

Super Learner Implementation in Corrosion Rate Prediction

**by
Joshua Ighalo**

B.Eng., Federal University Oye-Ekiti, 2018

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Sciences

© Joshua Ighalo 2021
SIMON FRASER UNIVERSITY
Spring 2021

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.

Declaration of Committee

Name: Joshua Ighalo
Degree: Master of Applied Science
Title: Super Learner Implementation in Corrosion Rate Prediction

Committee: **Chair: Anita Tino**
Lecturer, Engineering Science

Bonnie Gray
Supervisor
Professor, Engineering Science

Ryan D’Arcy
Committee Member
Professor, Engineering Science

Andrew Rawicz
Examiner
Professor, Engineering Science

Abstract

This thesis proposes a new machine learning model for predicting the corrosion rate of 3C steel in seawater. The corrosion rate of a material depends not just on the nature of the material but also on the material's environmental conditions. The proposed machine learning model comes with a selection framework based on the hyperparameter optimization method and a performance evaluation metric to determine the models that qualify for further implementation in the proposed models' ensembles architecture. The major aim of the selection framework is to select the least number of models that will fit efficiently (while already hyperparameter-optimized) into the architecture of the proposed model. Subsequently, the proposed predictive model is fitted on some portion of a dataset generated from an experiment on corrosion rate in five different seawater conditions. The remaining portion of this dataset is implemented in estimating the corrosion rate.

Furthermore, the performance of the proposed models' predictions was evaluated using three major performance evaluation metrics. These metrics were also used to evaluate the performance of two hyperparameter-optimized models (Smart Firefly Algorithm and Least Squares Support Vector Regression (SFA-LSSVR) and Support Vector Regression integrating Leave Out One Cross-Validation (SVR-LOOCV)) to facilitate their comparison with the proposed predictive model and its constituent models. The test results show that the proposed model performs slightly below the SFA-LSSVR model and above the SVR-LOOCV model by an RMSE score difference of 0.305 and RMSE score of 0.792. Despite its poor performance against the SFA-LSSVR model, the super learner model outperforms both hyperparameter-optimized models in the utilization of memory and computation time (graphically presented in this thesis).

Keywords: corrosion rate; super learner; selection framework; performance evaluation metrics; adaptive corrosion protection system

Acknowledgments

In these uncertain times, it is predominantly essential to thank all of those who have helped me research and write this thesis one way or the other.

To my supervisors, Dr. Bonnie Gray, Dr. Bozena Kaminska, and Dr. Ryan D'Arcy, thank you for your guidance, support, supervision, patience, mentorship, and constant encouragement, especially during research writing my thesis. Working on this research under your supervision was a fantastic experience. The freedom to implement my ideas and willingness to correct me immediately the study was drifting away from its objectives.

Thank you for your love, friendship, and support to my labmates from Ciber Labs, Micro-Mob, and BrainNet. Thank you for your patience, especially with catching up on every research I participated in.

To my parents, Elder James & Bosede Ighalo, and my siblings, thank you for your constant support, encouragement, push, and prayers all through. Thank you for all the fun, long-distance calls to ease the stress.

To my partner, Oreoluwa Olafisan, thank you for being a blessing in my life. Thank you for understanding my crazy work schedules, being patient, giving me a shoulder to rest on during the days I get burnt out, listening to my crazy ideologies and opinions on machine learning. You are next in line for this level of degree, and I'll be fully behind you to provide more than the support you have given me.

Ultimately, I say thank you, Jesus Christ, for the wisdom and guidance you imparted throughout this program. Without you, I wouldn't even be existing let alone seen this program through to this point. I am forever grateful for your unconditional love.

Table of Contents

| | |
|---|-----------|
| Declaration of Committee..... | ii |
| Abstract..... | iii |
| Acknowledgments..... | iv |
| Table of Contents..... | v |
| List of Tables..... | vii |
| List of Figures..... | viii |
| List of Acronyms..... | ix |
| Chapter 1. Introduction..... | 1 |
| 1.1. Contributions to the field of corrosion rate research | 2 |
| 1.2. Objectives | 2 |
| 1.3. Limitations | 3 |
| 1.4. Thesis Outline | 3 |
| Chapter 2..... | 4 |
| Background..... | 4 |
| 2.1. 3C Steel | 4 |
| 2.2. Seawater Corrosion and its effects | 4 |
| 2.3. Metamorphosis of Corrosion Rate Estimation..... | 8 |
| 2.3.1. Conventional methods | 9 |
| 2.3.2. Utilization of Machine Learning in Corrosion Rate Prediction | 10 |
| 2.4. Super Learner and its Constituents..... | 12 |
| 2.4.1. Linear Regression | 14 |
| 2.4.2. Ridge Regression..... | 15 |
| 2.4.3. Lasso Regression | 16 |
| 2.4.4. Support Vector Regression | 17 |
| 2.4.5. Extra Gradient Boosting Machine (XGBM) | 19 |
| 2.5. Random Search Hyperparameter Optimization | 20 |
| 2.6. Computation time and Memory Consumption Optimization | 22 |
| Chapter 3..... | 23 |
| Super Learner Model Implementation..... | 23 |
| 3.1. Tools Utilized..... | 23 |
| 3.2. Dataset employed | 25 |
| 3.3. Performance Evaluation Metrics | 27 |
| 3.4. Base Learners Selection Technique | 28 |
| 3.5. Development of the Super Learner Model | 32 |
| 3.6. System Characteristics | 33 |
| Chapter 4..... | 34 |
| Evaluation of the Super Learner Model..... | 34 |
| 4.1. Test Dataset..... | 34 |
| 4.2. Test Results | 35 |

| | |
|--|-----------|
| Chapter 5 | 42 |
| Conclusions..... | 42 |
| 5.1. Summary..... | 42 |
| 5.2. Recommendation for future work..... | 44 |
| References | 45 |
| Appendix | 54 |

List of Tables

| | | |
|---------|--|----|
| Table 1 | System Characteristics | 33 |
| Table 2 | Mean absolute percentage error of the models | 36 |
| Table 3 | Performance evaluation result of the super learner model and its constituent models | 38 |
| Table 4 | Prediction accuracy comparison between the super learner, SFA-LSSVR, and SVR-LOOCV model | 43 |

List of Figures

| | | |
|-----------|---|----|
| Figure 1 | Regions of change on a metal sample | 5 |
| Figure 2 | Corrosion starting from a developed crevice on a metal surface | 7 |
| Figure 3 | Evolution of corrosion rate determination | 8 |
| Figure 4 | Development process of the super learner model | 14 |
| Figure 5 | Layout implemented by random search | 21 |
| Figure 6 | Statistical description of the dataset on the Spyder IDE..... | 25 |
| Figure 7 | Flow chart showing the development of the base learners selection technique | 29 |
| Figure 8 | MSE scores of all ten models in spyder IDE's variable explorer | 31 |
| Figure 9 | Spyder IDE terminal window view of the accepted models | 31 |
| Figure 10 | Flow representation of super learner model fitting process | 32 |
| Figure 11 | feature set and target test dataset..... | 35 |
| Figure 12 | Line plot of the predicted values, actual values, and percentage error of the super learner model..... | 36 |
| Figure 13 | Line plots of extra gradient boosting machine, lasso regression, polynomial kernel of the support vector machine, ridge regression, and linear regression models | 37 |
| Figure 14 | Bar chart showing the root mean square values of the super learner model, SFA-LSSVR model, SVR-LOOCV model, and the constituent models of the super learner model | 39 |
| Figure 15 | Computation time of the super learner model, its constituent models, the SFA-LSSVR model and the SVR-LOOCV model..... | 40 |
| Figure 16 | Memory usage and computation time of the super learner model, its constituent models, the SFA-LSSVR model and the SVR-LOOCV model | 41 |

List of Acronyms

| | |
|------|---|
| ACPS | Adaptive Corrosion Protection System |
| GDP | Gross Domestic Product |
| IDE | Integrated Development Environment |
| MAPE | Mean Absolute Percentage Error |
| MSE | Mean Square Error |
| NACE | National Association of Corrosion Engineers |
| RMSE | Root Mean Square Error |
| SVC | Support Vector Classification |
| SVM | Support Vector Machine |
| XGBM | Extreme Gradient Boosting Machine |

Chapter 1.

Introduction

Corrosion is defined as an irreversible interfacial reaction of a material with its environment, which results in the degradation of the material [1,2]. It is the gradual destruction of materials (usual metals) by chemical and electrochemical reactions with their environment. Corrosion is a critical factor considered in the design phase of infrastructures, not only for its implications in structural resistance but also for its importance in economic calculation. It implies costly maintenance in the stage of operation [3,4]. In 1949, corrosion's cost was equivalent to 2.5% of the U.S. Gross Domestic Product (GDP) [5,6]. In 1998, the total annual direct cost of corrosion as economic losses in the U.S. was broken down into five specific industries: \$22.6 billion in infrastructure; \$17.6 billion in production and manufacturing; \$29.7 billion in transportation; \$20.1 billion in government; and \$47.9 billion in utilities. In total, the sum of these costs is up to \$276 billion (ca. 3.2% of the US gross domestic product) [7]. In 2016, The National Association of Corrosion Engineers (NACE) International published the "International Measures of Prevention, Application, and Economics of Corrosion Technology (IMPACT)" that approximates global corrosion cost to be US\$2.5 trillion, which is comparable to roughly 3.4 percent of the global Gross Domestic Product (GDP) in USD [8].

Corrosion-prone materials employed in the development of structures are regularly monitored, maintained, and replaced due to corrosion. An example of an infrastructure prone to corrosion is pipelines. Due to their large capacities, pipelines carry about 70% of fluids than the carrying capacity on land by roads and rails [9]. Corrosion of these structures represents severe environmental and economic problems. It leads to continual wearing and reducing pipelines' walls, causing leakage of potentially non-environment-friendly fluids to the environment. Furthermore, explosions are triggered when fluids with flashpoints at or below nominal temperatures meet air at ambient temperature, resulting in devastating accidents.

This thesis aims at developing a unique variant of a super learner machine learning model suitable for effectively predicting the corrosion rate of 3C steel in seawater at a high degree of accuracy & efficiency while consuming less computation time and memory. Five

supervised learning algorithms are implemented in building the super learner model to facilitate its estimation of the corrosion rate of a sample of 3C steel metal sample in seawater. The model's performance is evaluated and employed to compare (in terms of the corrosion rate prediction capability) with other models developed on the same dataset. This thesis not only covers a proof-of-concept for accurate corrosion rate prediction using machine learning but is expected to further serve as the foundation for the implementation of machine learning in corrosion protection studies and adaptive corrosion protection applications, respectively.

1.1. Contributions to the field of corrosion rate research

Research in corrosion rate measurement has advanced greatly over the years, especially with machine learning. The dataset employed in this thesis has been used in the past to develop machine learning models to predict the corrosion rate of 3C steel metal samples [10,11]. These models have successfully provided corrosion rate predictions but with limitations in memory consumption and computation time. This thesis builds on these limitations by generating corrosion rate predictions of 3C steel metal sample in different seawater conditions via the following contributions:

- i. Development of a machine learning model that not only predicts the corrosion rate of the 3C steel metal sample at a high degree of accuracy but does so while efficiently utilizing memory at less computation time.
- ii. Development of a machine learning model possessing a generalization comparable with the generalization of the models of previous works developed on the same dataset.
- iii. Development of a system that can estimate the corrosion rate that the ACPS can utilize to protect electrical tower grillages erected under seawater.

1.2. Objectives

The research's overall objective is to develop a machine learning model that accurately predicts 3C steel metal samples' corrosion rate under different environmental conditions. To meet this overall objective, the following sub-objectives must also be met:

- i. Acquire a satisfactory amount of data from which the training and test datasets are extracted.
- ii. Develop a framework that will help efficiently and technically unique machine learning algorithms that will best suit the super learner model's architecture for predicting the corrosion rate of 3C steel metal samples.
- iii. Evaluate and compare the super learner model's performance with other models using R-Squared, Root Mean Square Error (RMSE), and Mean Squared Error (MSE).
- iv. Evaluate the memory consumption and computation time of the model with other models developed on the same dataset.

1.3. Limitations

This thesis's primary limitation is the lack of sufficient training data to meet a near-perfect prediction accuracy. This is due to the loss and misplacement of the majority of the equipment required to replicate the 3C steel metal corrosion rate experiment during the relocation of CIBER lab due to SFU Applied Sciences Building renovations.

1.4. Thesis Outline

This thesis is presented as follows:

Chapter I covers the contributions, objectives, and limitations of the thesis.

Chapter II motivates the specific research objectives, summarizes the relevant literature and the recent works related to the study.

Chapter III covers the materials and methodologies employed in the thesis. The study parameters and test methods are given briefly in this chapter.

Chapter IV covers the results and discussions in which the analysis of test results, tables, and figures are presented.

Chapter V covers the conclusion and recommendations of the work

Chapter 2.

Background

Corrosion is a phenomenon that has had very devastating effects over the years in different parts of the world and other areas of the engineering industry. Aside from structural and financial losses, lives have also been lost due to corrosion-fueled disasters. These losses can be prevented through corrosion rate estimates since corrosion rate has become a key parameter implemented in corrosion protection research.

2.1. 3C Steel

3C steel is a form of carbon steel with carbon content ranging from 2.8% to 3.2% [12]. It is the most popular engineering structural material implemented largely in offshore engineering especially in the construction of underwater pipelines [13]. Other fields where it finds application includes, chemical processing, petroleum production and refining, and pipelines, etc. This variant of carbon steel not only possesses good mechanical, physical, and chemical properties, but also possesses characteristics, such as plastic, toughness, welding performance, stamping, and cutting performance, etc. [14]. With rise in offshore applications this form of steel finds itself as well as its increasing tonnages, the need for corrosion monitoring is paramount.

2.2. Seawater Corrosion and its effects

Corrosion is aggressive on metals regardless of its environment (commonly land (soil) and seawater), leaving behind devastating effects. Over the years, corrosion has led to catastrophic impacts, especially on structures built for land services.

In Appomattox, Virginia, a 30-inch natural gas pipeline failed in 2008, thereby causing an explosion that led to families being evacuated, roads closed, destruction of houses, and injured persons [15]. Elsewhere, investigations showed that pitting corrosion on the external walls of a 24-inch pipeline in Clark County, Kentucky, led to the emission of approximately 43,000 MSCF of natural gas. Although there were no deaths or injuries recorded, damages occurred to homes and properties due to the explosion caused by gas emission [16]. Massive property damage worth \$25,000 (per household) and more was

recorded due to the explosion resulting from corrosion wearing out 75% of the natural gas pipeline wall. The Missouri pipeline underwent failure, which led to approximately 13.5 million CF of natural gas [17]. In 2010, another incident was caused by corrosion in Green River, Wyoming. Pressure in the corroded pipeline caused the pipe to rupture and spill about 84,000 gallons of crude oil. The oil leaked into an irrigation ditch and contaminated the soil [18]. These accidents have had costly and detrimental effects on the environment, economy, and occupants' livelihood within the casualties' locations.

These numerous damages caused by corrosion can also occur to structures, ships, and other equipment used in seawater service, as seen when the oil tanker Erika broke in two and sank in the Bay of Biscay, destructive corrosion of the internal structure of the vessel [19]. Furthermore, the collapse of the Point Pleasant bridge that connects West Virginia and Ohio, known as the Silver Bridge, was due to stress corrosion within the bridge construction and bridge chain compound [20]. From these examples, the damage to these structures can be attributed to seawater contact, leading to corrosion. Corrosion by seawater is an electrochemical process during which all metals and alloys in contact with seawater develop a specific electrical potential (or corrosion potential) at a particular seawater pH [21]. Because seawater is rich in natural electrolytes and highly corrosive, metals and alloys corrode inevitably in this environment.

