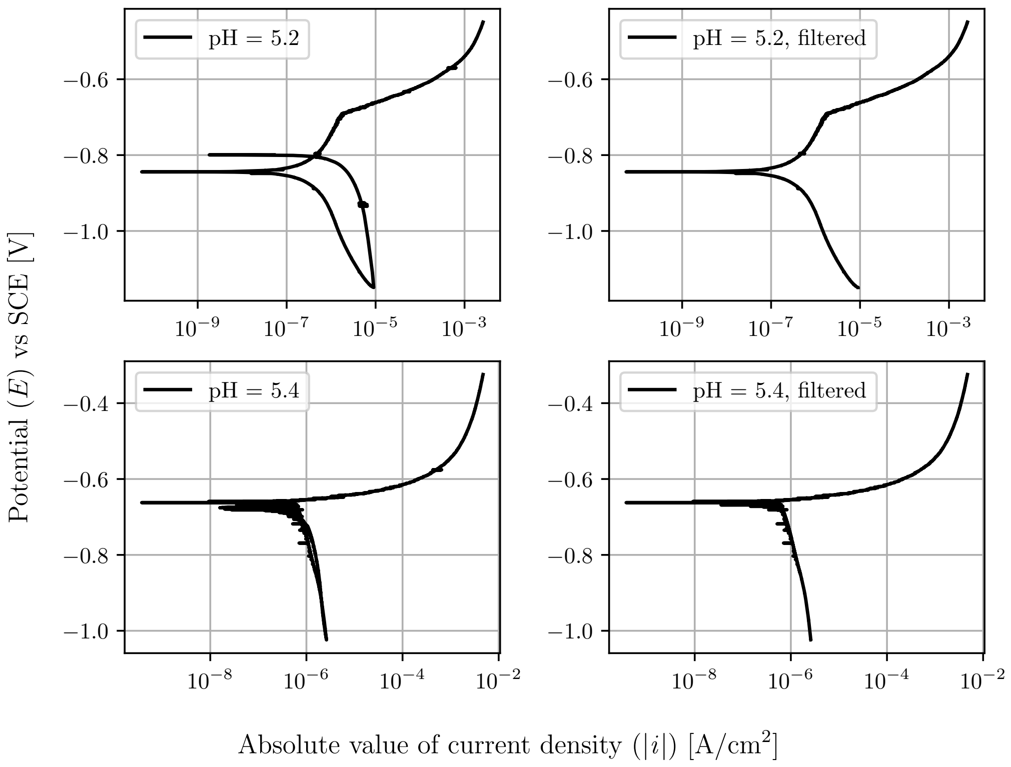
Figures E6: Empirical polarization curve #6.



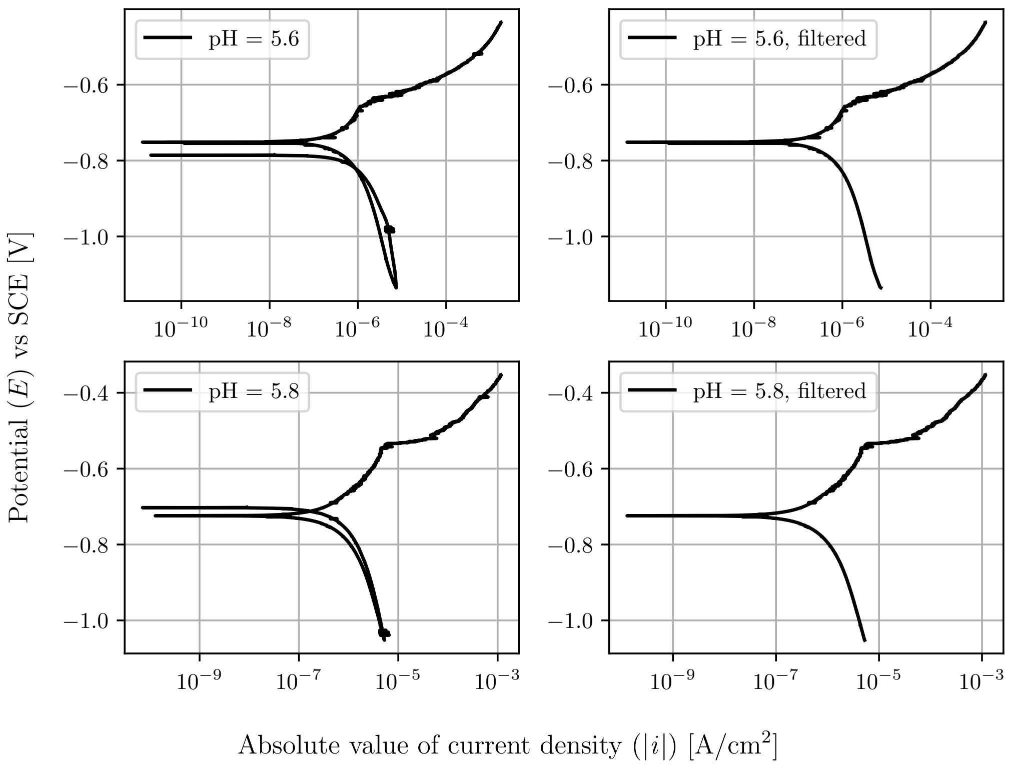
Figures E7: Empirical polarization curve #7.



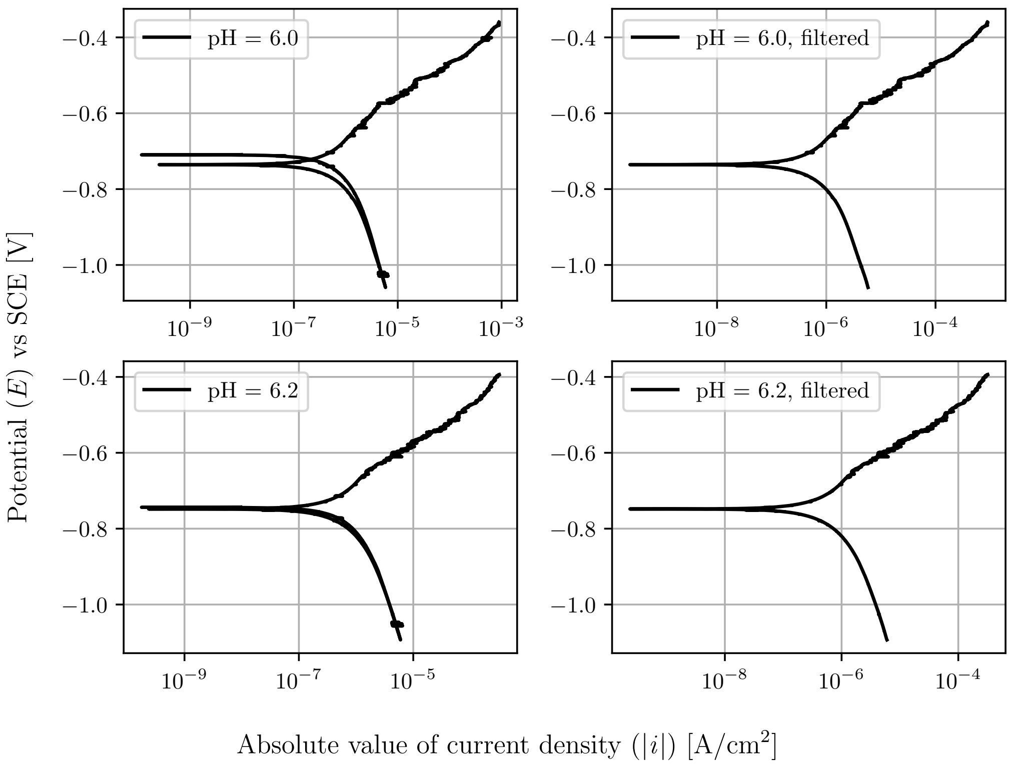
Figures E8: Empirical polarization curve #8.



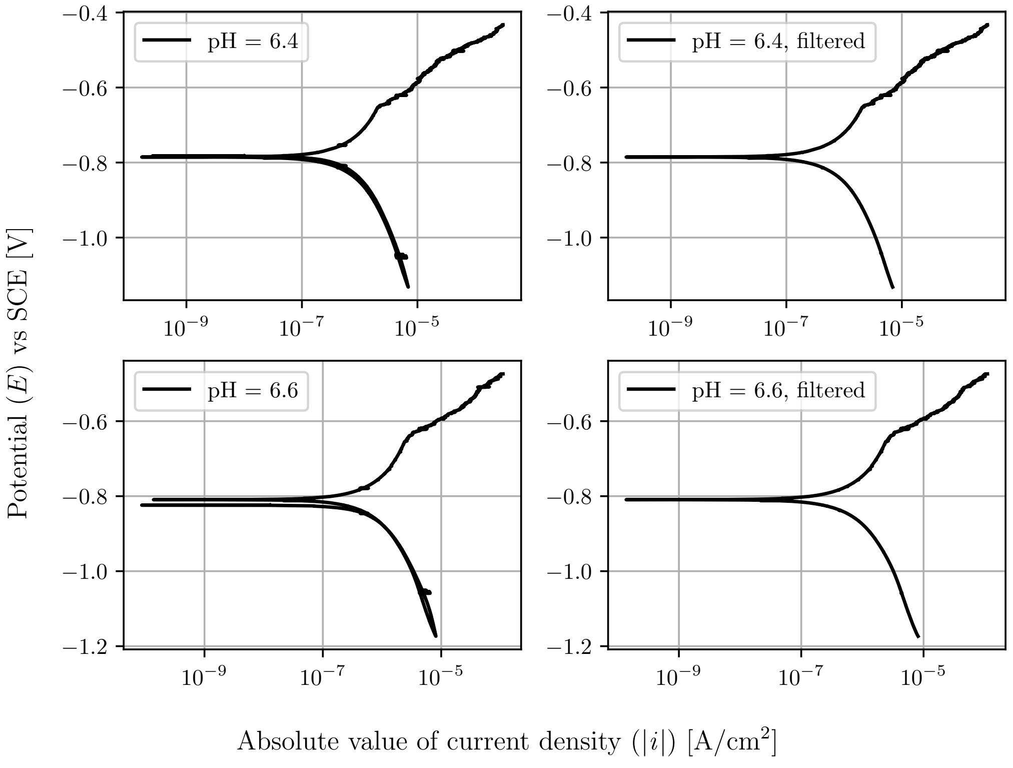
Figures E9: Empirical polarization curve #9.