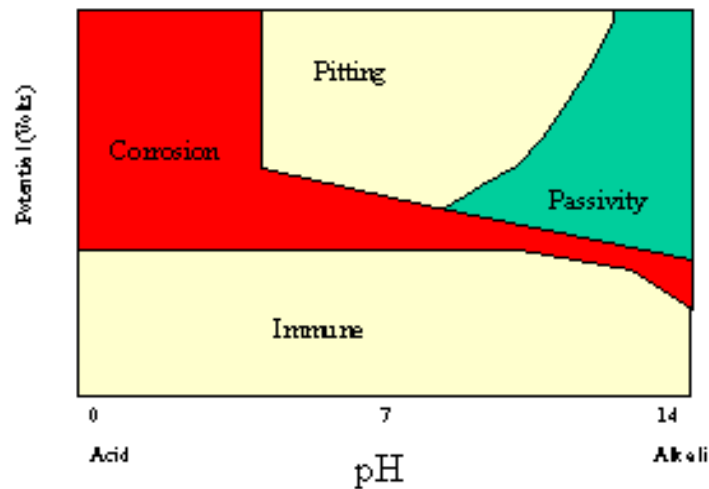


Figure 1 Regions of change on a metal sample [21] (permission for use has been requested from the publisher)

Using figure 1 above, the metamorphosis of a metal sample in a seawater environment is illustrated, showing the regions where the metal will freely corrode, to the region of passivation where stable oxide or other films form, and the corrosion process is stifled; the region of pitting corrosion where the corrosion potential of the metal exceeds that of its oxide; and the region of immunity where the metal sample is safe for use [21].

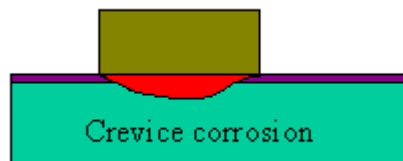
Corrosion resistance in metals is classified according to the source of resistance. Some metals can resist corrosion by themselves without a source of protection, e.g., such as an oxide film. Aluminium and galvanized steel are examples of corrosion-resistant metals in use today. Another class of corrosion resistive metals is those metals that require a source of protection for them to resist corrosion. This source of protection is the presence of an oxide film. If possessing self-healing and stable properties gives the metal a high corrosion resistant property, e.g., the oxide films coated on stainless steel titanium [22]. Regardless of these attractive properties, oxides with stable properties are still susceptible to degradation due to attacks from aggressive hydrochloric acid concentrations present and developed in chloride environments. These large chloride concentrations are present in seawater, making it the most effective corrosion-enhancing electrolyte amongst liquids [23].

Corrosion within seawater differs significantly from corrosion on land due to the influence that seawater characteristics play in metal corrosion law. In other words, there exist characteristics of corrosion in seawater.

Firstly, galvanic corrosion is induced when two dissimilar metals in contact within seawater. In this seawater corrosion characteristic, the two dissimilar metals must remain in electrical contact under a medium of high conductivity and low corrosion resistance (i.e., seawater). During this electrical contact, one of the metals assumes the position of the anode and corrodes faster than it usually would as a standalone metal. In contrast, the other metal in the contact assumes the position of the cathode and corrodes slower than it usually would as a standalone metal [21].

Furthermore, a vast amount of chlorine in seawater makes passive metals prone to various localized corrosion forms in seawater, ranging from pitting corrosion to crevice corrosion to stress corrosion. These localized corrosion forms in seawater vary from one

sea zone (sea zones: atmospheric zone, splash zone, tidal zone, immersion zone, and sea area) to another but are predominant in the splash zone [24]. An ever-constant presence of oxygen in the seawater atmosphere around the splash zone leads to an increase in salt attack's aggressiveness from seawater. The differential concentration of oxygen dissolved at the splash zone creates a cell in which attack is dominant where oxygen concentration is lowest, leading to structural deformities, e.g., crevices, in the metal sample, the hidden start points of seawater corrosion [25]. As seen in figure 2, these crevices are anodic and acidic because they permit more water and chloride from seawater while excluding oxygen.



**Figure 2 Corrosion starting from a developed crevice on a metal surface [21]
(permission for use has been requested from the publisher)**

Consequently, the non-uniformity in metals' physical properties and chemical properties leads to varying potential on different parts of the metal surface, which influences the corrosion rate. Parameters such as dissolved oxygen, salinity, pH, oxidation-reduction potential, temperature, conductivity, etc., are significant factors that influence and amplify the corrosion rate (and in turn the corrosion) of metals in seawater. Some of these parameters are used in the generation of the dataset employed in developing the super learner model.

In the case of carbon steel under water, stress corrosion cracking and the significant factors influencing corrosion rate are the characteristics that both contribute to the gradual degradation of carbon steel under seawater. This is because stress corrosion cracking is induced from the combined influence of the corrosive environment (developed through the amplification of the corrosion rate by the parameters) and tensile stress. This tensile stress could be in the form of residual stress (due to grinding, welding, machining, etc.) or directly applied stress on the carbon steel [25].

A study conducted by the National Association of Corrosion Engineers (NACE) showed that the effective implementation of corrosion protection methods could save between 15-35% of the cost of damage to corrosion-prone structures worldwide [26]. Accurate determination of the corrosion rate of structures (erected on the soil surface) and subsequent protection is being done using an advanced cathodic protection system known as the Adaptive Corrosion Protection System (ACPS). This device adaptively protects corroding metal using its corrosion potential and corrosion current. This ACPS cannot be applicable in providing corrosion protection for 3C steel metal structures erected within seawater due to the dissimilarity in parameters causing corrosion in soil and seawater. Furthermore, this device is characterized by a slow polarization curve whose accuracy in determining the corrosion potential and current is negatively influenced by the wear and tear of the ACPSs' hardware. Therefore, for accurate determination of the corrosion rate of 3C steel metal in seawater (which can be further employed in providing corrosion protection to the metal sample), a machine learning model that accurately estimates the corrosion rate of the metal sample should be implemented.

2.3. Metamorphosis of Corrosion Rate Estimation

The corrosion rate is the speed at which a sample metal fails in a specific environment. The rate is hugely dependent upon the environmental conditions and the type & condition of the metal. The methods employed in determining the corrosion rate of metals have evolved over the years, from the conventional methods for determining corrosion rates (gravimetric method and the electrochemical method) to the use of machine learning to predict the corrosion rates of metal samples.

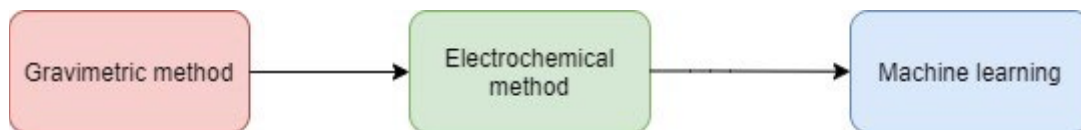


Figure 3 Evolvement of corrosion rate determination

2.3.1. Conventional methods

The gravimetric method, also known as metal loss measurement, is the oldest technique widely used in the industry. This technique represents global corrosion rates in the system. It works by calculating the weight loss of the sample undergoing corrosion (in the assumption that corrosion is uniform) over long periods of time. This method is time-consuming (due to the time it takes for the effects of corrosion to be pronounced) but its accuracy depends on precise weight measurements of the metal sample under corrosion [27].

The electrochemical method was developed to address the limitations faced by the gravimetric method. It involves using an electrochemical signal (developed when a metal sample undergoes corrosion in an aqueous system) as a primary source of information to relate the rate of corrosion to the potential, current and electric charge of the metal sample undergoing corrosion. This technique is subdivided into three techniques: linear polarization, Tafel extrapolation, and electrochemical impedance spectroscopy.

The Linear Polarization method is a technique that uses known Tafel parameters to convert the polarization resistance (resistance of the metal to oxidation during the application of external potential) into the corrosion rate, which is inaccurate due to errors in the Tafel parameters from polarization measurements [28].

Tafel extrapolation is a mathematical technique utilized to estimate corrosion current and corrosion potential for corrosion rate. The Tafel extrapolation method provides a simple and straightforward method to determine corrosion rate using the Tafel parameters. Its primary disadvantage is that this method requires the use of a wide range of voltage to polarize the material undergoing corrosion to extract the corrosion rate of the material, so the measurement is not only time consuming but also alters the surface conditions of the material (e.g., permanent change or surface damage), [29].

The electrochemical impedance spectroscopy method is more advanced than the linear polarization method due to its ability to study highly resistive structures (such as coatings and linings) and corrosion in a low conductive solution [30]. The major disadvantage of this method is its requirement of previous knowledge of Tafel parameters [31]. Although this requirement can be closely modeled to machine learning algorithms,

the Tafel parameters do not consider the empirical observation of major environmental corrosion influencers.

Many researchers have attempted to use all these estimation techniques to predict corrosion. One study identified the effects of applying gravimetric and electrochemical methods to pure magnesium and Z31 magnesium/aluminum alloys [32]. Results from this study showed inconsistencies between the corrosion rate values evaluated by both methods. The gravimetric method was time-consuming and inaccurate due to the dependence of its accuracy on correct weight measurements. In contrast, the electrochemical technique provided information on just the instantaneous corrosion rate (or corrosion resistance).

In another study, an experiment to demonstrate the electrochemical method's weak estimation power was carried out. They used an extensive series of studies reporting proof of direct chemical reaction of H₂O molecules with the metallic surface without the interference of electron transfer reactions [33]. Subsequent studies provided further details on the unusual chemical dissolution of metals [34,35]. The researchers proposed a numerical simulation to predict the corrosion rate of steel in concrete based on equivalent electrical circuit models. This method allowed for a quantitative prediction of microcell action in reinforced concrete. However, this model results in misleading corrosion rate prediction in several cases. It is based solely on the driving voltage, the concrete electrolyte's resistivity, and the passive steel's polarization behavior without considering direct environmental factors [36]. The inaccuracies in the corrosion rates generated by these techniques have made the need for a system to deliver accurate corrosion rate estimates imperative.

2.3.2. Utilization of Machine Learning in Corrosion Rate Prediction

With recent advancements in machine learning, several machine learning techniques have been implemented to develop models capable of predicting other parameters, e.g., soil resistivity, corrosion current, temperature, etc., contributing significantly to corrosion. There are only very few models developed to predict corrosion rate. As compared to protection hardware, these machine learning models are not only non-destructive (as they do not require a constant supply of large current) but also consider the empirical observations of environmental factors that directly influence

corrosion rate. However, these models have been plagued with poor generalization, loads of error, and enormous training and prediction time, causing memory issues.

Wen, et., al. [10] developed a model using a support vector regression approach combined with particle swarm optimization for its parameter optimization, forming a support vector regression with leave-out cross-validation (SVR-LOOCV). The model was developed to predict the corrosion rate of 3C steel under five different seawater environment parameters (temperature, dissolved oxygen, salinity, pH value, and oxidation-reduction potential). The prediction results highlighted its limited generalization ability and the volume of errors in its predictions after using this model. The results also highlighted the high bias and low variance characteristics of the model as the model failed to capture the complexity of the dataset. Poor implementation of metaheuristics added to the model introduced a difficulty in parameter tuning due to the lack of resources needed to identify the optimal settings for the support vector regression algorithm's parameters. This, therefore, leads to inefficient utilization of memory at increased computation time consumption.

The behavior of 3C steel was also studied in relation to its corrosion rate under the five different seawater environment parameters (temperature, dissolved oxygen, salinity, pH value, and oxidation-reduction potential) [11]. The researchers developed a metaheuristic known as the smart firefly algorithm. A metaheuristic is a high-level procedure designed to find, generate or select sufficiently good solution to an optimization problem. This was implemented with the least squares support vector regression model to form the SFA-LSSVR. After training and testing, the model outperformed the other models (i.e., voting, bagging, and tiering models) developed by the researchers, but was significantly flawed by its overfitting feature. This implies that the model learned the training data properly but performed poorly on test data. Aside from overfitting, poor memory utilization and excess computation time are experienced just like the previous model due to the model's inferior implementation of metaheuristics.

Although the prediction of the corrosion rate of 3C steel in seawater is a regression problem, a useful classification-based model effectively and accurately predicted corrosion rate in severe, moderate, and minor corrosion rate was developed. The work analyzed supervised learning algorithms' suitability to estimate corrosion rate causing corrosion in oil pipelines [37]. Compared to other works, this model is highly flawed due to

the lack of structure required to serve as a foundational algorithm towards corrosion protection. Aside from its output's user-friendly nature, its text-based output cannot give the certainty of corrosion rate as a numerical output would deliver.

In this thesis, a variant of the super learner model is developed, which addresses the issues encountered by the models mentioned in the prior paragraphs, learns the training data, and delivers prediction estimates at a swift computation time with fewer memory requirements.

2.4. Super Learner and its Constituents

Machine learning is a branch of artificial intelligence (AI) that offers systems the ability to learn and improve from experience without being explicitly programmed automatically. It deals with the design and development of algorithms that allow computers to develop behaviors based on empirical data [38]. Machine learning concentrates on the design of programs that can read and learn data.

The learning process begins with observations or data, such as examples, direct experience, or instruction (known as training data) to look for patterns in data and estimate the unknown relationship between input and target parameters using known examples. The relationship is then used to estimate outputs for new inputs. Machine learning is split into supervised, unsupervised, and reinforced learning [39]. Supervised learning is employed in solving classification and regression problems. A major difference between these two types of problems is the nature of their outputs. Classification problems have nominal outputs, while regression problems have numerical outputs [40]. Supervised learning aims to develop a model from the training data that can be used to predict future responses [41]. There are several forms of supervised learning-based machine learning algorithms. They include regression analysis, ensemble methods, decision trees, support vector machines, neural networks, etc.

Early machine learning enthusiasts were faced with merging single prediction from individual models into one powerful model with superior prediction accuracy than the individual models. In 1992, a prediction model named stacking was developed [42]. This model constituted several low-level prediction algorithms combined to form a high-level

model with a prediction accuracy far better than the individual low-level algorithms' prediction accuracy. This model's prediction strength was tested by implementing the model in a regression problem that generated higher prediction accuracy results. Additionally, the model's architecture was exploited to improve prediction accuracy by placing certain boundaries on the higher-level model [43].

The term "Super Learner" was coined on the foundation of an improved model featuring not just a large library of diverse algorithms to enhance performance while creating the best-weighted combination of candidate algorithms to improve performance further, but also a model performing far better than the best individual algorithm [44,45]. The super learner is a general loss-based learning method designed to find the optimal combination of a collection of prediction algorithms. The super learner algorithm combines the algorithms (as base learners) by minimizing the cross-validated risk. This implies that the model takes care of the problem of overfitting by implementing k-fold cross-validation. The super learner framework is built on the theory of cross-validation and allows for a general class of prediction algorithms to be considered for the ensemble [46].

The super-learner model is a variation of stacking or k-fold cross-validation where individual models would be trained on k-fold data split. A final meta-model would then be trained on their output, also called an out-of-fold prediction from each model. Overall, the super learner is an important tool implemented in a very limited number of studies to reduce parametric assumptions, boost predictive accuracy, and avoid overfitting [47].