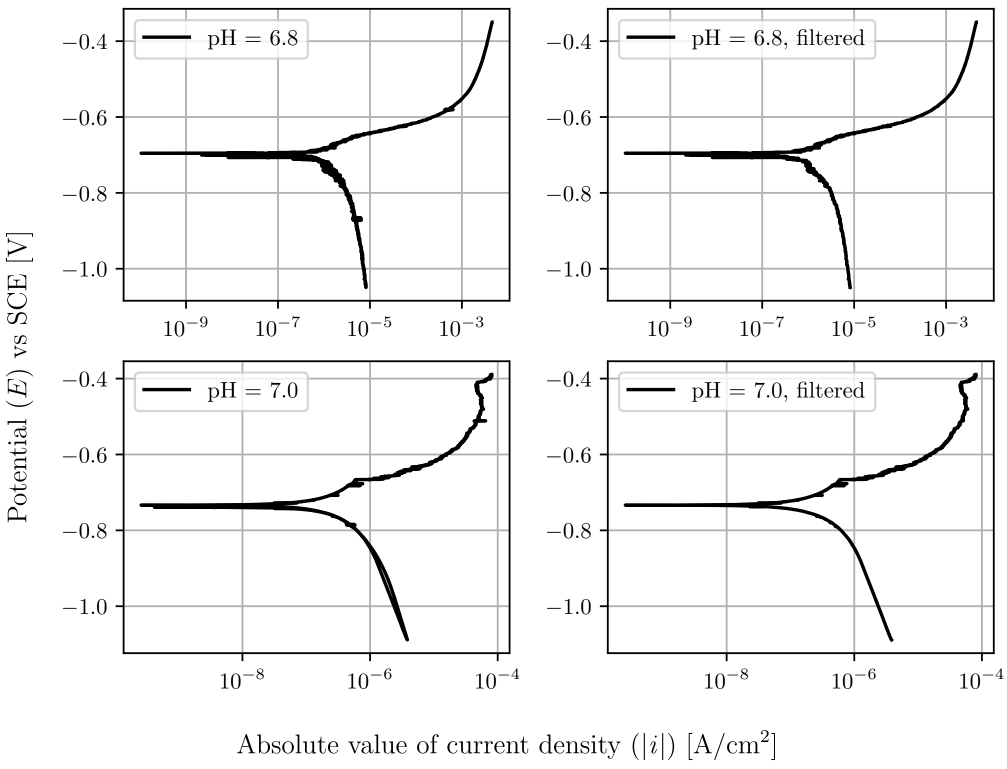
Figures E10: Empirical polarization curve #10.



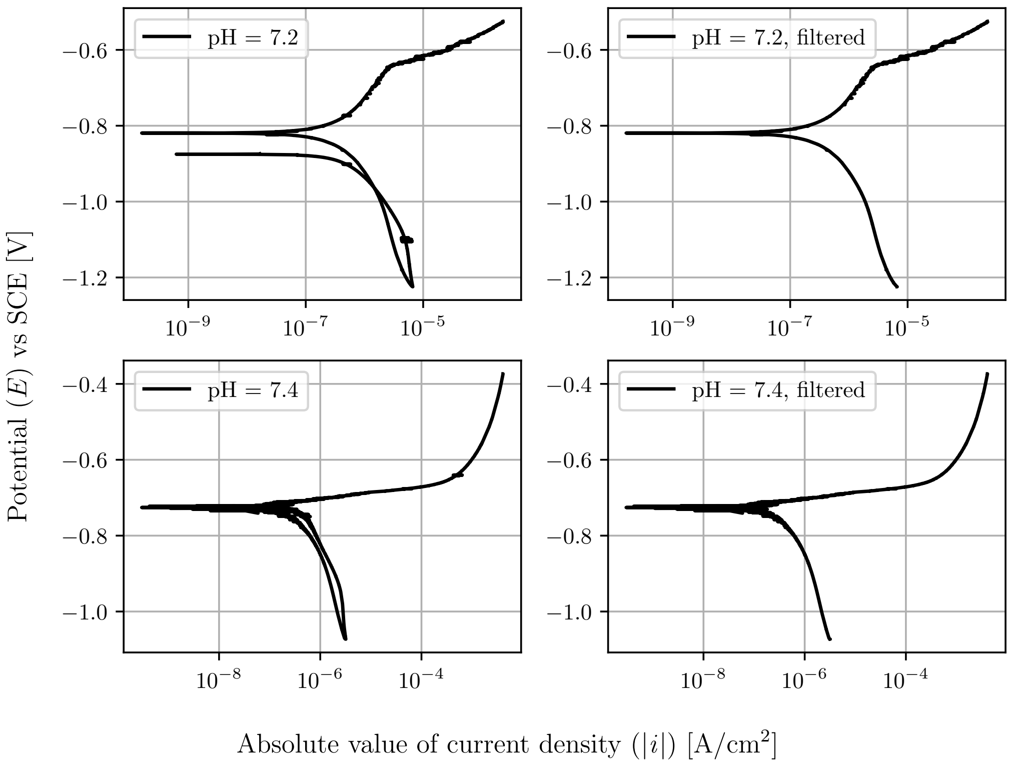
Figures E11: Empirical polarization curve #11.



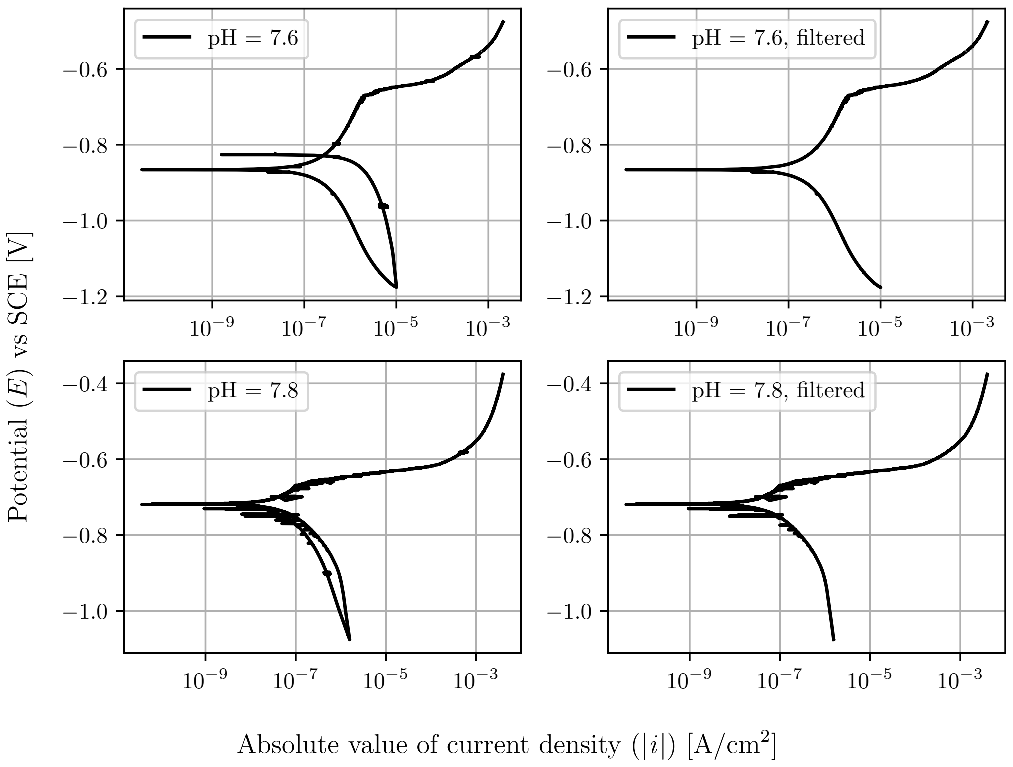
Figures E12: Empirical polarization curve #12.



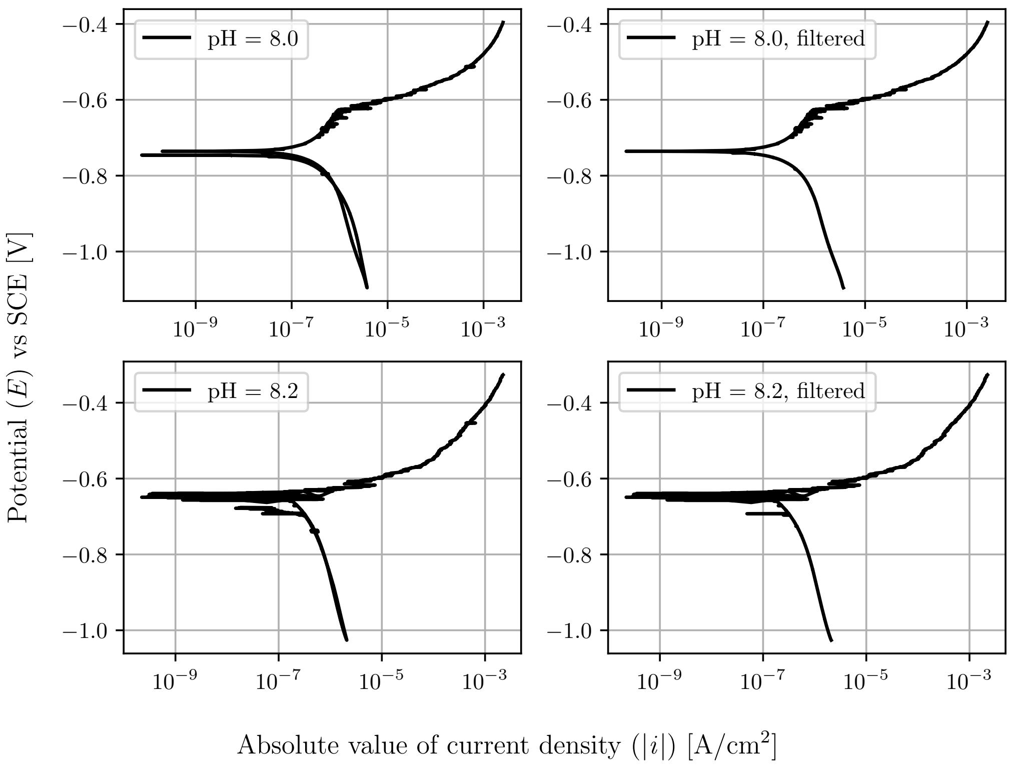
Figures E13: Empirical polarization curve #13.