In this thesis, a variant of the super learner model was developed strictly to predict 3C steel metal's corrosion rate in seawater using five algorithms chosen by the developed selection framework. All five algorithms are trained, tested individually on the same dataset, and compared with the super learner model; they are combined to create five other algorithms trained and tested on the same dataset. These five selected algorithms performed the best individually on the same dataset to be put forward by the selection framework for implementation in developing the super learner model. The algorithms chosen by the developed selection framework are:

- i. Linear regression
- ii. Ridge regression
- iii. Lasso regression

- iv. Support vector regression using the polynomial kernel
- v. Extra Gradient Boosting

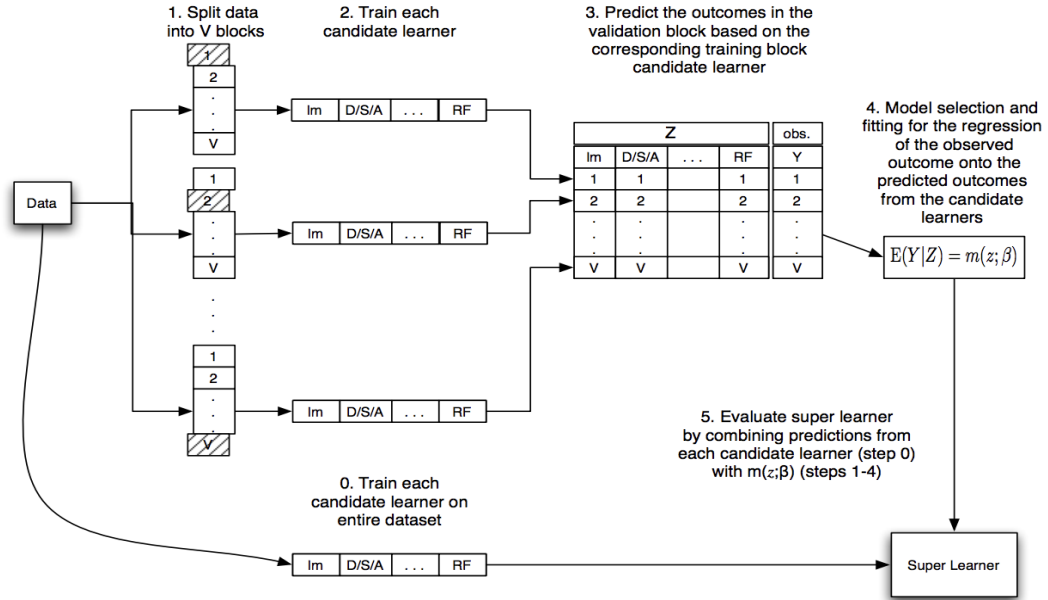


Figure 4 Development process of the super learner model [47] (permission for use has been requested from the publisher)

2.4.1. Linear Regression

This is the simplest and basic form of regression used for predictive analysis. In linear regression, the independent and dependent variables are continuous and linear [48]. Linear regression can be classified into simple linear regression and multiple linear regression [49]. Simple linear regression analysis assumes a linear relationship between the independent variable (x) and the dependent variable (y). Equation (3) is the fitting line's mathematical representation for a simple linear regression [50,51].

$$y = ax + bx + \varepsilon \tag{3}$$

Where Y is the dependent variable, whose value depends proportionally to x a and b are the unknown coefficients, and ε is the error term.

A study highlighted how multiple linear regression analysis assumes a linear relationship between multiple independent variables (x_1, x_2, \dots, x_n) and dependent variables (y), which calculates the effects of each independent variable (β) [52,53] as shown in equation (4).

$$y = \beta_o + \beta_1x_1 + \beta_2x_2 \dots \dots \beta_nx_n + \varepsilon \quad (4)$$

where, n = number of observations, Y = dependent variable, x_{1-n} = independent variables, β_o = constant term, and ε = the model's error term.

Although the model is simple, in some cases, it is shown to produce quite good results, as well as very fast predictions due to its simple form [54]. Also, the input variables in multiple linear regression are susceptible to noise data or contain unnecessary information, reducing the predictive power of regression analysis [55]. In this study, the `LinearRegression` module is used to fit the linear model with coefficients to minimize the residual sum of squares between the observed targets in the dataset and the targets predicted by the linear approximation. This thus forms a function called upon by the super learner model for training and testing.

2.4.2. Ridge Regression

Ridge Regression is a machine learning algorithm for analyzing multiple regression data that suffer from overfitting and multicollinearity (a situation in which two or more dependent variables in a multiple regression model are highly linearly related) [56]. Ridge regression is a member of the family of penalized regression approaches because it shrinks the regression coefficients towards zero by imposing a penalty (using a shrinkage parameter). Multicollinearity amongst the training data leads to unbiased estimates with very large variances meaning estimates are very far from the true value [57]. Ridge regression gives more reliable estimates by adding the shrinkage parameter or penalty term to the regression estimates in equations (5) & (6).

$$RSS(\beta) = \sum_{i=1}^n (y_i - (\beta_o + \beta_1x_{i1} \dots + \beta_px_{ip}))^2 \quad (5)$$

$$s(\beta, \lambda) = RSS(\beta) + \lambda \sum_{j=1}^p \beta_j^2 \quad (6)$$

Where,

$\lambda \geq 0$: shrinkage parameter which controls the amount of shrinkage applied to the estimates

$RSS(\beta)$ = sum of squared residuals

$\lambda \sum_{j=1}^p \beta_j^2$: ridge penalty

Tikhonov regularization tries to find estimates that fit the data reasonably well by making the $RSS(\beta)$ small, while the ridge penalty is also small when the estimates $(\beta_1 \dots \dots \beta_p)$ are shrunken approximately to zero. In terms of controlling the amount of shrinkage parameter, λ , when $\lambda = 0$, ridge regression becomes least squares estimation, but as λ gets larger, the ridge coefficient estimates, $\|\beta^R(\lambda)\|_2^2$ shrinks towards zero. Nevertheless, the ridge penalty in (2.5) will shrink all the coefficient estimates towards zero, but it will not set any of them exactly zero. Hence, ridge regression has the disadvantage of including all the predictors in the final model [58].

One study presented ridge regression as a means of estimating regression coefficients with smaller mean-square error than their least-squares counterparts when predictors are correlated [59]. In another study on the penalized ridge regression approaches, ridge regression stood apart as the approach that offered the best predictive performance [60]. In a comparative study, the penalized and unpenalized regression methods for predicting complex disease were compared, resulting in ridge regression outperforming other prediction methods in diverse disease phenotypes [61]. In this study, the `Ridge` module is used to solve the regression model. The loss function is the linear least-squares function. Regularization is given by the l2-norm fit, forming a function with adjusted hyperparameters be freely called upon by the super learner model for training and testing.

2.4.3. Lasso Regression

Lasso Regression, also known as the least absolute shrinkage and selection operator, is a form of regression that deals with many predictor variables. Unlike ridge regression, this technique shrinks some coefficients while setting others to 0. The lasso estimates are defined as follows [62].

$$L(\beta, \lambda) = RSS(\beta) + \lambda \sum_{j=1}^p |\beta_j| \quad (7)$$

Where,

$\lambda \geq 0$: tuning parameter, which controls the amount of shrinkage applied to the estimates.

$RSS(\beta)$: sum of squared residuals

$\lambda \sum_{j=1}^p |\beta_j|$: lasso penalty

The lasso estimator, $\beta^L(\lambda)$ minimizes (2.6) over β for a given λ . The tuning parameter λ decides whether $\beta^L(\lambda)$ is sparse or not (setting some coefficient estimates exactly to zero). When λ gets larger, $\|\beta^L(\lambda)\|$ gets smaller, which results in a sparse solution [59].

Lasso regression compensates for ridge regression's inability to reduce the number of predictors in the final model [63]. This compensation is carried out by using the variable selection and shrinkage feature of the lasso penalty, which forces some of the coefficient estimates to be exactly equal to zero when λ is satisfactorily large [64]. In this thesis, the `lasso()` module is used to develop a function that forms part of the algorithms used to develop the super learner model. Its hyperparameters are changed to tune the model towards perfect prediction accuracy. The function is then freely called by the super learner model for training and testing.

2.4.4. Support Vector Regression

Support vector machine is a machine learning algorithm that implements a structural risk minimization principle to obtain a good generalization level. The foundational support vector machine algorithm solves the bipartition problem at the AT & T Laboratories [65,66]. A study conducted showed the application of kernel tricks facilitated the development of non-linear classifiers to maximize the margin of the hyperplanes [64]. Support vector machines are used in recognizing delicate patterns in complex datasets. SVM has two forms: support vector regression (SVR) and support vector classification (SVC). Support Vector Regression (SVR) is the most common application form of SVMs. Support Vector Regression is the form of support vector machine that utilizes a subset of training data to reduce the generalization error bound to achieve a generalized performance [68]. Support vector machines project the data into a higher-dimensional space and maximize the margins between classes or minimize the

error margin for regression [69]. A study successfully explained the mathematical derivation of the Support Vector Regression (SVR) model using the training data given as [70]:

$$[(x_1, y_1) \dots (x_n, y_n)] \in X \times \mathbb{R} \quad (8)$$

Where,

X = space of the input pattern

In support vector regression, the goal is to estimate the function $f(x)$ with the most ϵ deviation from the obtained targets y_n for all the training data and at the same time as flat as possible. The case of linear function f has been described in the form as

$$f(x) = (w, x) + b \quad \text{with } \omega \in \mathbb{R}, b \in \mathbb{R} \quad (9)$$

(..) represents the dot product in \mathbb{R}

ω = flatness parameter

To achieve the flatness parameter, the Euclidean norm, i.e., $\|\omega\|^2$. This can be written as a convex optimization problem.

$$\begin{aligned} & \text{Minimize } \frac{1}{2} \|\omega\|^2 \\ & \text{Subject to } \begin{cases} y_i - (w, x_i) - b \leq \epsilon \\ (w, x_i) + b - y_i \leq \epsilon \end{cases} \end{aligned} \quad (10)$$

A Lagrange function, which expresses w as seen in equation (10), is derived from the objective function as seen in equation (9)

$$w = \sum_{i=a}^l (\alpha_i - \alpha_i^*) (x_i, x) x_i \quad (11)$$

With the aid of the determined w expression, the value of the function f can be derived using the expression in equation (11). This is the standard support vector regression algorithm to solve approximation problems.

$$f(x) = \sum_{i=a}^l (\alpha_i - \alpha_i^*) (x_i, x) + b \quad (12)$$

Kernels are functions that take low-dimensional input space and transform it into a higher-dimensional space. In other words, they convert a non-separable problem into a separable problem [59]. Kernel functions are used to take data as input and transform it into the required processing data. The three major types of kernels are the linear, polynomial, and radial basis function [71,72,73].

In this study, the polynomial kernel was implemented to develop the super learner model ahead of the linear and radial basis function kernel due to its “fit tuning” feature due to the lack of limit to the degree that can be applied to the kernel. This kernel represents the similarity of vectors in the training set of data in a feature space over polynomials of the original variables used in the kernel [74].

$$k(x_i, x_j) = (1 + x_i \cdot x_j)^d \quad (13)$$

In this thesis, the `SVR(kernel = 'poly')` module is used to develop a function that induces the polynomial kernel. A linear model whose accuracy can be tuned by changing the degree parameter's integer values is created. The super learner model freely calls this function for training and testing.

2.4.5. Extra Gradient Boosting Machine (XGBM)

Gradient Boosting methods works like Adaboost by sequentially adding weak learners to an ensemble, each one correcting its predecessor. Unlike Adaboost, Gradient Boosting method fits new learners to the residual errors made by the previous learners instead of reallocating weights for every instance not properly learned at every iteration [75]. Two forms of gradient boosting are XGBM and LightGBM [76]. XGBM is one of the fastest implementations of gradient boosted trees [77]. XGBM tackles the major challenge posed

by gradient boosting trees (consideration of the loss function) by reducing the search space of possible feature splits after looking at the spread of the dataset's input features [78]. The regularization feature of XGBM allows for its hyperparameters to be tuned efficiently. Also, its implementation of out-of-core computing for large datasets sums up its presence in this work [79]. In this thesis, the `xgb.XGBRegressor` module was implemented to fit the regressor module of XGBM on the original dataset. Further tuning of its hyperparameters drives the accuracy of the algorithm's prediction accuracy. The super learner model subsequently calls the function for training and testing.

Before these modules were implemented in calling the algorithms put forward by the selection framework, there exists a core in the framework that provided it with a unique means by which it not only chose the right algorithms for the development of the super learner model but also provided these choice algorithms with their hyperparameters pre-optimized. This core is known as the Random Search Hyperparameter Optimization.

2.5. Random Search Hyperparameter Optimization

Random search hyperparameter optimization is the backbone of the selection framework developed and implemented in designing the super learner model used in this thesis. To understand random search hyperparameter optimization, the difference between model parameters and model hyperparameters must be highlighted. Model Parameters are the parameters estimated by the model from the given data, e.g., weights of a deep neural network [80]. On the other hand, Model Hyperparameters are the parameters that the given data model cannot estimate. They are parameters whose values are set before the learning process begins. They are the parameters used to estimate the model parameters, e.g., the ridge regressor's alpha value [81,82].

The choice of hyperparameters determines the level of performance of the model. Hence the need for a means of determining these hyperparameters. This means knowing as Hyperparameter Tuning [83]. It is the process of determining the right combination of hyperparameters that facilitates the maximization of model performance. With the gruesome number of hyperparameters per model, automated hyperparameter tuning is employed in determining the optimal hyperparameters using an algorithm that automates and optimizes the process [84,85]. Examples of automated hyperparameter tuners are Grid Search, Random Search, Bayesian Optimization, Tree-Structured Parzen

Estimators, etc. [86]. In this thesis, the random search hyperparameter tuner was implemented.

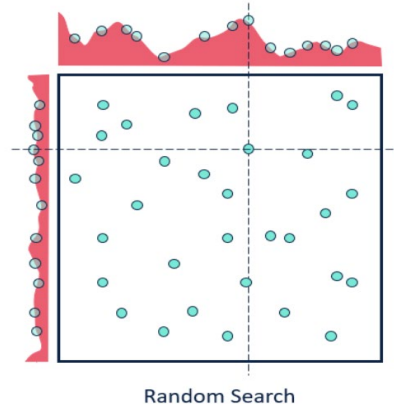


Figure 5 Layout implemented by random search (permission for use has been requested from the publisher)

Random search is a basic improvement on grid search. It is based on a random search of hyperparameter values from user-defined distributions [87]. It involves the user providing a statistical distribution of the hyperparameter values for the optimization method or algorithm to test. These distributions are created for each hyperparameter, after which the optimization algorithm or method tests them at every iteration by generating a model. The iteration continues until the predefined (by user) distributions for each hyperparameter are exhausted or until the desired accuracy is reached [88]. Random search was proven to be better than grid search because, [89]: firstly, in random search, a range can be assigned independently according to the distribution of search space, therefore making random search perform better in cases where some hyperparameters are not uniformly distributed as seen in figure 5.

On the other hand, in grid search, the budget for each hyper-parameter set is a fixed Value $\frac{1}{B^N}$, Where B is the total budget and N is the number of hyper-parameters. Secondly, in the context of handling large datasets, the random search offers a less time-consuming technique (same as the grid search), leading to an outcome with a large probability of finding the best hyperparameters, unlike grid search, whose longer search time cannot guarantee better results [91,90].

2.6. Computation time and Memory Consumption Optimization

Computation time and memory consumption have been a genuine issue over the years in transferring machine learning models onto portable physical systems. Most machine learning models' size mitigates their utilization in basic physical systems, therefore requiring complex-expensive systems with greater hardware requirements to carry these models. In comparison with the model developed to predict the same dataset's corrosion rate [11], the super learner's architecture facilitates efficient utilization of memory at less computation time by its models using multilayered ensembles (ML-Ensembles).

Computation time and memory consumption issues are common, mostly with a moderately sized dataset, becoming more prominent with an increase in the training dataset's size. When ensembles such as bagging, voting, etc., are executed, serialization of the training dataset occurs (i.e., training ensembles in parallel), which are then stored subprocess (memory). As the number of ensembles in parallel increases, multiple copies of the training dataset are stored in the same subprocess, leading to increased memory consumption at slower computation time when the model is called for execution [92].

Multilayered Ensembles (ML-Ensemble) is a python library that utilizes memmapping to builds models in the form of a feed-forward network. Layers are stacked sequentially, with each layer taking the previous layer's output as input. With the aid of multilayered ensembles, ensembles of any shape and form can be built, features can be propagated through layers, estimation method can be varied between layers, preprocessing can be differentiated between a subset of base learners but most importantly, the computation time and memory consumption are optimized [93]. Also, the serialization of the training data, sending the serialized data to the subprocess, and copying of the dataset (leading to an increase in the number of subprocesses) are all avoided, leading to the memory consumption being constant [94,95].

Ultimately, the knowledge on the architecture of the super learner provided in this chapter is essential in developing a super learner model that not only provides a comparable prediction accuracy (compared to other model trained on the same dataset) but also efficiently utilizes system memory (at lesser computation time) while predicting the corrosion rate of 3C steel in different environmental conditions.

Chapter 3.

Super Learner Model Implementation

Due to the lack of super learner modules in the press and play tools, e.g., WEKA, MATLAB machine learning application, etc., the python programming language was utilized in developing the architecture that operates and runs the desired super-learning model with the target features of less memory consumption (implying lesser computation time) and comparable prediction accuracy required for this thesis. This chapter covers the methodology employed in the development of the super learner model. This covers the tools implemented in the selection and development of base learners and the super learner model, respectively, to the super learner model's performance evaluation. A justification for the choice of tools, dataset employed, algorithm selection technique, model hyperparameter tweaks, and the developed model's performance evaluation is provided.

3.1. Tools Utilized

Several specific software packages were employed to implement the base learners and develop the super learner model. These packages include Spyder integrated development environment (IDE), Numpy, Pandas, Matplotlib, Scikit-learn, and multilayer ensemble libraries.

Spyder is a free and open-source scientific environment written in python programming language. Spyder IDE is used to develop data science, machine learning, predictive analytics applications, etc. Spyder distribution package was implemented for this study because of its suitability with the Windows OS (operating system of the PC), facilitating the importation of the libraries used in the development of the super learner model [96]. In addition, features such as the iPython console (a terminal used to display print commands output), variable explorer (stores all variables, lists, arrays, etc.), and the plots tab (visualization of plots) available to Spyder cannot be found in any other python integrated development environments such as Jupyter notebook, Google Cloud, etc.

NumPy stands for Numerical Python and is one of the most effective mathematical and scientific libraries in python programming. This library was implemented due to its support for large, multidimensional arrays objects and various tools to work with these

arrays [97]. Numpy provides functions spanning from high performance, efficient storage to other data operations, all to transform arrays. Furthermore, NumPy came pre-installed in Spyder (unlike other IDEs requesting special and tedious installation methods) but was still imported at the beginning of the code block. In this study, Numpy was used to develop arrays implemented in each model covered by this study.

Pandas is an open-source, fast, flexible data manipulation and analysis tool built off python programming language. Pandas was implemented in this work due to its ease with the analysis and importation of the dataset. Pandas is built on the Numpy package with its key data structure call the data-frame. This structure facilitates the manipulation and storage of tabular data in rows and columns of observation and variables, respectively [98]. The Pandas package was imported in the Spyder IDE and was used to import the file containing the dataset used in this thesis. The statistical explanation of the dataset, dataset head, and tail was developed using the describe method, head method, and tail method, respectively, made available by the Pandas package.

Matplotlib is a plotting library for both Numpy and the python programming language. It is a library used in the development of static and animated visualizations [99]. The major modules offered by matplotlib include plot, pyplot, and pylab. Matplotlib was implemented in this work due to the availability of more plotting functions compared to Seaborn.

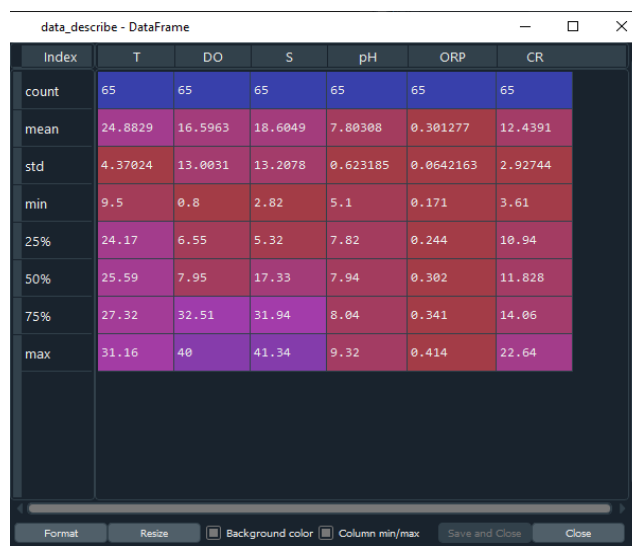
Also known as Sklearn, scikit-learn is a free, open-source machine learning library for python programming. It is home to various supervised and unsupervised machine learning algorithms. Also, it supports libraries (such as Numpy and SciPy), cross-validation models, feature extraction, feature selection, parameter tuning, manifold learning, etc. [100], which no other module in the world of python provides, hence its utilization in this work. Its ease of importation into the Spyder IDE facilitated the importation of performance evaluation metrics, model selection method, algorithm functions, and splitting method, all required for the algorithm selection technique and super-learner model development process.

3.2. Dataset employed

The dataset employed in this work was generated from a marine corrosion rate of 3C steel experiment conducted by [10] in China. Using the electrochemical technique, 46 sample points for the corrosion rate of 3C steel were measured under five different seawater conditions. The dataset is detailed in Appendix A. Figure 6 shows the statistical description of the dataset.

The raw dataset was pre-processed by removing null values and implementing data augmentation. Data augmentation was implemented using an autoencoder (on the training data) to develop 19 more data points. Therefore, our dataset was made up of 70% empirical data and 30% artificial data.

For post processing, the dataset was randomly split into training data set and a test data set containing 90% and 10% of the dataset, respectively, adhering to the standard machine learning paradigm that the amount of learning data should be much larger than the amount of test data. The objective is to avoid overfitting and underfitting problems [96]. The training dataset is the portion of the dataset in which the super learner model learned to facilitate its prediction of the target feature. The training dataset was used to make sure that the model recognizes the patterns in the dataset. While the test dataset was used to provide an unbiased evaluation of the final model fit on the training dataset. It was used to test the prediction accuracy of the developed super learner model.



The screenshot shows a window titled "data_describe - DataFrame" with a table of statistical data. The table has 7 columns: Index, T, DO, S, pH, ORP, and CR. The rows represent various statistical measures: count, mean, std, min, 25%, 50%, 75%, and max. The values are as follows:

| Index | T | DO | S | pH | ORP | CR |
|-------|---------|---------|---------|----------|-----------|---------|
| count | 65 | 65 | 65 | 65 | 65 | 65 |
| mean | 24.8829 | 16.5963 | 18.6049 | 7.80308 | 0.301277 | 12.4391 |
| std | 4.37024 | 13.0031 | 13.2078 | 0.623185 | 0.0642163 | 2.92744 |
| min | 9.5 | 0.8 | 2.82 | 5.1 | 0.171 | 3.61 |
| 25% | 24.17 | 6.55 | 5.32 | 7.82 | 0.244 | 10.94 |
| 50% | 25.59 | 7.95 | 17.33 | 7.94 | 0.302 | 11.828 |
| 75% | 27.32 | 32.51 | 31.94 | 8.04 | 0.341 | 14.06 |
| max | 31.16 | 40 | 41.34 | 9.32 | 0.414 | 22.64 |

Figure 6 Statistical description of the dataset on the Spyder IDE

The dataset is made up of five features, namely, temperature (T), dissolved oxygen (DO), salinity (Sal), pH value (pH), oxidation-reduction potential (ORP), and corrosion rate.

The temperature of seawater was measured using temperature probes at every sample point. The unit for temperature measurement is degrees Celsius ($^{\circ}\text{C}$).

Dissolved oxygen is the amount of oxygen content present in seawater. The higher the dissolved oxygen content in seawater, the higher the metal's electrode potential in the sea, hence the faster the corrosion rate of the metal. The unit for dissolved oxygen is mg/L.

Salinity is a term used to describe the degree of dissolved salt in water. The salt content in water directly affects the conductivity and oxygen content of water. With the increase of salt content in water, water's electrical conductivity increases, but the oxygen content decreases. The salinometer was used to determine the degree of salinity of each seawater sample for all 46 epochs. The unit of salinity measurement for seawater is parts per thousand.

pH value is the degree of acidity or alkalinity of a medium. The pH of the seawater is conducive to the inhibition of seawater corrosion of steel. However, the pH of the seawater is far from the effect of oxygen content on corrosion. Although the surface seawater pH is higher than that of the deep seawater, the corrosion of the surface seawater is far higher due to the seawater's photosynthesis in the surface stronger than deep seawater, which is consistent with the actual experimental conclusion.

Oxidation-reduction potential (ORP), also known as *Redox*, is a measurement that reflects the ability of a molecule to oxidize or reduce another molecule. The unit of oxidation-reduction potential measurement is mV.

Corrosion rate is the rate at which a corrosion-prone material or metal deteriorates in a specific environment. The rate, or speed, depends on environmental conditions and the type and condition of the metal. The unit of corrosion rate employed in the dataset is μAcm^{-2}

3.3. Performance Evaluation Metrics

In this thesis, performance evaluation metrics were employed not just in the base learners selection technique but also in developing the super learner model. Evaluating the model's performance with more than one method is critical as it provides an avenue to analyze the performance consistency of the model.

Asides from being the same metrics employed in the evaluation of the models in [11], these performance evaluation metrics were also selected because of their reliability in evaluating the performance of regression models. The metrics are Root Mean Square Error (RMSE), R Squared (R2), and Mean Squared Error (MSE). These metrics (except the symmetric mean absolute percentage error) were applied to the test dataset and predicted dataset to generate values that indicated the models' performance.

Root Mean Squared Error (RMSE) takes the square root of the average of the square of the difference between the actual values and the predicted values. With RMSE, the gradient of the model is easy to compute. Just like the mean absolute error, the numeric value of the root mean square error was used to determine the accuracy of the model it evaluates. When the super learner model earns a root mean square error score tending towards zero at every iteration (a score of 0 means the model is perfect), it implies a continuous increase in the model's prediction accuracy. This metric is used in this thesis to provide a level-based metric system for comparing accuracy with scores in [101]. The root mean square error is mathematically represented as:

$$\text{RMSE} = \sqrt{\frac{\sum_{j=1}^{M_{\text{test}}} |y_{\text{true}}^j - y_{\text{predicted}}^j|^2}{M_{\text{test}}}} \quad (14)$$

R-square is used to measure the degree of variability in the dependent variable highlighted by the model. It is referred to as R Square because it is the square of the correlation coefficient(R). As in the previous two metrics, the numeric value of the R Square was used to determine the accuracy of the model it evaluates. When the super learner model earns an R-Square score tending towards 1 (a score of 1 means the model is perfect) at every iteration, it implies a continuous increase in the model's prediction

accuracy. This metric is used in this thesis to provide a level-based metric system for comparing accuracy with scores in [102]. The R-Square is mathematically represented as:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_{\text{predicted}} - y_{\text{true}})^2}{\sum_{i=1}^n (y_{\text{true}} - \bar{y}_{\text{mean}})^2} \quad (15)$$

Mean Square Error (MSE) measures the squared average distance between the true values and the predicted values. The mean square error measures the average of the squares of the errors which is, the average squared difference between the predicted values and the true values. MSE is a risk function, corresponding to the expected value of the squared error loss. The numeric value of the Mean Square Error (MSE) is also used to determine the accuracy of the model it evaluates. When the super learner model earns a mean square error score tending towards 0 (a score of 0 means the model is perfect) at every iteration, it implies a continuous increase in the model's prediction accuracy. This metric is used in this thesis to provide a level-based metric system for comparing accuracy with scores [103]. The mean square error (MSE) is mathematically represented as:

$$\text{MSE} = \frac{\sum_{j=1}^{M_{\text{test}}} |y_{\text{true}}^j - y_{\text{predicted}}^j|^2}{M_{\text{test}}} \quad (16)$$

3.4. Base Learners Selection Technique

There is no literature stating outrightly the type of algorithms selected as the base learners to be implemented in the development of the super learner model. In this section, a novel technique was developed to determine the best machine learning algorithms used as base learners implemented in the super learner model. As seen in figure 7, ten supervised learning algorithms were selected for the base learner's selection technique flow. These algorithms were selected because they are well-known regression algorithms capable of learning the features present in our dataset, a regression problem. Firstly, the 3C steel corrosion rate dataset, which the algorithms will learn, is imported (in CSV format) using the pandas import function `pd.read_csv` into the Spyder IDE. In addition to the 3C steel corrosion rate dataset, the ten algorithms' libraries and modules were imported into Spyder via algorithm-specific scikit learn functions. These libraries and modules are

necessary for the algorithms' functionality as they serve as the architecture of each algorithm. Their absence will lead to an error if the selection system is run as the algorithms need to call on certain arguments, methods, and functions embedded within their respective libraries and modules.

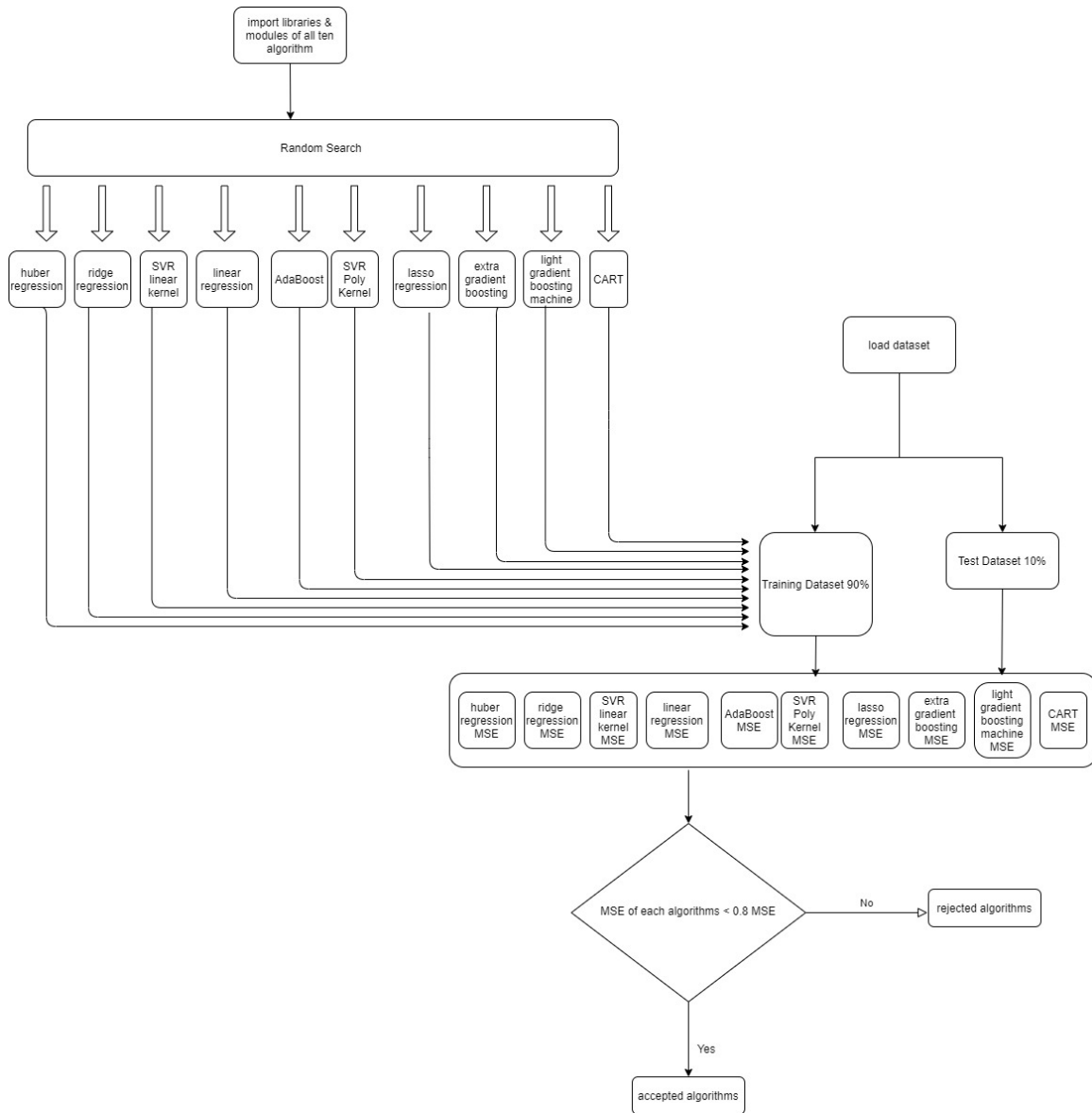


Figure 7 Flow chart showing the development of the base learners selection technique

Next, a standard machine learning paradigm is introduced: the data head and data tail function. The data head is the first couple of rows of the dataset displayed to facilitate the proper loading of the dataset. In python, the pandas 'head method' is used to return the top n (5 by default) rows of the dataset. The data tail is the last couple of rows of the dataset presented to facilitate the proper loading of the dataset. In python, the pandas 'tail method' is used to return the bottom n (5 by default) rows of the dataset. The pandas head `data.head()`, and tail function `data.tail()` are used to carry out this task to double-checking the dataset before running the model.