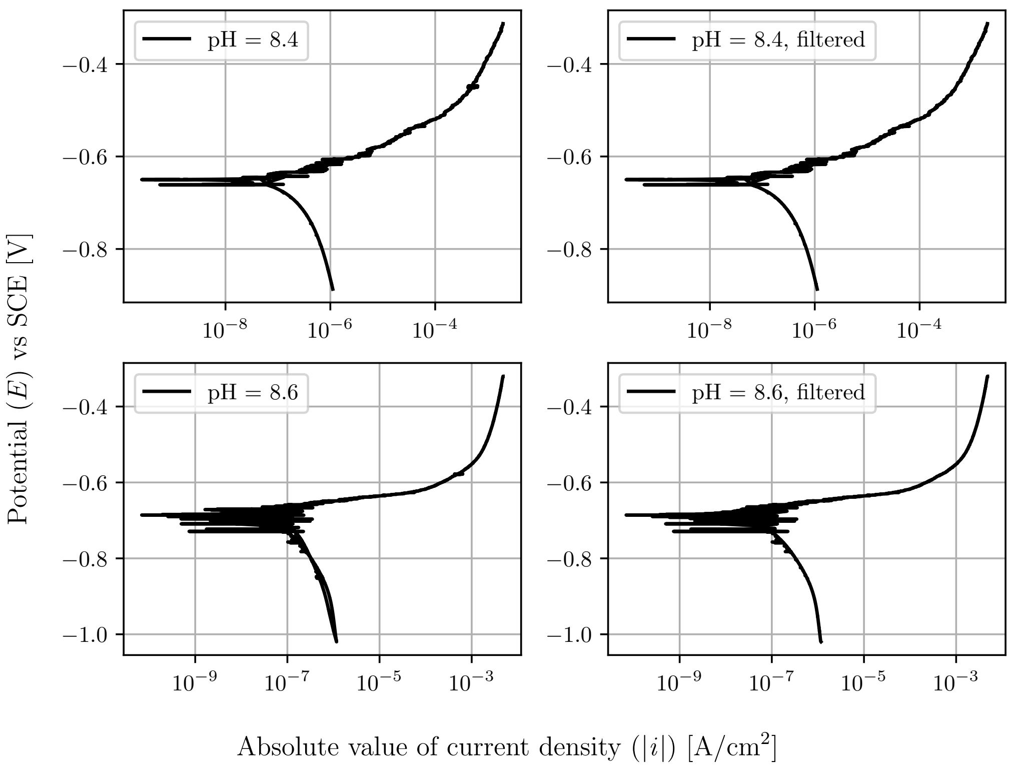
Figures E14: Empirical polarization curve #14.



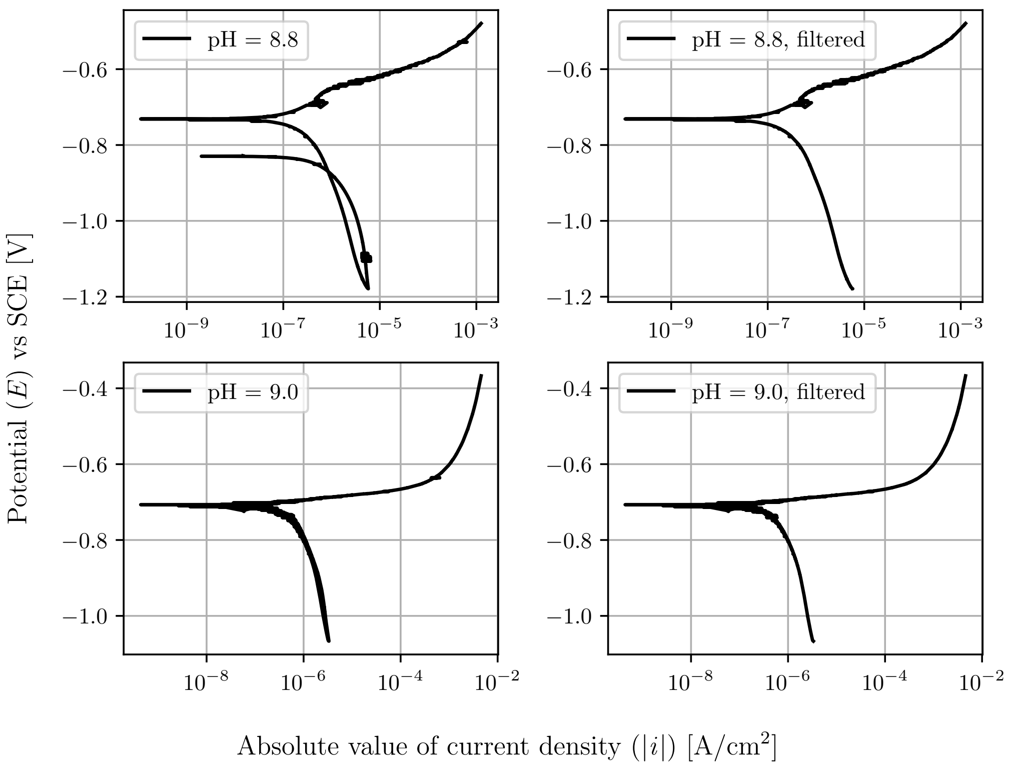
Figures E15: Empirical polarization curve #15.



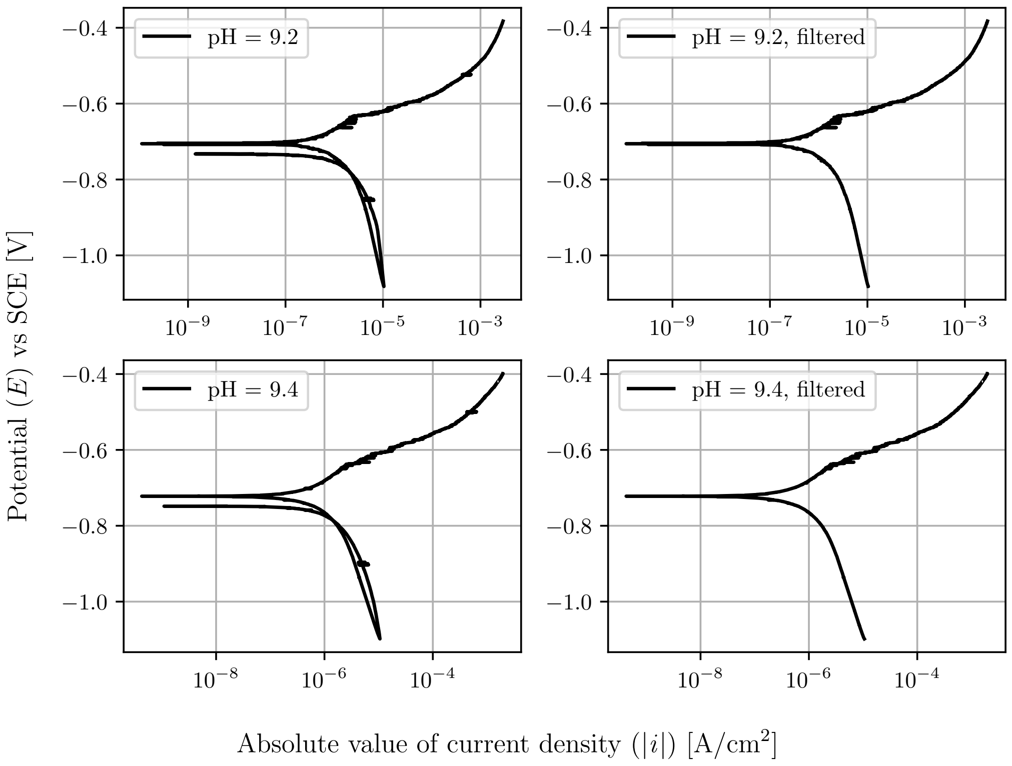
Figures E16: Empirical polarization curve #16.



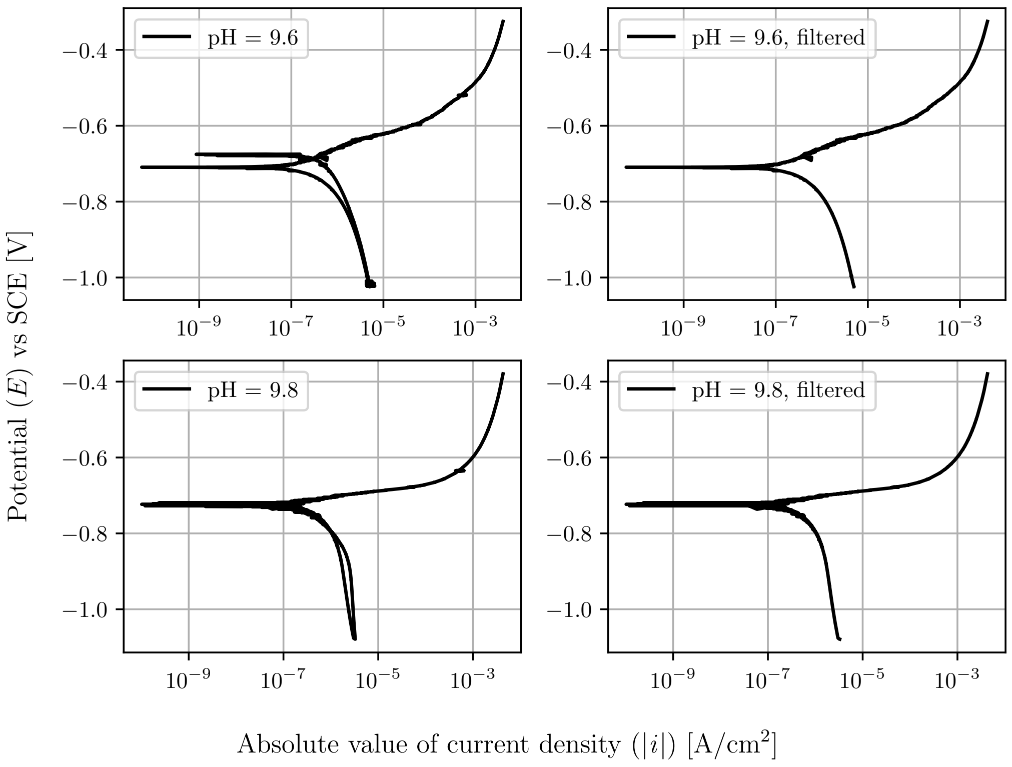
Figures E17: Empirical polarization curve #17.



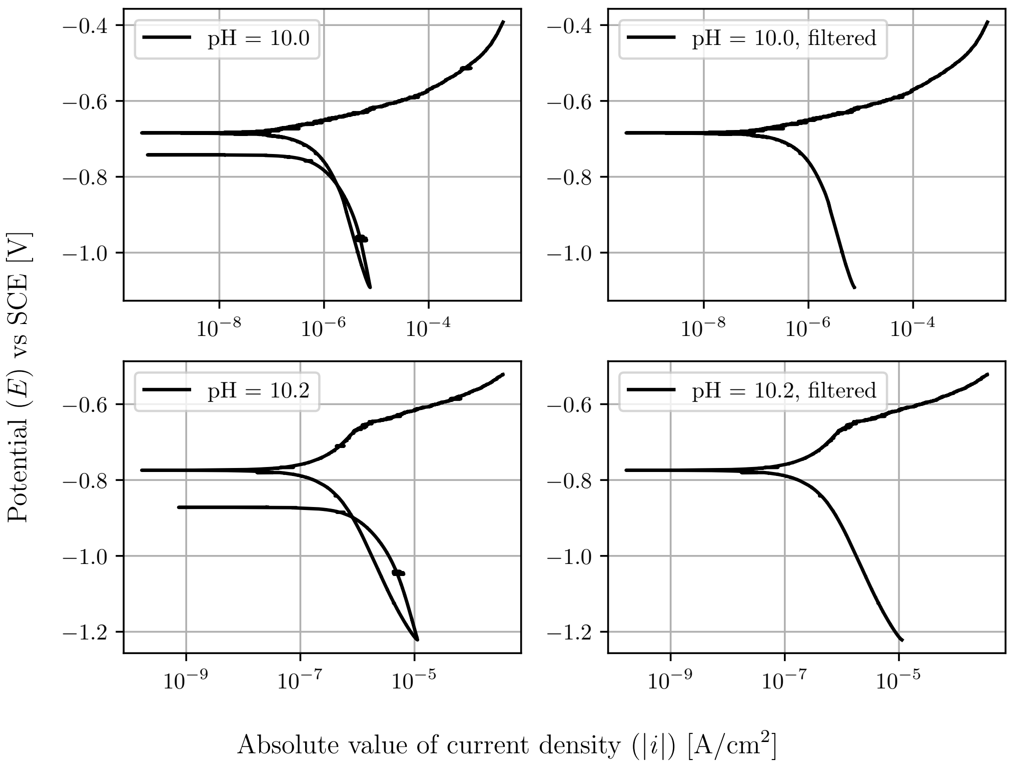
Figures E18: Empirical polarization curve #18.



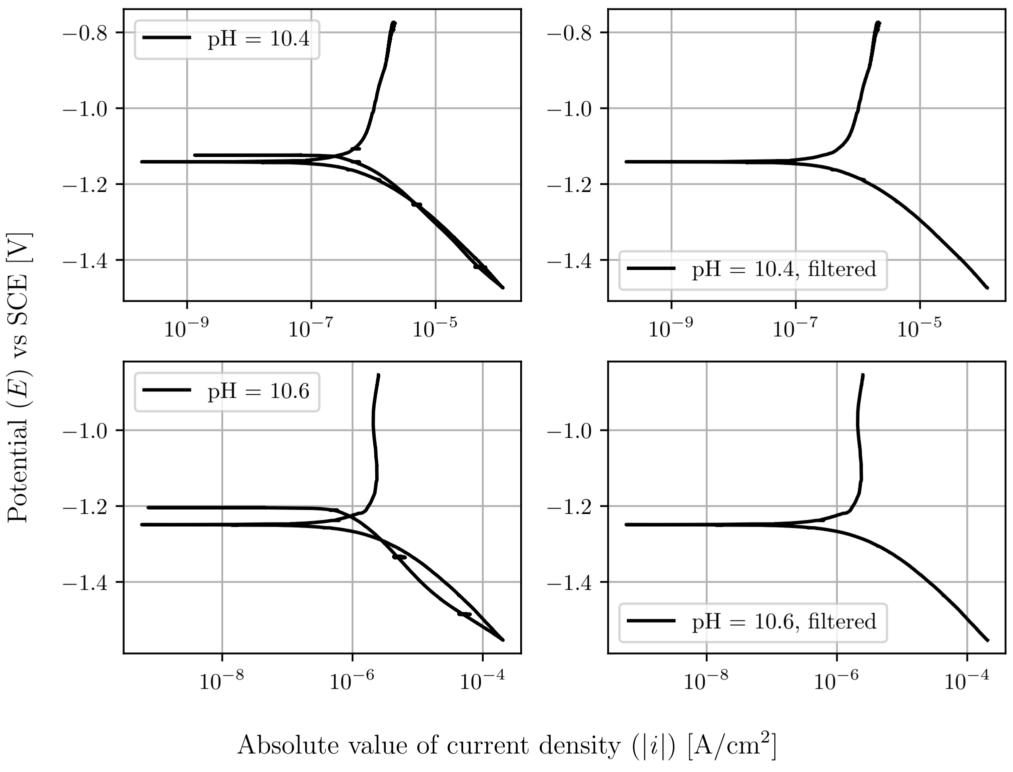
Figures E19: Empirical polarization curve #19.



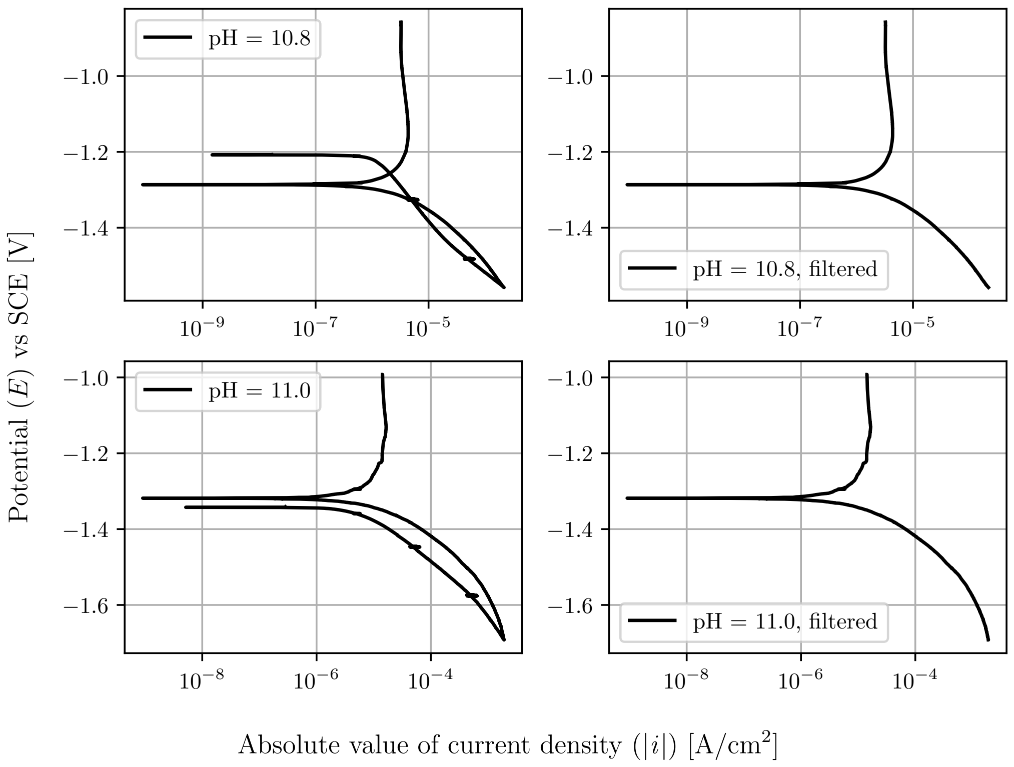
Figures E20: Empirical polarization curve #20.