The model selection is subsequently implemented to split the input data frame (X). The target features series (y) into training and test datasets using an imported scikit learn method known as `model_selection` method. The splitting process ensured 90% of the entire dataset was allotted for training data and 10% allotted for test data while keeping the splitting process's random state at 1.

Furthermore, each of the ten algorithms is then fitted and assessed simultaneously on the 90% training dataset and 10% test dataset (consisting of five sample points of the dependent features) to develop individual prediction models. The predictions (`y_pred`) of each model are utilized along with the output test dataset (`y_test`) in the formulation of mean square error (MSE) values for each model, as seen in figure 8. Mean square error was used instead of R squared and RMSE because of its ability to penalize large errors compared to R squared [104].

| Key | Type | Size | Value |
|-------------|---------|------|--------------------|
| adaBoostMSE | float64 | 1 | 1.3979479999999995 |
| dtrMSE | float64 | 1 | 1.3076986499999983 |
| huberMSE | float64 | 1 | 0.9142500907488669 |
| lassoMSE | float64 | 1 | 0.6648448380473356 |
| lgbmMSE | float64 | 1 | 1.8929265028324576 |
| linregMSE | float64 | 1 | 0.6441488663970067 |
| ridgeMSE | float64 | 1 | 0.6465366890188768 |
| svrlinMSE | float64 | 1 | 1.9552119853758216 |
| svrpolyMSE | float64 | 1 | 0.5440057893921365 |
| xgbMSE | float64 | 1 | 0.7709462410698095 |

Figure 8 MSE scores of all ten models in spyder IDE's variable explorer

Subsequently, a score of 0.8 is taken as the selection threshold. Therefore, models with MSE scores below 0.8 are accepted, while those with values above the threshold are rejected. In addition, the selection framework also gives us a baseline of hyperparameters (for each accepted model) to be implemented in the super learner. The accepted models of the algorithms and their equivalent MSE scores are extra gradient boosting (xgbMSE), lasso regression (lassoMSE), linear regression (linregMSE), ridge regression (ridgeMSE), and support vector regression using the polynomial kernel (svrpolyMSE), as seen in figure 9.

```

Console 3/A
['svrpolyMSE', 'ridgeMSE', 'linregMSE', 'lassoMSE', 'xgbMSE']
In [2]:

```

Figure 9 Spyder IDE terminal window view of the accepted models

3.5. Development of the Super Learner Model

With all the libraries of all five accepted base learners on the Spyder canvas, the super learner ensemble is launched using the `SuperLearner()` method. This method houses the SVR libraries (Polynomial kernel), lasso regression, ridge regression, linear regression, and extreme gradient boosting, as well as the baseline hyperparameters of each model. In addition, the only important hyperparameter of the super learner, i.e., fold, is kept at 2 (i.e., `fold=2`) since we have two layers implemented in its architecture. All the base learners are fitted on the training data with the fitted estimators stored separately. Subsequently, the training data is split into two folds (each containing a subset of training data and test data) based on the super learner hyperparameter, after which each of the five base learners is fitted on each fold, and predictions carried out on the test datasets in each fold sequentially. Each fold's predictions are then stacked upon each other based on the super learner model being an ensemble. The linear regression model is implemented in the meta learner, which is subsequently fitted on the prediction matrix to produce a fitted meta learner. Figure 2 is a diagrammatic representation of a super learner fitting process.

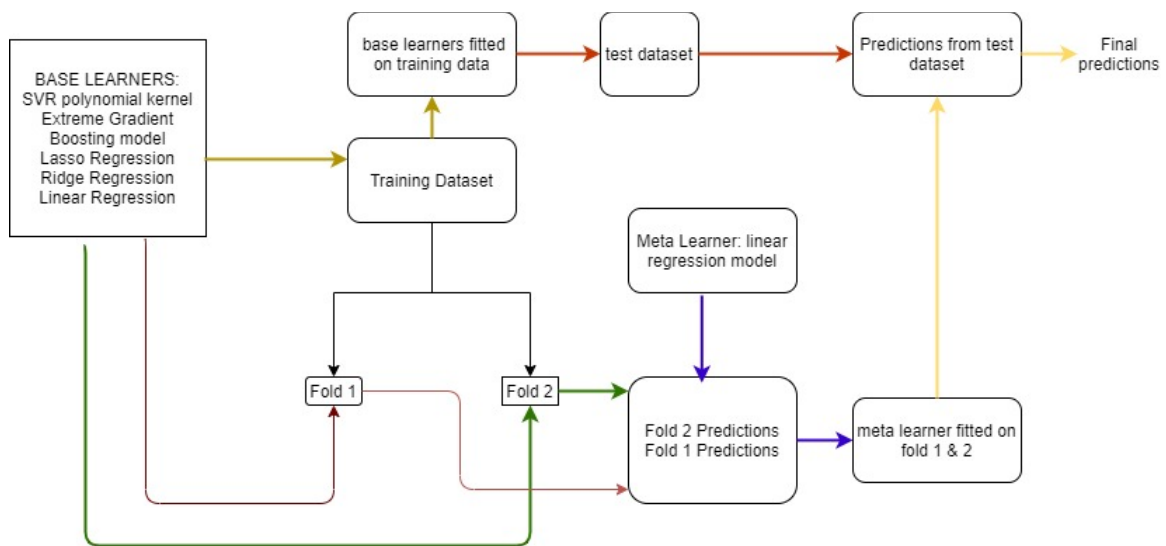


Figure 10 Flow representation of the super learner model fitting process

The fitted estimators (formed by fitting each of the base learners on the entire training data) were fitted on the new test dataset to carry out predictions. This fitting process facilitated the entry of the test dataset into the prediction matrix. The final

predictions based on the test dataset were generated using the fitted meta learner, which converted the prediction matrix into corrosion rate predictions. Figure 10 shows a flow representation of the fitting process of the developed super learner.

3.6. System Characteristics

The system implemented in the execution of the super learner model possessed the following characteristics:

Table 1 System Characteristics

| | |
|--------------------------|--|
| OS Name | Microsoft Windows 10 Home |
| System Model | HP ENVY x360 Convertible 15-cn0xxx |
| System Type | X64-based PC |
| Processor | Intel® Core™ i5-8250U @ 1.60GHz, 1800Mhz |
| Memory | 8.00 GB DDR4 SDRAM 1199MHz |
| Base Frequency | 1.80GHz |
| Maximum Frequency | 3.39GHz |

Chapter 4.

Evaluation of the Super Learner Model

In this chapter, the developed super learner models' performance is evaluated using root mean square error, mean square error, and R- squared error. Not only are these metrics the best evaluation metrics utilized for supervised learning regression problems, but they also allow us to compare the prediction capability of our model with not only the algorithms that make up its architecture but also the prediction capability of models developed on the same training data [10,11]. For these comparisons to occur, the Smart Firefly Algorithm & Least Squares Support Vector Regression (SFA-LSSVR) and Support Vector Regression integrating Leave Out One Cross-Validation (SVR-LOOCV) models were developed (following the model architectural design detailed in the works of literature) to generate an RMSE score similar to the scores in [10,11]. The methods employed in the presentation and evaluation of results in this chapter span from tables to histograms.

4.1. Test Dataset

The dataset used to test the super learner model was extracted from the dataset itself. This is because the dataset was generated from an experiment on studying the corrosion rate of 3C steel in different marine environmental conditions [10]. This dataset is foreign to the model due to its absence in the dataset used to train the super learner. Hence, the model is tested on a fresh dataset, taking out the possibility of high variance and subsequent overfitting. The test dataset is 10% of the entire dataset and comprises five randomly selected data points. The random selection was automatically carried out by the `model_selection` method imported from the scikit learn library. The dataset takes the form of the features variables (`X_test`) containing the five environmental parameters and the target variable (`X_test`), corrosion rate. From the two forms, the target variable is mapped to the `ensemble.Predict` the method of the super learner model to estimate the corrosion rate. Figure 11 shows the feature set and target variable dataset extracted from the variable explorer of spyder IDE.

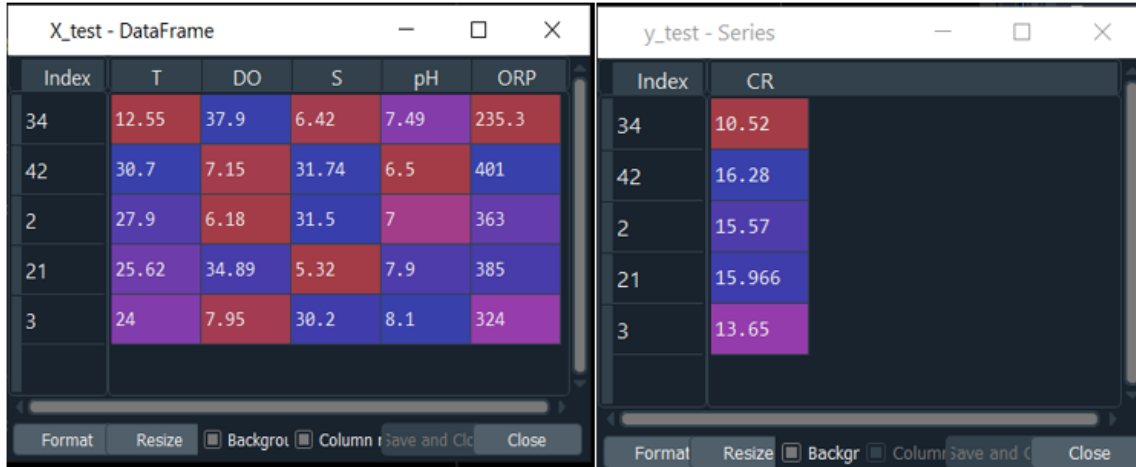


Figure 11 feature set and target test dataset

4.2. Test Results

This section outlines the corrosion rate predicted by the super learner model. From the results, the developed model provides a comparable prediction accuracy against the Smart Firefly Algorithm & Least Squares Support Vector Regression (SFA-LSSVR) [11] and outperforms the Support Vector Regression integrating Leave Out One Cross-Validation (SVR-LOOCV) model [10].

From table 1, the mean absolute percentage error (MAPE) formulated from the percentage error column indicates the presence of a 1.41% mean absolute percentage error in the predictions made by the super learner model in comparison to the 1.26% and 3.2% mean absolute percentage error present in the predictions of the SFA-LSSVR and SVR-LOOCV models respectively. The root mean square error will be solely implemented to evaluate the models due to the mean absolute percentage error values being excessively too large or undefined [99].

Table 2 Mean absolute percentage error of the models

| Models | Mean Absolute Percentage Error (%) |
|---------------|------------------------------------|
| Super learner | 1.41 |
| SFA-LSSVR | 1.26 |
| SVR-LOOCV | 3.2 |

A visual aid is necessary to analyze further the distance between the experimental and predicted rates outlined in table 1. Figure 12 is a line plot showing the relationship between the experimental rate (actual values), predicted rate (predicted values), and the percentage error. From the figure, the distance between the true values and the predicted values represents the percentage error. Therefore, trials 0, 1, 2, 3 & 4 of the super learner have minimal percentage error, with just trial 0 being the highest at 7.6% error and trial 4 having the least percentage error of 0.5%. All other trial points have their percentage errors lying between both error points with a median percentage error of 4.3%

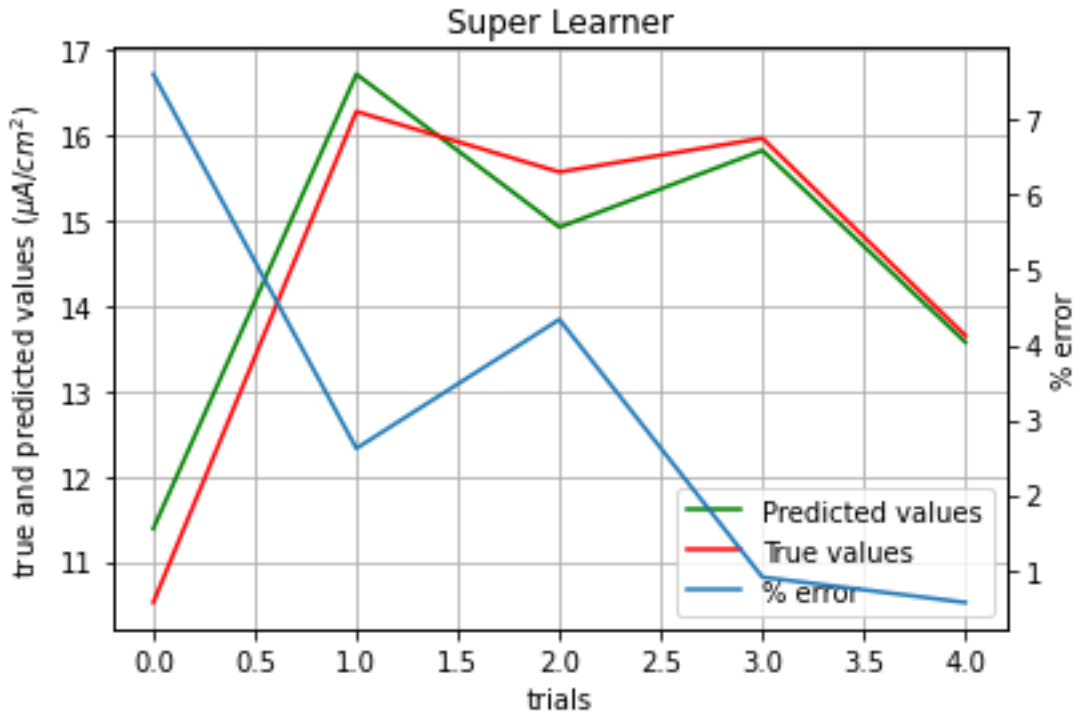


Figure 12 Line plot of the predicted values, actual values, and percentage error of the super learner model

Comparing the plot of figure 12 to the visualization of each of the constituent models of the super learner model in figure 13 shows not only the vast difference amongst the results of the models (super learner model included) but also a visual estimate of how much correction was carried out by the super learner model to generate accurate corrosion rate predictions. In figure 13, the extra gradient boosting machine plot indicates that the model generated the most percentage error (14.65%) on the test dataset compared to not just the constituent models but the super learner model itself. Another observation is the near similarity in plot between the ridge regression model and the linear regression model, which stems from the fact that the ridge regression model is a close variant of the linear regression model and hyperparameter optimization gives a mirror-like prediction ability of its parent regression model. Compared to the ridge regression and linear regression model, the lasso regression model generated a nearly perfect inverted-V error line plot, not conforming to a nearly similar prediction power of its parent model (i.e., linear regression model). Lastly, the linear, ridge, and lasso regression models generated nearly similar median scores (based on their percentage error values) of 8.47%, 8.45%, and 8.7%, respectively. All of which are inferior to the median value of the super learner model.