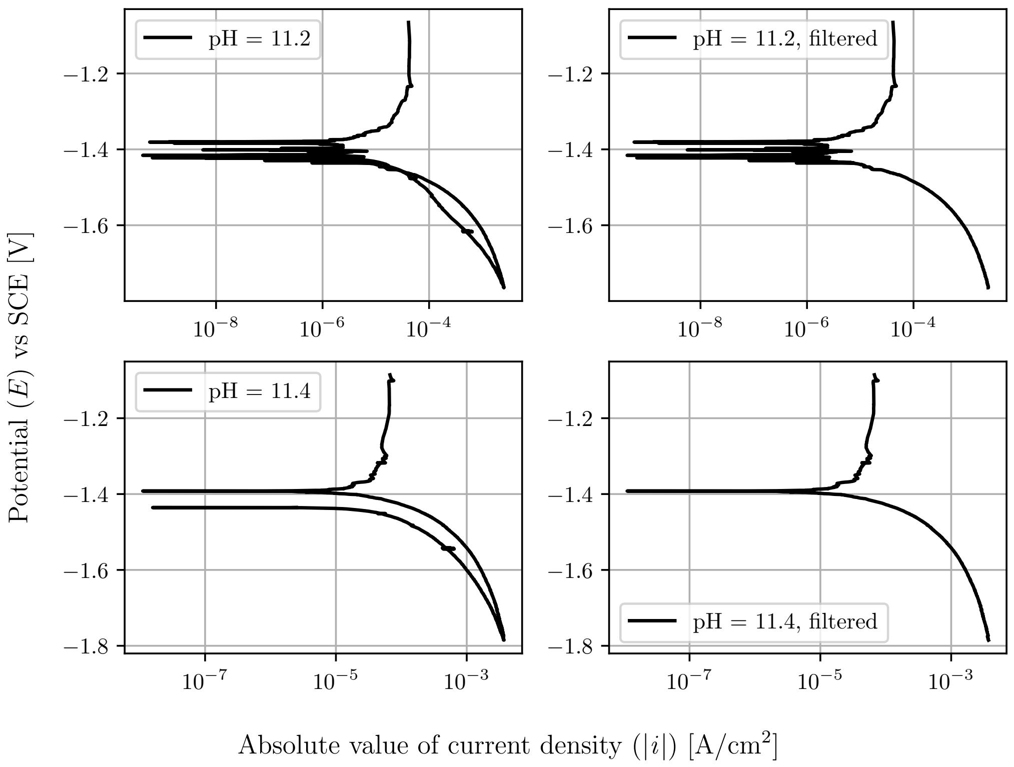
Figures E21: Empirical polarization curve #21.



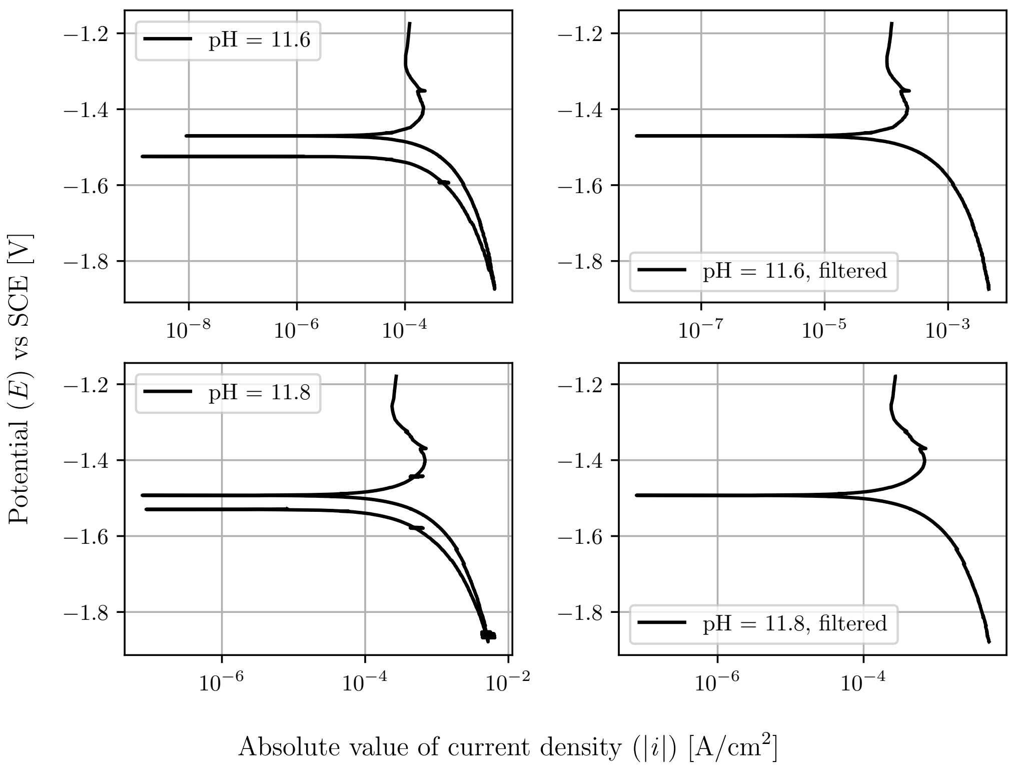
Figures E22: Empirical polarization curve #22.



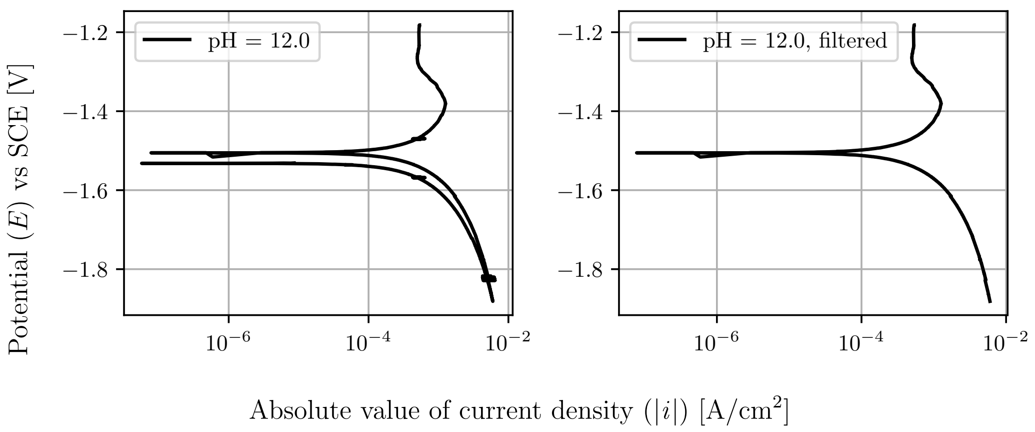
Figures E23: Empirical polarization curve #23.



Figures E24: Empirical polarization curve #24.



Figures E25: Empirical polarization curve #25.



Figures E26: Empirical polarization curve #26.

F Corrosion potential measurements

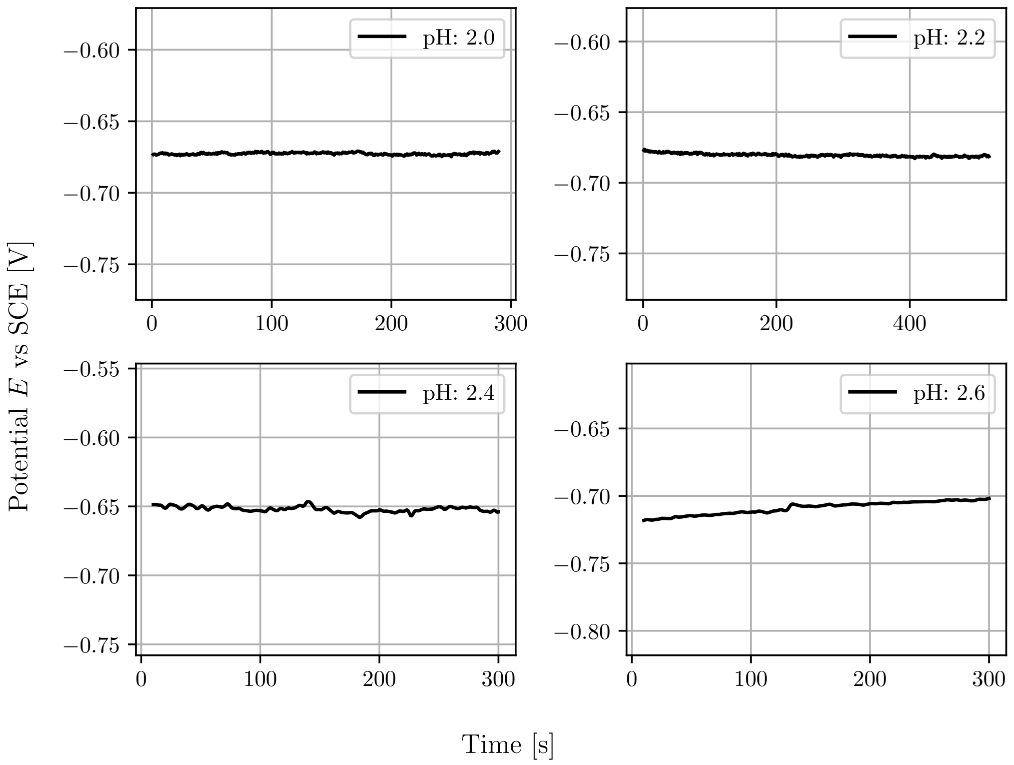
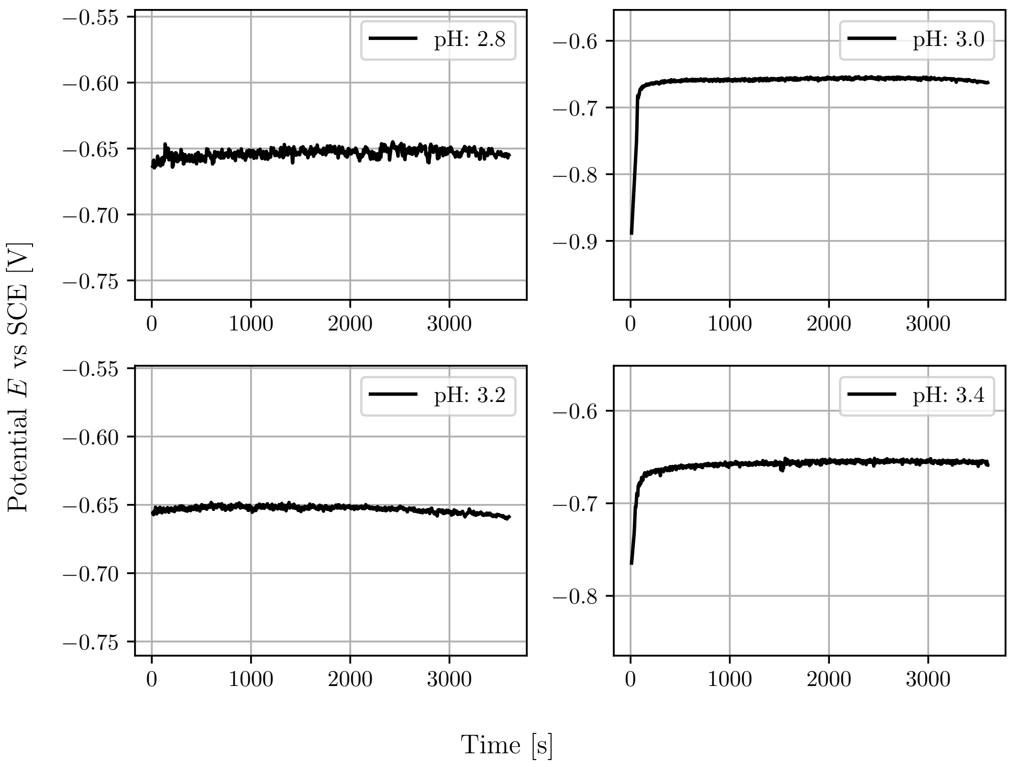
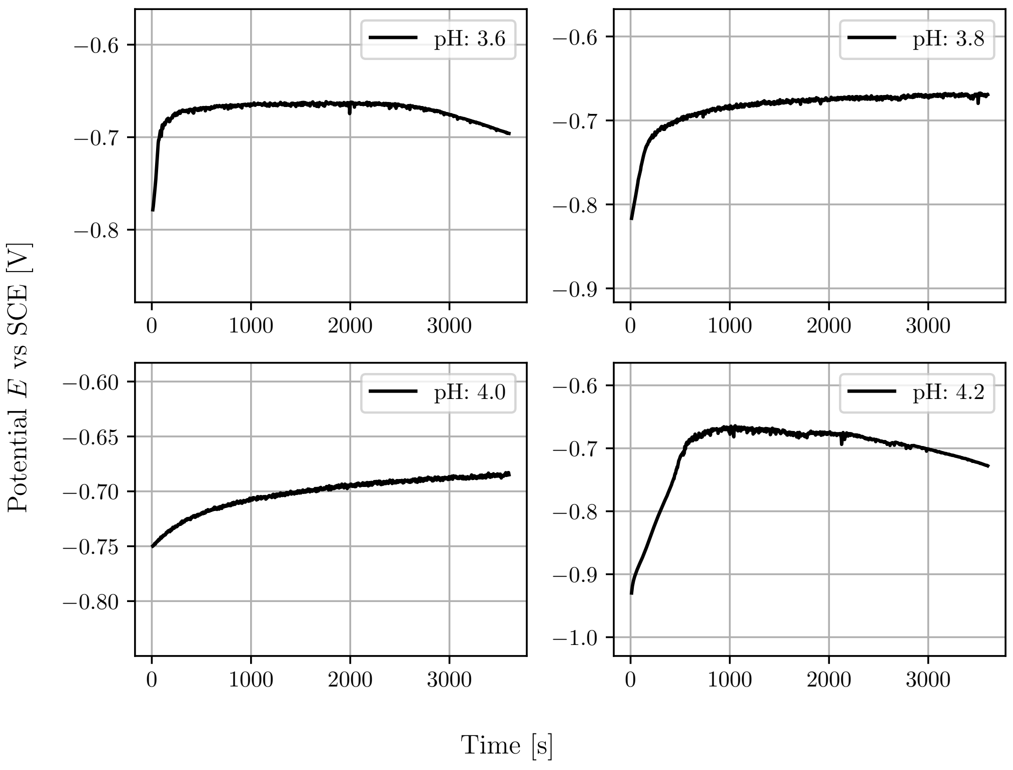