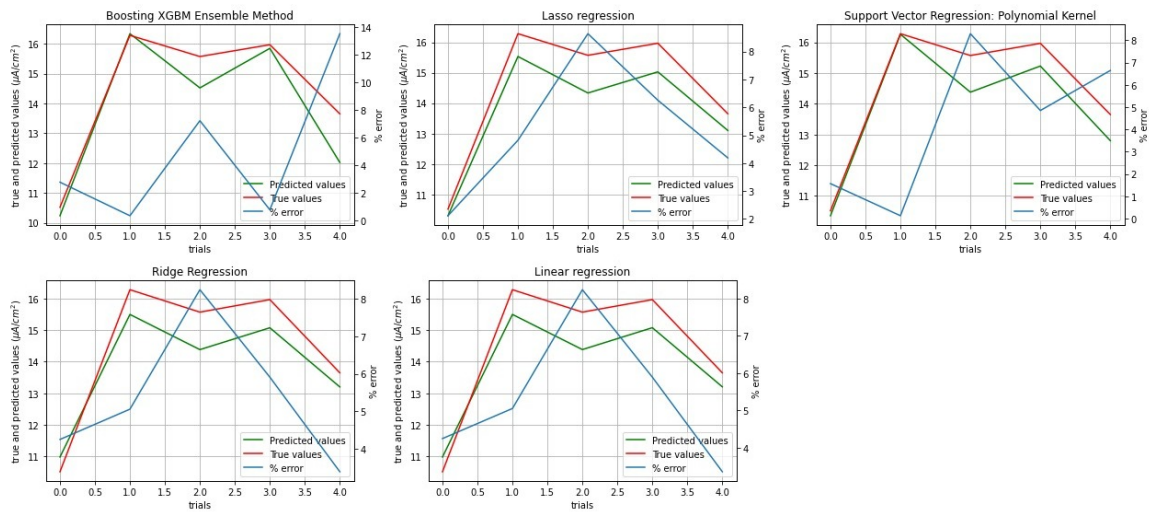


Figure 13 Line plots of extra gradient boosting machine, lasso regression, polynomial kernel of the support vector machine, ridge regression, and linear regression models

The results from the performance evaluation of the super learner model are outlined in table 2. In comparison with its constituent models, the super learner's performance was evaluated using the root mean square error, mean square error, and R squared error metric. Based on these metrics' results, the super learner's lower RMSE score in comparison to the RMSE score of its constituent models suggests that the super learner model fits better than its constituent models. The same principle applies to its lower MSE score in comparison to that of its constituent models. In the case of R Squared, the superiority of the super learner model is shown due to the closeness of its score to 1, indicating that the super learner model predicted the corrosion rate (based on the test dataset) with very high accuracy compared to its constituent models possessing R squared scores far below therefore indicating their poor prediction accuracy.

Table 3 Performance evaluation result of the super learner model and its constituent models

| Model | RMSE | MSE | R2 |
|------------------------|-------|-------|-------|
| Super Learner | 0.485 | 0.235 | 0.949 |
| Extra Gradient Boost | 0.878 | 0.771 | 0.832 |
| Lasso Regression | 0.82 | 0.66 | 0.86 |
| SVR: Polynomial Kernel | 0.737 | 0.544 | 0.882 |
| Linear Regression | 0.803 | 0.644 | 0.860 |
| Ridge Regression | 0.804 | 0.646 | 0.859 |

Due to the ruthless nature of the root mean square error (RMSE) metric towards outliers [98] and its use as the major performance evaluation metric for the SFA-LSSVR and SVR-LOOCV model, a bar chart is generated in figure 14 to compare the RMSE scores of the super learner model to the SFA-LSSVR model, SVR-LOOCV model and the constituent models of the super learner model. From the plot, the taller bars indicate the degree of inaccuracy of the representing model. This inaccuracy representation is greatest, as shown by the light blue bar's length representing the SVR-LOOCV model and then followed by the bars representing the constituent models. A huge reduction in the bars'

height (compared with the bars of the constituent models and the SVR-LOOCV model) representing the super learner model and SFA-LSSVR model is experienced. This reduction shows that the SFA-LSSVR model predictions possess the least amount of errors closely followed by the super learner model.

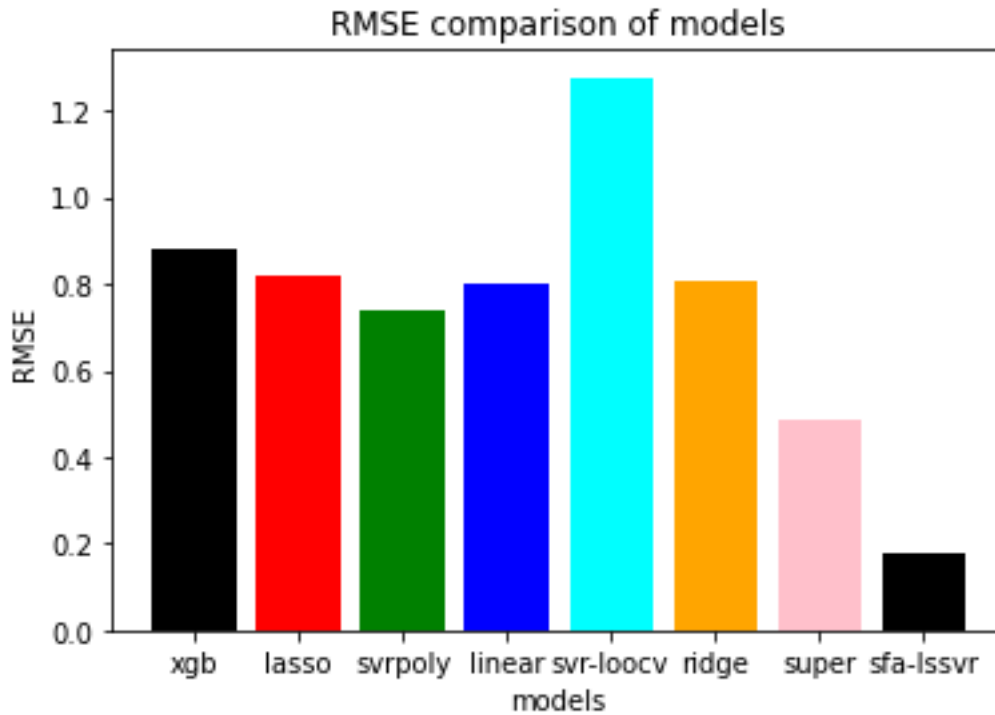
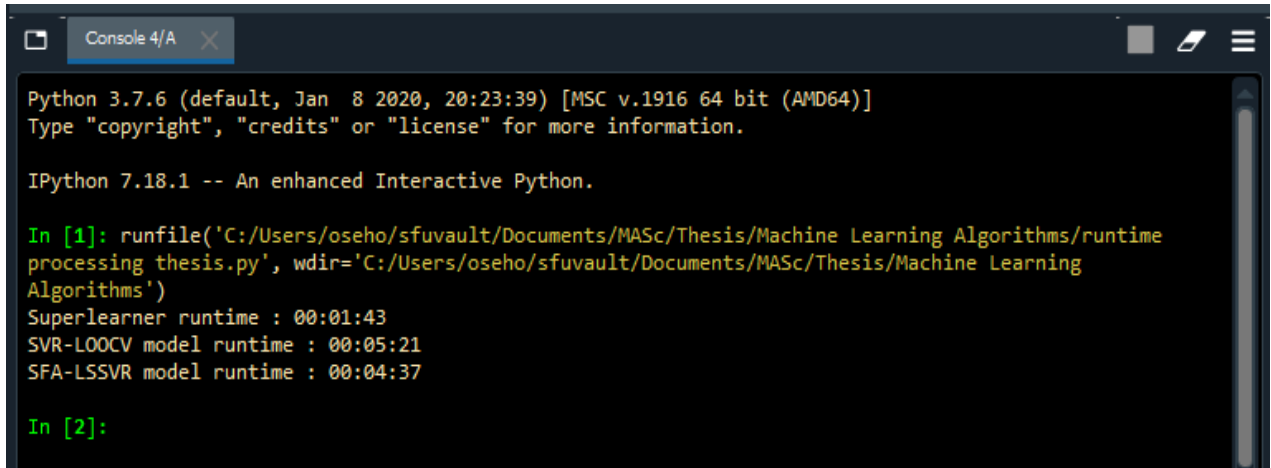


Figure 14 Bar chart showing the root mean square values of the super learner model, SFA-LSSVR model, SVR-LOOCV model, and the constituent models of the super learner model

Despite the SFA-LSSVR model slightly outperforming the super learner model in terms of prediction accuracy when evaluated with the root mean square, the super learner model makes up for its loss by utilizing less system memory (faster computation time). This is due to the models' utilization of the `mlens` framework. The SFA-LSSVR and SVR-LOOCV models were developed (while following the model architecture detailed in the literatures) to facilitate the memory utilization and computation time speed comparison between them and the super learner model. Proper hyperparameter optimization was carried out to ensure the SFA-LSSVR and SVR-LOOCV models generated an RMSE values approximately similar to the values generated in their respective works of literature. Figure

15 shows the execution or computation time of the super learner model, the SFA-LSSVR model, and the SVR-LOOCV model. From the figure, the super learner model utilizes the least computation time closely followed by the SFA-LSSVR model while the SVR-LOOCV model utilizes the most execution time.



```
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.18.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/oseho/sfuvault/Documents/MASc/Thesis/Machine Learning Algorithms/runtime
processing thesis.py', wdir='C:/Users/oseho/sfuvault/Documents/MASc/Thesis/Machine Learning
Algorithms')
Superlearner runtime : 00:01:43
SVR-LOOCV model runtime : 00:05:21
SFA-LSSVR model runtime : 00:04:37

In [2]:
```

Figure 15 Computation time of the super learner model, its constituent models, the SFA-LSSVR model, and the SVR-LOOCV model

The 3D clustered bar plot in figure 16 shows the super learner model's memory, its constituent models, the SFA-LSSVR model, and the SVR-LOOCV model. From the figure, the super learner model possesses a better memory usage in comparison to the other models with the SVR-LOOCV model possessing the least memory utilization ability. Memory consumption test was carried out by executing each model on one after the other on spyder IDE and simultaneously monitoring the memory consumed on the task manager of a windows 10 PC. This test was carried out three times for each model and the results from each test were averaged to give what is presented in figure 16. The memory usage and computation time was determined using the friedmann module

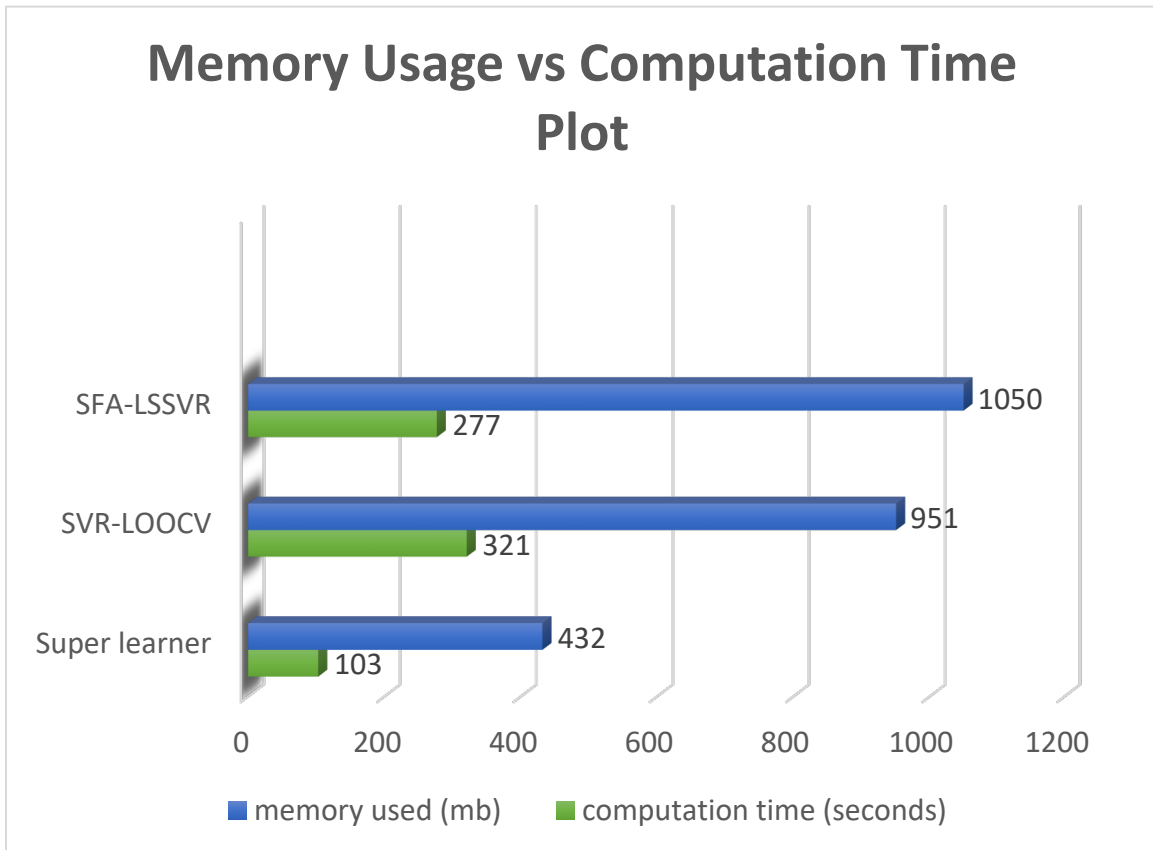


Figure 16 Memory usage and computation time of the super learner model, its constituent models, the SFA-LSSVR model, and the SVR-LOOCV model

From the results detailed in this chapter, it is clear to conclude that the super learner model's prediction accuracy (in comparison with that of the SFA-LSSVR and SVR-LOOCV model) can be ignored given the super learner models' superiority in the area of memory utilization and computation time especially in applications requiring large datasets, modules, objects, and functions.

Chapter 5.

Conclusions

5.1. Summary

We presented a machine learning model for the prediction of the corrosion rate of 3C steel in five different seawater environments (with differences in temperature (T), dissolved oxygen (DO), salinity (Sal), pH value (pH), oxidation-reduction potential (ORP)). These five environmental conditions represent the dataset's feature variables on which the super learner model was fitted. The super learner's constituent models, the SFA-LSSVR model, and the SVR-LOOCV model were also fitted on the dataset's feature variables. Also presented was the design of a selection framework used to select the algorithms implemented in the development of the constituent model and optimize the hyperparameters of the constituent models for easy plug and play into the super learner model architecture. As seen in table 3, the super learner model's prediction capability was judged to perform below the SFA-LSSVR model slightly but outperformed the SVR-LOOCV model.

The super learner model capitalizes on this unfair disadvantage by utilizing less computation time and memory than the SFA-LSSVR model and SVR-LOOCV model, both of which utilize more computation time and memory the numerous iterations of parameters optimization within its architecture. The super learner model displayed its high accuracy and efficient memory utilization at less computation time as it is approximately three times faster than the SFA-LSSVR model while utilizing approximately 59% of the memory consumed by the SFA-LSSVR model; it is four times faster than the SVR-LOOCV model while utilizing 54.6% of the memory consumed by the same SVR-LOOCV model. Also, the results from the super learner model not only elude any high bias and low variance problems but also successfully capture the complexity of the dataset. The results presented in this thesis also highlighted the super learner's model ability to make predictions based on data instances not trained on. The super learner model generalization is far better than the SVR-LOOCV model while slightly below the generalization strength of the SFA-LSSVR model. The RMSE score presented in table 2

highlights the superiority of the super learner model generalization compared to the generalization of its constituent models.

The super learner model offers a new approach for estimating the corrosion rate of 3C steel in sea water with high accuracy and less computation time than the hybrid metaheuristic regression model (SFA-LSSVR). With minor adjustments, this model and the entirety of its architecture could be employed to predict the corrosion rate of other metals within seawater conditions. Conditions outside of seawater (e.g., land and air) will require numerous adjustments to the model and its selection framework, and the packages implemented. Table 3 outlines how this model compares with others from the literature.

Table 4 Prediction accuracy comparison between the super learner, SFA-LSSVR, and SVR-LOOCV model

| Model | RMSE | Computation Time (seconds) |
|---------------------|-------|----------------------------|
| Super Learner model | 0.485 | 103 |
| SFA-LSSVR [11] | 0.180 | 277 |
| SVR-LOOCV [10] | 1.277 | 321 |

Advancement in machine learning has seen the properties of physical GPUs and operating systems being replicated on the cloud to facilitate the usage of the latest GPUs in developing machine learning models. These GPUs are expensive with an increase in specifications, e.g., memory size, processor speed etc. With the availability of large datasets in the form of big data, the need for models that can deliver accurate predictions while efficiently utilizing memory (and execution at a faster computation time) is paramount. The super learner model can meet these needs. It can be utilized instead of neural networks due to the super learner model's superiority in memory utilization and computation time. To meet the desired level of accuracy required in this thesis on an affordable GPU on the cloud, neural network models would likely require the utilization of numerous layers to its architecture resulting in a network with several nodes greater than that of a basic multilayer perceptron. The utilization of numerous layers within the neural network model architecture would thus ultimately lead to very slow computation time during the model's execution, unlike the super learner model that utilizes just two-fold and five algorithms to attain the required level of prediction while leveraging memory and computation time.

5.2. Recommendation for future work

Major future work on this model will be improving the prediction accuracy of the model. This can be achieved by increasing the dataset's size via more seawater environmental conditions that correlate with the other five conditions. The higher the correlation between the features of the variables, the greater the accuracy of the estimated corrosion rate of the 3C steel. The dataset can also be increased by adding more trials during the experiment's execution, which should be followed with proper dataset preprocessing to account for large outliers and missed data points.

The Adaptive Corrosion Protection System (ACPS) utilizes corrosion rate (using corrosion current and corrosion potential) for the protection of electrical tower grillage built on the land. This facilitates a potential utilization of the developed super learner model in the ACPS to provide accurate corrosion rate values to be implemented in the protection algorithm for potential protection of electrical tower grillage structures under seawater. The potential utilization of the super learner model (for underwater tower grillage protection) eliminates the challenge of additional signal processing (towards better memory utilization) as the lightweight nature of the model will facilitate its smooth execution within the processors of the raspberry pi modules utilized in the adaptive corrosion protection system (ACPS).

Furthermore, with Python being the language upon which the super learner model is developed, the synchronization between the visualizations of the model and the graphic user interface of the ACPS (also written in python) is improved for potential use in the corrosion protection of underwater tower grillages. In addition, this synchronization also provides a foundation for further upgrades to the graphic user interface of the ACPS using the machine learning visualization libraries and modules.

References

- [1] M. Clarke, "Corrosion and corrosion control, an introduction to corrosion science and engineering," *Corrosion Science*, vol. 4, no. 1–4, pp. 372–373, Jan. 1964, doi: 10.1016/0010-938x(64)90035-6.
- [2] R. Singh, "Introduction to a New Journal: Corrosion and Materials Degradation," *Corrosion and Materials Degradation*, vol. 1, no. 1, pp. 1–2, Apr. 2018, doi: 10.3390/cmd1010001.
- [3] G. H. Koch, M. P. H. Brongers, N. G. Thompson, Y. P. Virmani, and J. H. Payer, "Corrosion Costs and Prevention Strategies in the United States," [nace.org. https://www.nace.org/uploadedFiles/Publications/ccsupp.pdf](https://www.nace.org/uploadedFiles/Publications/ccsupp.pdf) (accessed Sep. 15, 2020).
- [4] "The Global Cost and Impact of Corrosion," *Inspectioneering.com*, 2016. <https://inspectioneering.com/news/2016-03-08/5202/nace-study-estimates-global-cost-of-corrosion-at-25-trillion-ann>.
- [5] J. C. Schouten and P. J. Gellings, "Quantitative measures of corrosion and prevention: application to corrosion in agriculture," *Journal of Agricultural Engineering Research*, vol. 36, no. 3, pp. 217–231, Mar. 1987, doi: 10.1016/0021-8634(87)90075-8.
- [6] Nace.org, 2013. <http://impact.nace.org/economic-impact.aspx>.
- [7] G. Koch, B. Thompson, N. Thompson, Y. Virmani, and J. H. Payer, "Corrosion Costs and Preventive Strategies in the United States," US Department of Transportation.
- [8] G. Vatchsevanos, K. A. Natarajan, R. Rajamani, and P. Sandborn, *Corrosion processes: sensing, monitoring, data analytics, prevention/protection, diagnosis/prognosis and maintenance strategies*, vol. 13. Switzerland: Springer Nature Switzerland AG 2020, 2020.
- [9] "Land Transportation System: Road, Railways and Pipelines (with maps)," *Your Article Library*, Feb. 07, 2014. <https://www.yourarticlelibrary.com/transport/land-transportation-system-road-railways-and-pipelines-with-maps/25366> (accessed Dec. 11, 2020).
- [10] Y. Wen, C. Cai, X. Liu, J. Pei, X. Zhu, and T. Xhao, "Corrosion rate prediction of 3C steel under different seawater environment by using support vector regression," Elsevier: *Corrosion Science*, 2008.
- [11] J. Chou, N. Ngo, and W. Chong, "The use of artificial intelligence combiners for modeling steel pitting risk and corrosion rate," Elsevier: *Engineering Applications of Artificial Intelligence Elsevier Corrosion Science*, 2016.

- [12] E. Specht and R. Jeschar, "Kinetics of steel melting in carbon-steel alloys," *Steel Research*, vol. 64, no. 1, pp. 28–34, Jan. 1993, doi: 10.1002/srin.199300978.
- [13] C. Bejinariu, D.-P. Burduhos-Nergis, and N. Cimpoesu, "Immersion Behavior of Carbon Steel, Phosphate Carbon Steel and Phosphate and Painted Carbon Steel in Saltwater," *Materials*, vol. 14, no. 1, p. 188, Jan. 2021, doi:10.3390/ma14010188.
- [14] T. OGURA, "Stainless Steel for Vacuum Industry Use," *SHINKU*, vol. 5, no. 9, pp. 347–358, 1962, doi: 10.3131/jvsj.5.347.
- [15] Pipelines And Informed Planning Alliance and United States. Pipeline And Hazardous Materials Safety Administration. Office Of Pipeline Safety, Hazard mitigation planning : practices for land use planning and development near pipelines. Washington, D.C.: The Office, 2015.
- [16] A. Dobuzinskis, "Kentucky gas line erupts in ball of flames, killing one and igniting homes," Reuters, Aug. 01, 2019.
- [17] B. Brody and A. Weinstein, "Energy Transfer Gas Rupture Caps Missouri Mainline Capacity," Bloomberg.com, Mar. 03, 2019.
- [18] O. C. News, "3 Injured in Wyoming Oil and Gas Field Explosion," SweetwaterNOW, Dec. 06, 2019. <https://www.sweetwaternow.com/3-injured-in-wyoming-oil-and-gas-field-explosion/>.
- [19] M. Moigne and L. Laubier, "The 'Erika' Oil spill: Environmental Contamination and Effects in the Bay of Biscay," *Aquatic Living Resources*, vol. 17, pp. 235–236, 2004, doi: 10.1051/alr:2004045.
- [20] "Silver Bridge," Wv.gov, 2020. https://transportation.wv.gov/highways/bridge_facts/Modern-Bridges/Pages/Silver.aspx.
- [21] "MCF - Marine Corrosion Explained," www.marinecorrosionforum.org. <https://www.marinecorrosionforum.org/explain.htm#:~:text=Corrosion%20by%20sea%20water%2C%20aqueous> (accessed Dec. 23, 2020).
- [22] M. G. Fontana, "Erosion-Corrosion Resistance of Metals CORROSION.," *Industrial & Engineering Chemistry*, vol. 44, no. 9, pp. 101A-104A, Sep. 1952, doi: 10.1021/ie50513a006.
- [23] G. Fajardo, G. Escadeillas, and G. Arliguie, "Electrochemical chloride extraction (ECE) from steel-reinforced concrete specimens contaminated by 'artificial' sea-water," *Corrosion Science*, vol. 48, no. 1, pp. 110–125, Jan. 2006, doi: 10.1016/j.corsci.2004.11.015.

- [24] S. Feliu, M. Morcillo, and B. Chico, "Effect of state of sea on atmospheric corrosion in coastal zones," *British Corrosion Journal*, vol. 36, no. 2, pp. 157–160, Feb. 2001, doi: 10.1179/000705901101501604.
- [25] M. Schneider and K. Galle, "Investigation of the initial stage of crevice corrosion on Al99.5 by electrochemical noise analysis," *Materials and Corrosion*, vol. 58, no. 12, pp. 983–991, Dec. 2007, doi: 10.1002/maco.200704092.
- [26] P. R. Roberge, *Corrosion Basics : an introduction*. Houston, Tx, United States Of America Nace International, 2018.
- [27] L. Yang and Institute Of Materials, Minerals, And Mining, *Techniques for corrosion monitoring*. Cambridge, England: Woodhead Publishing ; [England, 2008.
- [28] J. N. Defrancq, "Linear polarization in the presence of a changing corrosion potential," *Corrosion Science*, vol. 16, no. 9, pp. 645–647, Jan. 1976, doi: 10.1016/s0010-938x(76)80023-6.
- [29] Z. Shi, M. Liu, and A. Atrens, "Measurement of the corrosion rate of magnesium alloys using Tafel extrapolation," *Corrosion Science*, vol. 52, no. 2, pp. 579–588, Feb. 2010, doi: 10.1016/j.corsci.2009.10.016.
- [30] K. Darowicki, "Corrosion rate measurements by non-linear electrochemical impedance spectroscopy," *Corrosion Science*, vol. 37, no. 6, pp. 913–925, Jun. 1995, doi: 10.1016/0010-938x(95)00004-4.
- [31] S. Feliu, "Electrochemical Impedance Spectroscopy for the Measurement of the Corrosion Rate of Magnesium Alloys: Brief Review and Challenges," *Metals*, vol. 10, no. 6, p. 775, Jun. 2020, doi: 10.3390/met10060775.
- [32] A. Pardo, S. Feliu, M. C. Merino, R. Arrabal, and E. Matykina, "Electrochemical Estimation of the Corrosion Rate of Magnesium/Aluminium Alloys," *International Journal of Corrosion*, vol. 2010, pp. 1–8, 2010, doi: 10.1155/2010/953850.
- [33] V. A. Makarov, G. M. Florianovich, and Ya. M. Kolotykin, "Theoretical aspects of the electrochemical protection of metals in aggressive media," *Soviet Materials Science*, vol. 22, no. 6, pp. 541–548, 1987, doi: 10.1007/bf00718098.
- [34] G. M. Florianovich, "Electroless dissolution of metals: Substantiation and alternative notions," *Russian Journal of Electrochemistry*, vol. 36, no. 10, pp. 1037–1042, Oct. 2000, doi: 10.1007/bf02757521.
- [35] D. Drazic and J. Popic, "Anomalous dissolution of metals and chemical corrosion," *Journal of the Serbian Chemical Society*, vol. 70, no. 3, pp. 489–511, 2005, doi: 10.2298/jsc0503489d.

- [36] M. Raupach, "Investigations on the influence of oxygen on corrosion of steel in concrete—Part I," *Materials and Structures*, vol. 29, no. 3, pp. 174–184, Apr. 1996, doi: 10.1007/bf02486163.
- [37] S. Zukhrufany, "The Utilization of Supervised Machine Learning in Predicting Corrosion to Support Preventing Pipelines Leakage in Oil and Gas Industry," Universitas Stavager : Faculty of Science and Technology, 2018.
- [38] "Machine learning,"
http://en.wikipedia.org/wiki/Machine_learning.
- [39] B. Zaman, Remotely Sensed Data Assimilation Technique to Develop Machine Learning Models for Use in Water Management. Ph.D. dissertation, Utah State University, Logan, UT, 2010.
- [40] S. Gorthi and H. Dou, "Prediction models for estimation of soil moisture content," in 2011 ASME /IEEE International Conference on Mechatronic and Embedded Systems and Applications (MESA2011), Washington, DC, August 28-31, 2011.
- [41] S. Ahmad, A. Kalra, and H. Stephen, "Estimating soil moisture using remote sensing data: A machine learning approach," *Advances in Water Resources*, vol. 33, pp. 69–80, 2010.
- [42] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, Jan. 1992, doi: 10.1016/s0893-6080(05)80023-1.
- [43] L. Breiman, "Stacked regressions," *Machine Learning*, vol. 24, no. 1, pp. 49–64, Jul. 1996, doi: 10.1007/bf00117832.
- [44] M. J. van der Laan, S. Dudoit, and A. W. van der Vaart, "The cross-validated adaptive epsilon-net estimator," *Statistics & Decisions*, vol. 24, no. 3, Jan. 2006, doi: 10.1524/std.2006.24.3.373.
- [45] M. J. van der Laan, E. C. Polley, and A. E. Hubbard, "Super Learner," *Statistical Applications in Genetics and Molecular Biology*, vol. 6, no. 1, Jan. 2007, doi: 10.2202/1544-6115.1309.
- [46] M. A. Hedeya, A. H. Eid, and R. F. Abdel-Kader, "A Super-Learner Ensemble of Deep Networks for Vehicle-Type Classification," *IEEE Access*, vol. 8, pp. 98266–98280, 2020, doi: 10.1109/access.2020.2997286.
- [47] A. I. Naimi and L. B. Balzer, "Stacked generalization: an introduction to super learning," *European Journal of Epidemiology*, vol. 33, no. 5, pp. 459–464, Apr. 2018, doi: 10.1007/s10654-018-0390-z.
- [48] S. Weisberg, *Applied linear regression*. Chichester: Wiley-Blackwell, 2014.