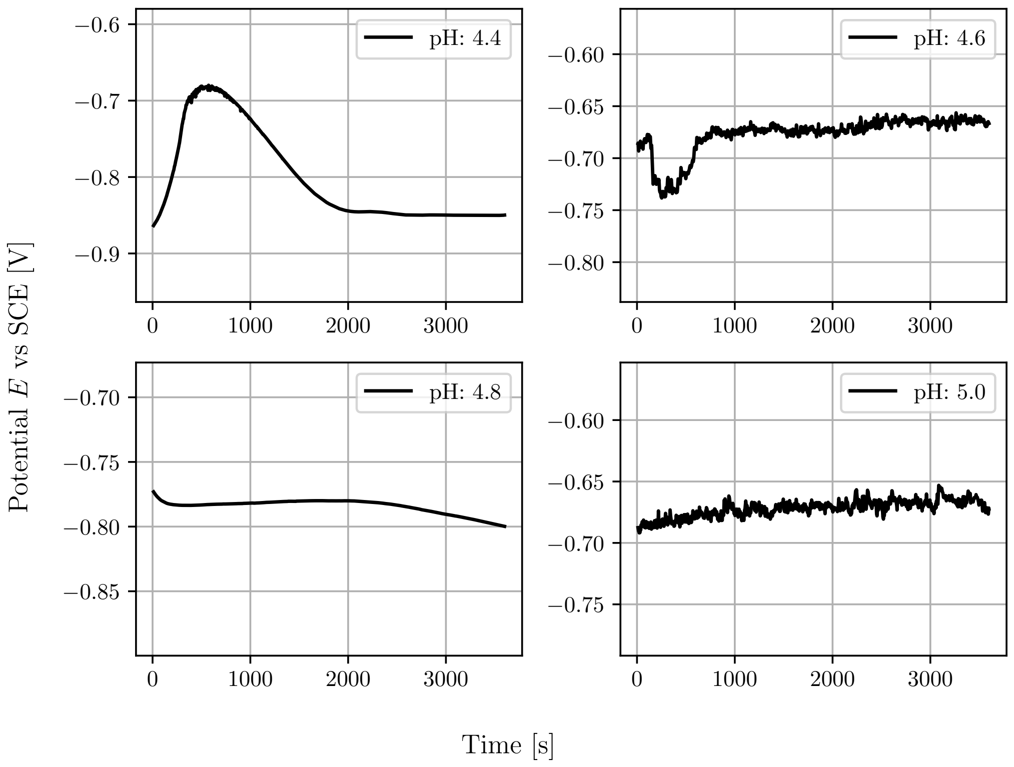
Figure F1: Empirical corrosion potential measurement #1.



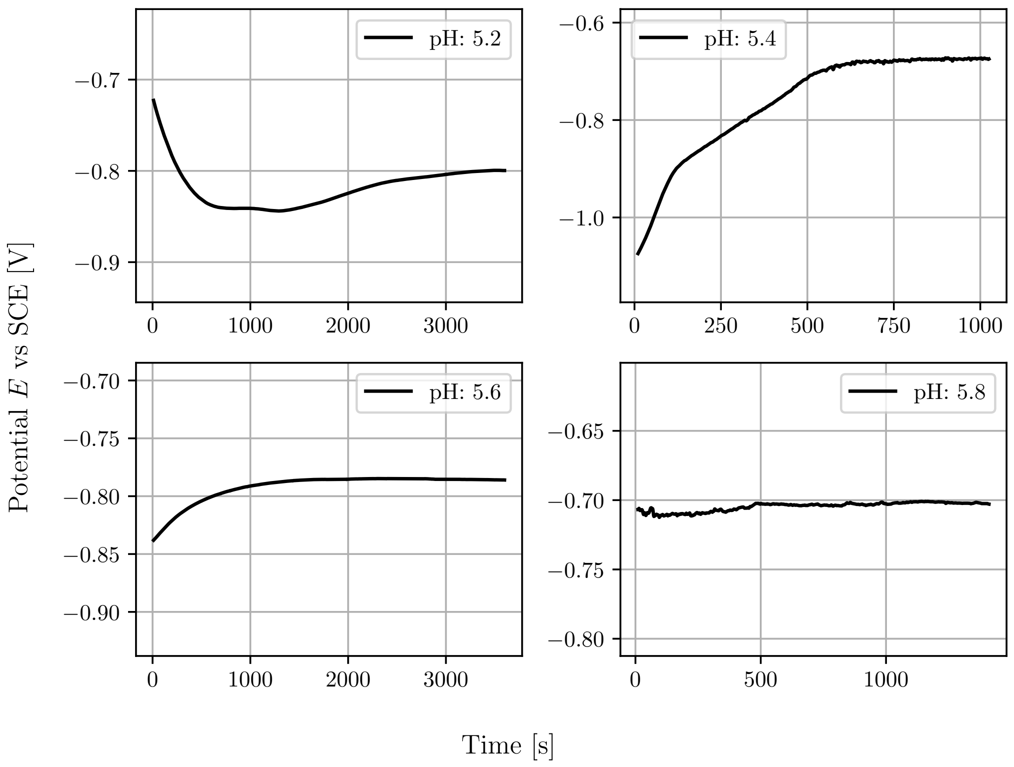
Figures F2: Empirical corrosion potential measurement #2.



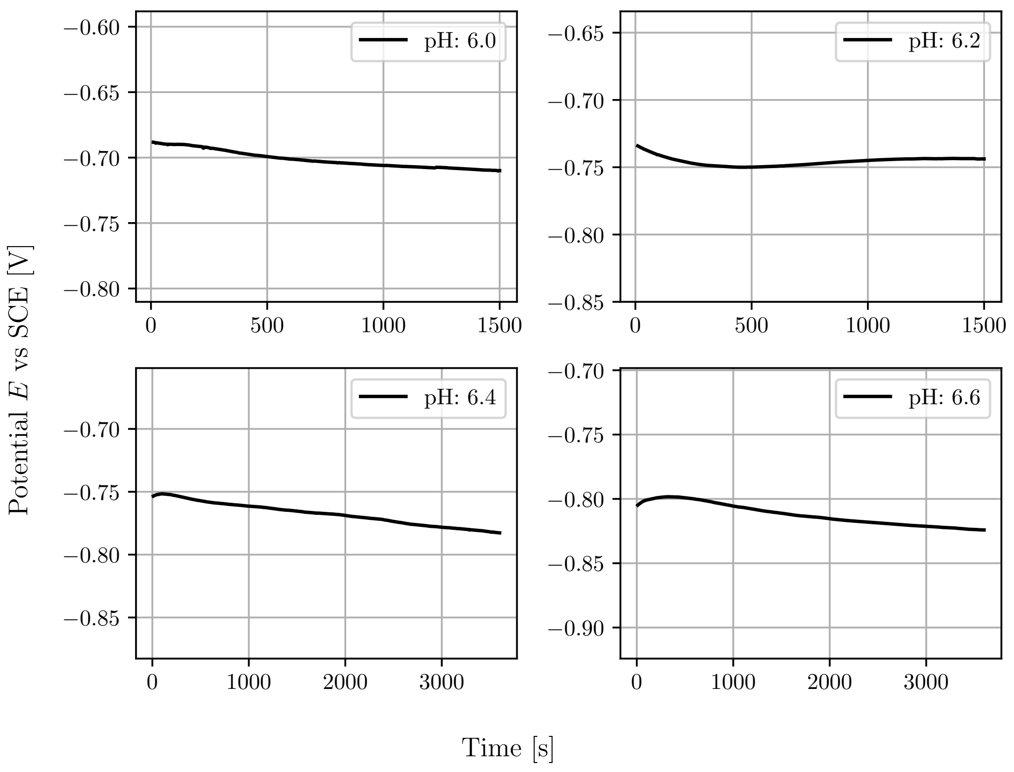
Figures F3: Empirical corrosion potential measurement #3.



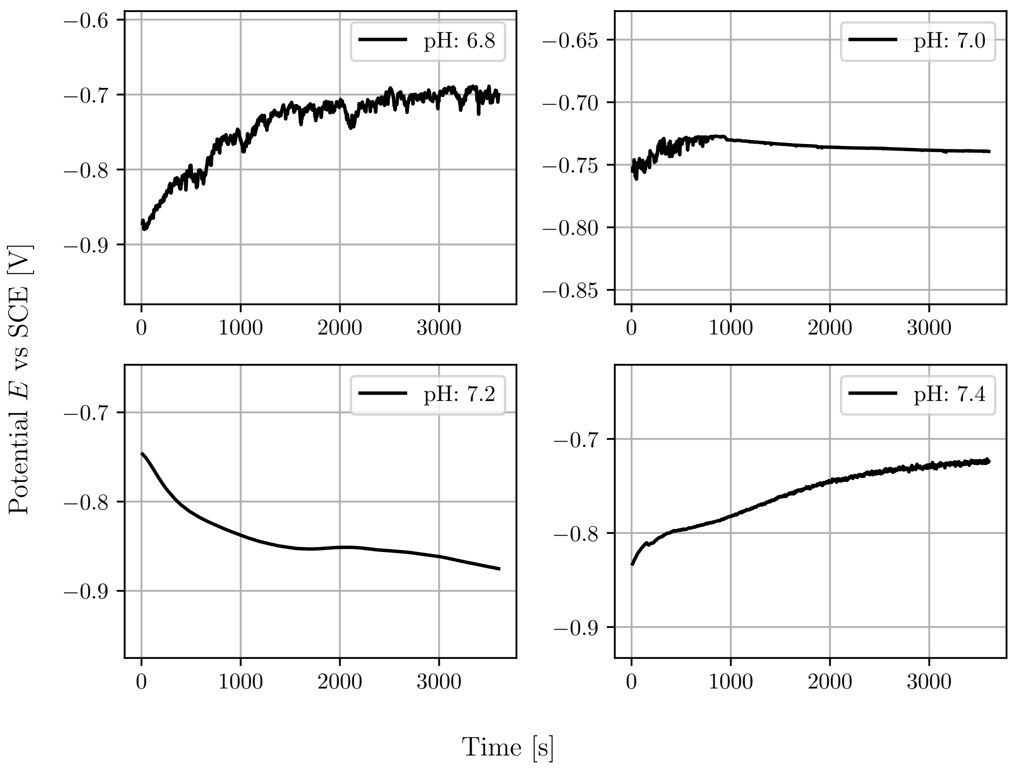
Figures F4: Empirical corrosion potential measurement #4.



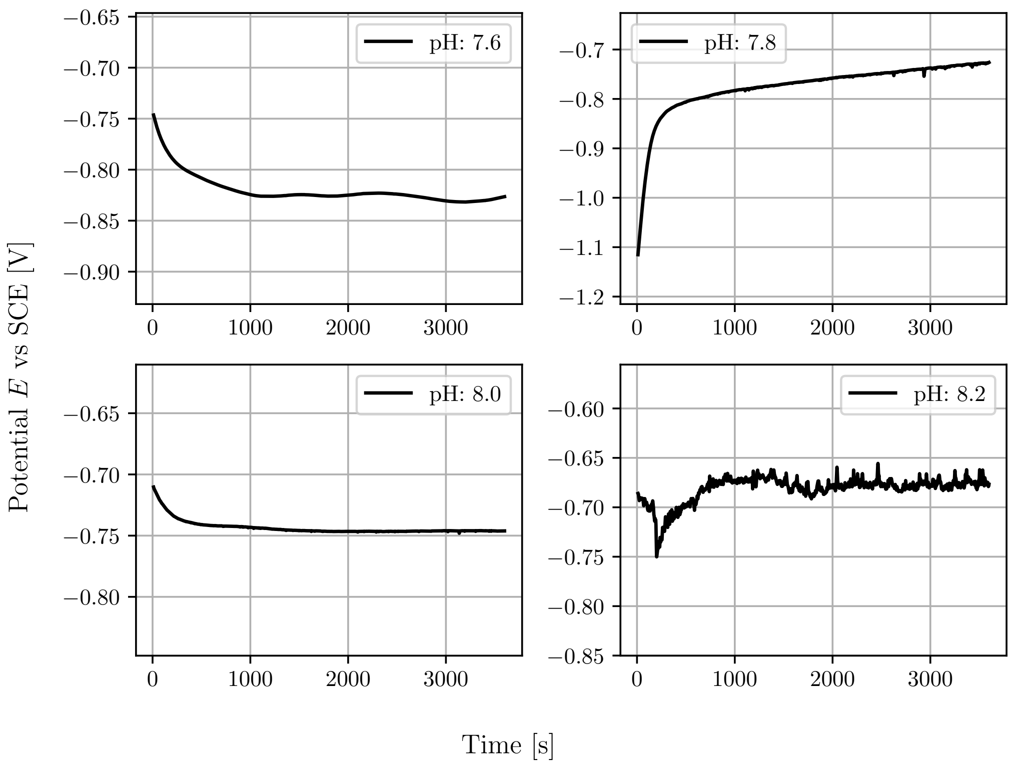
Figures F5: Empirical corrosion potential measurement #5.



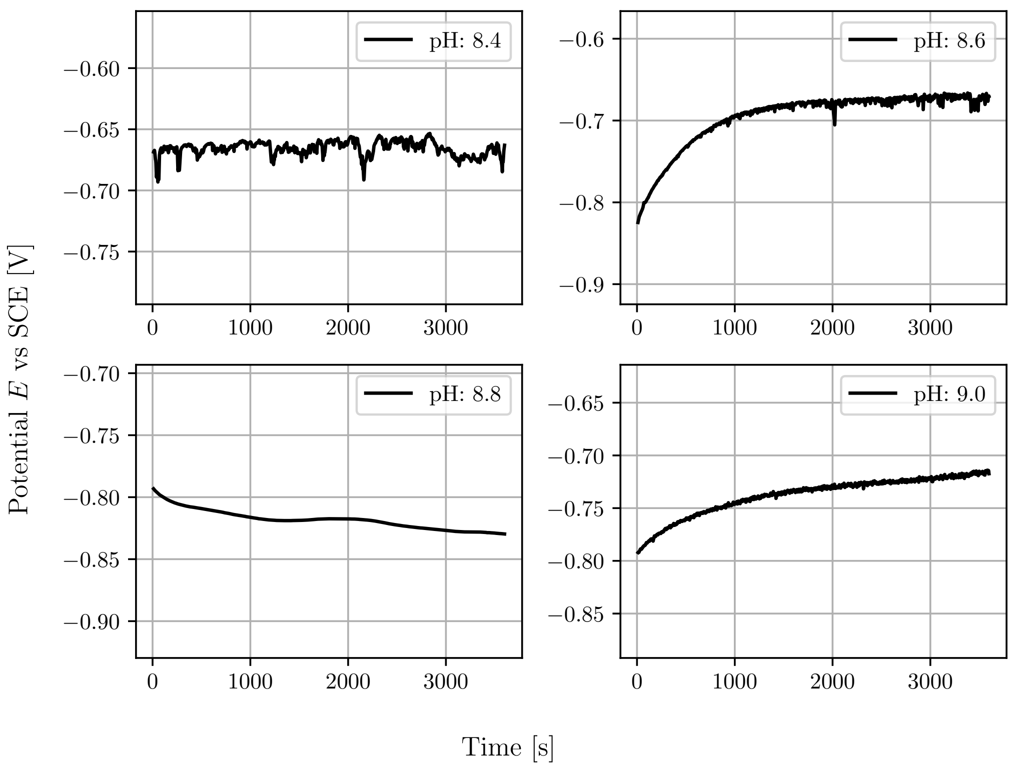
Figures F6: Empirical corrosion potential measurement #6.



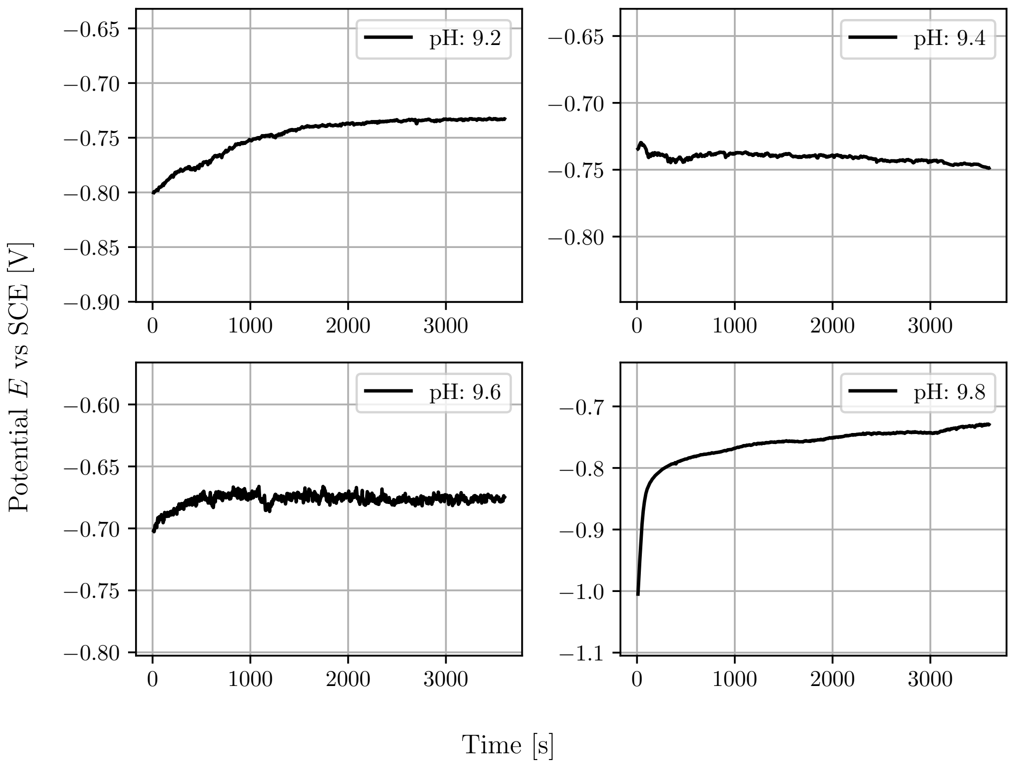
Figures F7: Empirical corrosion potential measurement #7.



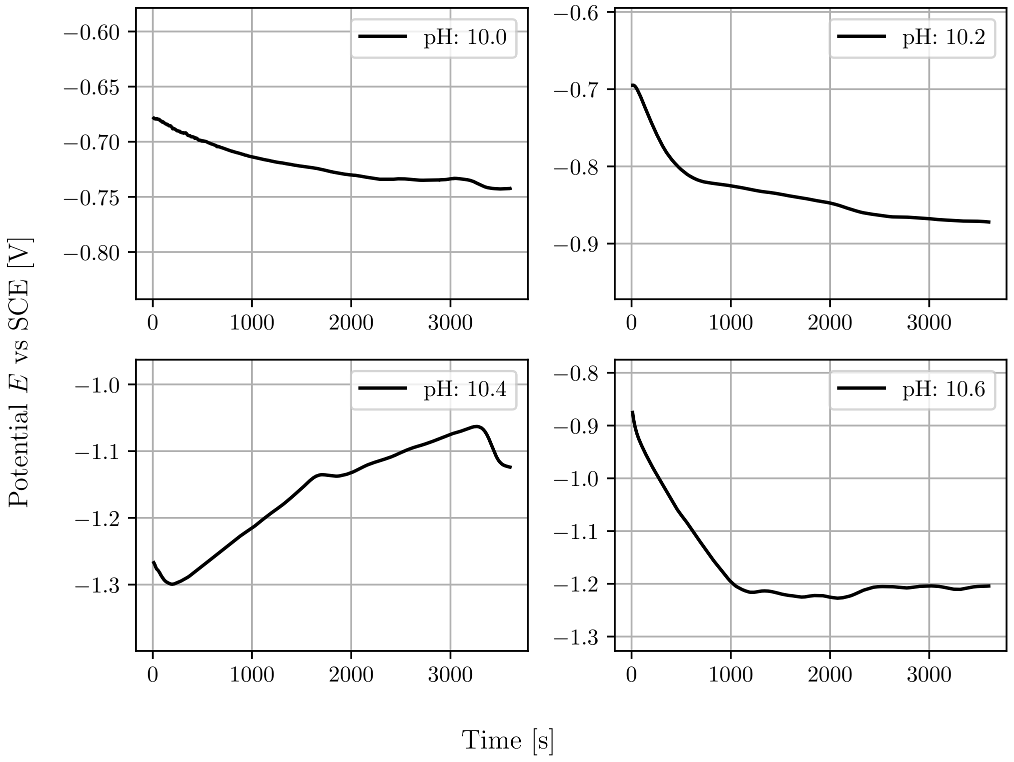
Figures F8: Empirical corrosion potential measurement #8.



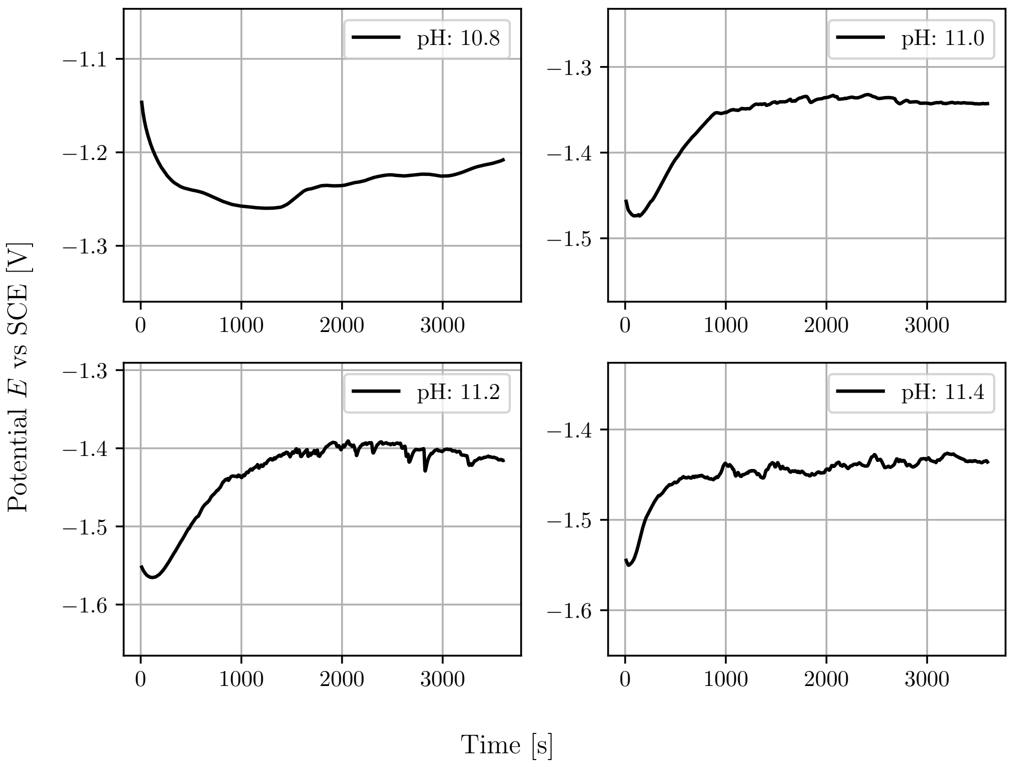
Figures F9: Empirical corrosion potential measurement #9.



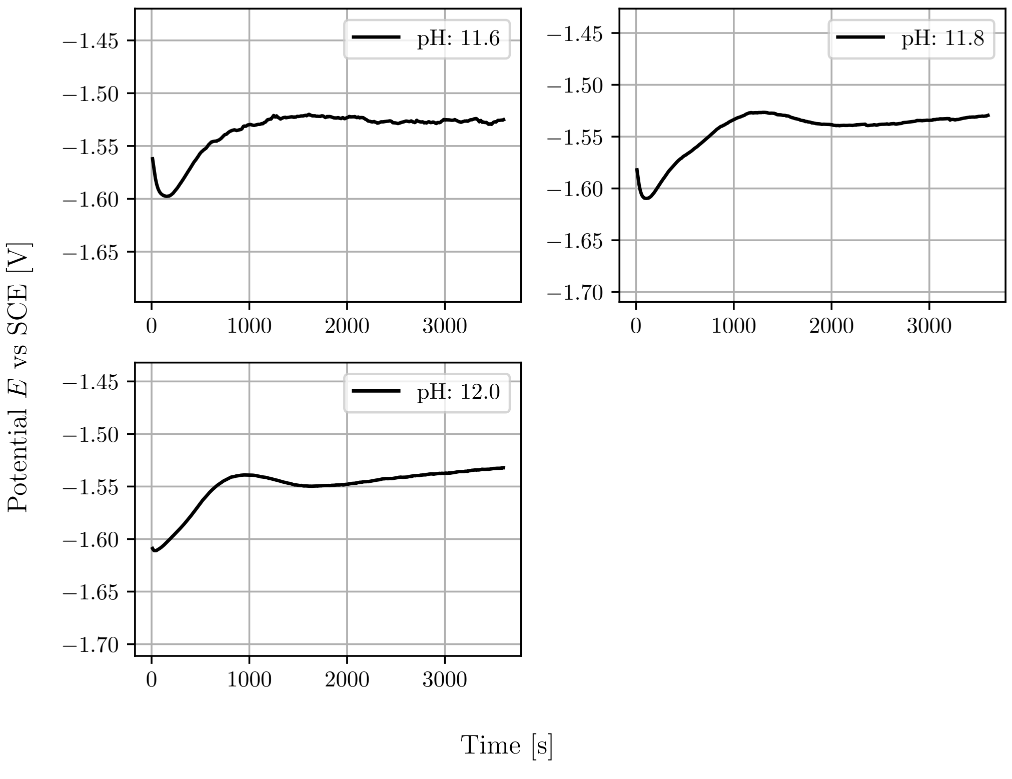
Figures F10: Empirical corrosion potential measurement #10.



Figures F11: Empirical corrosion potential measurement #11.



Figures F12: Empirical corrosion potential measurement #12.



Figures F13: Empirical corrosion potential measurement #13.