- [49] P. J. Twomey and M. H. Kroll, "How to use linear regression and correlation in quantitative method comparison studies," *International Journal of Clinical Practice*, vol. 62, no. 4, pp. 529–538, Mar. 2008, doi: 10.1111/j.1742-1241.2008.01709.x.
- [50] A. L. Edwards, *An introduction to linear regression and correlation*. Oxford: Freeman, 1984.
- [51] Yu. V. Kuk and Yu. I. Petunia, "Linear regression parameter estimation in the presence of constraints on linear regression coefficients," *Ukrainian Mathematical Journal*, vol. 28, no. 4, pp. 357–364, 1977, doi: 10.1007/bf01101655.
- [52] R. G. Willey and H. E. Kubik, *Multiple linear regression*. Davis, Calif.: Hydrologic Engineering Center, [197].
- [53] K. A. Marill, "Advanced Statistics: Linear Regression, Part II: Multiple Linear Regression," *Academic Emergency Medicine*, vol. 11, no. 1, pp. 94–102, Jan. 2004, doi: 10.1197/j.aem.2003.09.006.
- [54] M. S. Dueker and M. Nabel, "Statistical Analysis Software Package: NCSS from a Teacher's Viewpoint," *The American Statistician*, vol. 46, no. 3, p. 226, Aug. 1992, doi: 10.2307/2685221.
- [55] P. C. Bruce, A. Bruce, and P. Gedeck, *Practical statistics for data scientists : 50+ essential concepts using R and Python*. Sebastopol, Ca: O'reilly Media, Inc, 2020.
- [56] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, Feb. 2000, doi: 10.1080/00401706.2000.10485983.
- [57] E. Cule and M. De Iorio, "Ridge Regression in Prediction Problems: Automatic Choice of the Ridge Parameter," *Genetic Epidemiology*, vol. 37, no. 7, pp. 704–714, Jul. 2013, doi: 10.1002/gepi.21750.
- [58] L. Peele and T. P. Ryan, "Minimax Linear Regression Estimators With Application to Ridge Regression," *Technometrics*, vol. 24, no. 2, pp. 157–159, May 1982, doi: 10.1080/00401706.1982.10487740.
- [59] G. Abraham, A. Kowalczyk, J. Zobel, and M. Inouye, "Performance and Robustness of Penalized and Unpenalized Methods for Genetic Prediction of Complex Human Disease," *Genetic Epidemiology*, vol. 37, no. 2, pp. 184–195, Nov. 2012, doi: 10.1002/gepi.21698.
- [60] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, Jan. 1996, doi: 10.1111/j.2517-6161.1996.tb02080.x.

- [61] R. J. Tibshirani, "The lasso problem and uniqueness," *Electronic Journal of Statistics*, vol. 7, no. 0, pp. 1456–1490, 2013, doi: 10.1214/13-ejs815.
- [62] L. Meier, S. Van De Geer, and P. Bühlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 53–71, Jan. 2008, doi: 10.1111/j.1467-9868.2007.00627.x.
- [63] M. Kyung, J. Gill, M. Ghosh, and G. Casella, "Penalized regression, standard errors, and Bayesian lassos," *Bayesian Analysis*, vol. 5, no. 2, pp. 369–411, Jun. 2010, doi: 10.1214/10-ba607.
- [64] S. R. Sain and V. N. Vapnik, "The Nature of Statistical Learning Theory," *Technometrics*, vol. 38, no. 4, p. 409, Nov. 1996, doi: 10.2307/1271324.
- [65] X.-J. DING and Y.-L. ZHAO, "Support Vector Regression Optimization Problem without Bias," *Journal of Software*, vol. 23, no. 9, pp. 2336–2346, Nov. 2012, doi: 10.3724/sp.j.1001.2012.04150.
- [66] B. Boser, I. Guyon, and V. Vapnik, "A Training Algorithm for Optimal Margin Classifier ChaLearn Looking At People: First Impressions View project Smart Dust View project A Training Algorithm for Optimal Margin Classifiers," 1996, doi: 10.1145/130385.130401.
- [67] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Neural Information Processing Systems*, Cambridge MA: MIT Press, 1997.
- [68] S. Paul, C. Boutsidis, M. Magdon-Ismail, and P. Drineas, "Random Projections for Linear Support Vector Machines," *ACM Transactions on Knowledge Discovery from Data*, vol. 8, no. 4, pp. 1–25, Oct. 2014, doi: 10.1145/2641760.
- [69] J. Smola and B. Scholkopf, "A tutorial on support vector regression," in *Statistics and Computing*, 2004, pp. 199–222.
- [70] Naiyang Deng, Yingjie Tian, and Chunhua Zhang, *Support vector machines : optimization based theory, algorithms, and extensions*. Boca Raton: Crc Press, 2013.
- [71] R. Ahmad and H. Jamaluddin, "Radial Basis Function (RBF) for Non-Linear Dynamic System Identification," *Jurnal Teknologi*, vol. 36, no. 1, Jun. 2002, doi: 10.11113/jt.v36.561.
- [72] G. Verma and H. Verma, "Predicting Breast Cancer using Linear Kernel Support Vector Machine," *SSRN Electronic Journal*, 2019, doi: 10.2139/ssrn.3350254.

- [73] J.-W. TAO and S.-T. WANG, “Kernel Support Vector Machine for Domain Adaptation,” *Acta Automatica Sinica*, vol. 38, no. 5, pp. 797–811, Dec. 2012, doi: 10.3724/sp.j.1004.2012.00797.
- [74] V. Van Belle, B. Van Calster, S. Van Huffel, J. A. K. Suykens, and P. Lisboa, “Explaining Support Vector Machines: A Color Based Nomogram,” *PLOS ONE*, vol. 11, no. 10, p. e0164568, Oct. 2016, doi: 10.1371/journal.pone.0164568.
- [75] J. D’Souza, “A Quick Guide to Boosting in ML,” *Medium*, Mar. 21, 2018. <https://medium.com/greyatom/a-quick-guide-to-boosting-in-ml-acf7c1585cb5>.
- [76] B. Boehmke, “Chapter 12 Gradient Boosting | Hands-On Machine Learning with R,” *Github.io*, Oct. 19, 2019. <https://bradleyboehmke.github.io/HOML/gbm.html> (accessed Nov. 17, 2019).
- [77] “Gradient Boosting Machines · UC Business Analytics R Programming Guide,” *Github.io*, 2017. http://uc-r.github.io/gbm_regression.
- [78] G. Tseng, “Gradient Boosting and XGBoost,” *Medium*, Nov. 29, 2018. <https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5>.
- [79] “Hybrid Extreme Gradient Boosting Models To Impute The Missing Data In Precipitation Records,” *Informatics and Applications*, Sep. 2019, doi: 10.14357/19922264190306.
- [80] Baris Mustafa Kazar and Mete Celik, *Spatial AutoRegression (SAR) model : parameter estimation techniques*. New York: Springer, 2012.
- [81] Y. Bengio, “Gradient-Based Optimization of Hyperparameters,” *Neural Computation*, vol. 12, no. 8, pp. 1889–1900, Aug. 2000, doi: 10.1162/089976600300015187.
- [82] M. Kim, “Supervised learning-based DDoS attacks detection: Tuning hyperparameters,” *ETRI Journal*, vol. 41, no. 5, pp. 560–573, Sep. 2019, doi: 10.4218/etrij.2019-0156.
- [83] Y. Zebin and Z. Aijun, “Hyperparameter tuning methods in automated machine learning,” *SCIENTIA SINICA Mathematica*, vol. 50, no. 5, p. 695, May 2020, doi: 10.1360/n012019-00092.
- [84] PyTorch, “Accelerate your Hyperparameter Optimization with PyTorch’s Ecosystem Tools,” *Medium*, Sep. 14, 2020. <https://medium.com/pytorch/accelerate-your-hyperparameter-optimization-with-pytorchs-ecosystem-tools-bc17001b9a49> (accessed Dec. 11, 2020).
- [85] “Overview of hyperparameter tuning | AI Platform Training,” *Google Cloud*. <https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview>.

- [86] Vadim Ermolayev, *Information and Communication Technologies in Education, Research, and Industrial Applications : 15th International Conference, ICTERI 2019, Kherson, Ukraine, June 12-15, 2019, revised selected papers*. Cham Springer, 2020.
- [87] J. Bergstra and Y. Bengio, "Random Search for hyper-parameter Optimization," *Journal of Machine Learning Research*, pp. 281–305, Feb. 2012.
- [88] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.
- [89] K. Maladkar, "Why Is Random Search Better Than Grid Search For Machine Learning," *Analytics India Magazine*, Jun. 14, 2018. <https://analyticsindiamag.com/why-is-random-search-better-than-grid-search-for-machine-learning/#:~:text=Random%20search%20is%20a%20technique> (accessed Dec. 11, 2020).
- [90] N. Krivulin, D. Guster, and C. Hall, "Parallel Implementation of a Random Search procedure: an Experimental Study," *5th WSEAS International Conference on Simulation, Modeling and Optimization (SMO'05)*, Aug. 2005.
- [91] A. A. Zhiglavskii and J. Pinter, *Theory of global random search*. Dordrecht Netherlands ; Boston: Kluwer Academic Publishers, 1991.
- [92] M. Stelmach and M. Chlebus, "Novel Multilayer Stacking Framework with Weighted Ensemble Approach for Multiclass Credit Scoring Problem Application," Faculty of Economic Sciences, University of Warsaw, 2020. Accessed: Sep. 2020. [Online]. Available: https://www.researchgate.net/publication/340924091_NOVEL_MULTILAYER_STACKING_FRAMEWORK_WITH_WEIGHTED_ENSEMBLE_APPROACH_FOR_MULTICLASS_CREDIT_SCORING_PROBLEM_APPLICATION_Novel_multilayer_stacking_framework_with_weighted_ensemble_approach_for_multiclass.
- [93] "ML-Ensemble — mlens 0.1.6 documentation," *mlens.readthedocs.io*. <https://mlens.readthedocs.io/en/0.1.x/> (accessed Dec. 12, 2020).
- [94] "Memory consumption — mlens 0.1.6 documentation," *mlens.readthedocs.io*. <https://mlens.readthedocs.io/en/0.1.x/memory/#memory> (accessed Dec. 12, 2020).
- [95] *Emerging technologies for information systems, computing, and management*. New York: Springer, 2013.
- [96] "Spyder — Anaconda documentation," *docs.anaconda.com*. <https://docs.anaconda.com/anaconda/user-guide/tasks/integration/spyder/#:~:text=Spyder%2C%20the%20Scientific%20Python%20Development> (accessed Nov. 20, 2020).

- [97] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011, doi: 10.1109/mcse.2011.37.
- [98] D. Y. Chen, *Pandas for everyone : Python data analysis*. Boston: Addison-Wesley, 2018.
- [99] K. Yuen, C. Yik, A. Chi, and Sandro Tosi, "Matplotlib for Python developers : effective techniques for data visualization with Python". Birmingham, Uk: Packt Publishing, 2018.
- [100] J. Avila, *Scikit-learn cookbook*. Birmingham: Packt Publishing, Limited, 2017.
- [101] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning, second edition : data mining, inference, and prediction". New York: Springer, 2009.
- [102] M. Harwell, "A Strategy for Using Bias and RMSE as Outcomes in Monte Carlo Studies in Statistics," *Journal of Modern Applied Statistical Methods*, vol. 17, no. 2, Mar. 2019, doi: 10.22237/jmasm/1551907966.
- [103] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, et al. "The accuracy of extrapolation (time series) methods: Results of a forecasting competition". *Journal of Forecasting*. 1982; 1(2):111±153. <https://doi.org/10.1002/for.3980010202>
- [104] C. Chen, J. Twycross, and J. M. Garibaldi, "A new accuracy measure based on bounded relative error for time series forecasting," *PLOS ONE*, vol. 12, no. 3, p. e0174202, Mar. 2017, doi: 10.1371/journal.pone.0174202.

Appendix

Below is the training which not just the super learner, but its constituent models and the hybrid metaheuristic regression model were fitted on.

| T | DO | S | pH | ORP | CR |
|-------|-------|-------|------|-------|--------|
| 25.9 | 6.71 | 30.1 | 5.1 | 378 | 16.4 |
| 29.35 | 6.09 | 29 | 6.3 | 400 | 16.9 |
| 27.9 | 6.18 | 31.5 | 7 | 363 | 15.57 |
| 24 | 7.95 | 30.2 | 8.1 | 324 | 13.65 |
| 28 | 5.05 | 31.4 | 9.2 | 240 | 13.24 |
| 27.87 | 6.55 | 31.68 | 7.2 | 356 | 14.06 |
| 28.27 | 6.98 | 28.2 | 6.6 | 384 | 15.47 |
| 29.37 | 6.82 | 30.12 | 6.2 | 414 | 17.11 |
| 24.27 | 0.8 | 32.56 | 8.1 | 171 | 3.61 |
| 27.45 | 2.6 | 35.37 | 7.96 | 287 | 7.94 |
| 27.48 | 5.9 | 32.39 | 7.83 | 331 | 10.578 |
| 28.75 | 6.8 | 32.22 | 8 | 340 | 11.43 |
| 28.52 | 8.4 | 32.1 | 8.01 | 345 | 12.52 |
| 28.45 | 9.9 | 31.95 | 7.93 | 309 | 22.64 |
| 24.73 | 6.06 | 17.33 | 7.88 | 321 | 11.446 |
| 24.51 | 7.02 | 32 | 8.16 | 308 | 12.553 |
| 23.65 | 6.51 | 41.34 | 7.67 | 245 | 8.402 |
| 16.74 | 7.11 | 33.55 | 8.25 | 178 | 10.85 |
| 21.11 | 6.03 | 33.44 | 8.03 | 295 | 11.448 |
| 25.57 | 6.7 | 32.19 | 8.09 | 325 | 11.872 |
| 31.16 | 4.38 | 33.21 | 7.94 | 242 | 8.924 |
| 25.62 | 34.89 | 5.32 | 7.9 | 385 | 15.966 |
| 24.95 | 16.29 | 6.8 | 7.82 | 341 | 12.12 |
| 24.5 | 18.37 | 5.31 | 7.93 | 302 | 12.07 |
| 25.59 | 21 | 7.04 | 7.95 | 244 | 11.4 |
| 26.11 | 34.84 | 2.82 | 7.8 | 335.2 | 11.288 |
| 24.96 | 40 | 6.32 | 8.08 | 254 | 9.3 |

| | | | | | |
|-------|-------|-------|------|-------|--------|
| 9.5 | 32.31 | 4.26 | 8.2 | 195 | 10.56 |
| 12.05 | 32.04 | 4.95 | 8.17 | 232 | 11.04 |
| 14.86 | 32.51 | 6.3 | 7.95 | 198 | 11.06 |
| 28.13 | 34.34 | 5.14 | 7.8 | 362.9 | 13.93 |
| 24.17 | 16.09 | 7.68 | 8.04 | 283.8 | 11.55 |
| 23.54 | 15.04 | 8.27 | 8.06 | 243.8 | 11.72 |
| 25.31 | 15.22 | 7.59 | 9.32 | 246.7 | 11.39 |
| 12.55 | 37.9 | 6.42 | 7.49 | 235.3 | 10.52 |
| 16.81 | 39.49 | 6.61 | 7.73 | 258.7 | 10.24 |
| 24.09 | 36.72 | 5.59 | 7.83 | 281.8 | 9.93 |
| 26.34 | 35.97 | 3.25 | 7.98 | 367.1 | 14.37 |
| 25.35 | 16.94 | 4.05 | 8 | 341.2 | 15.07 |
| 26.07 | 35.34 | 4.07 | 7.94 | 404 | 18.13 |
| 26.52 | 34.48 | 4.94 | 7.9 | 326.1 | 11.828 |
| 27.32 | 3.21 | 29.31 | 8.2 | 281 | 12.91 |
| 30.7 | 7.15 | 31.74 | 6.5 | 401 | 16.28 |
| 27.23 | 4.2 | 31.94 | 7.89 | 289 | 9.63 |
| 23.95 | 7.61 | 9.17 | 8.04 | 231 | 10.94 |
| 24.6 | 7.52 | 24.42 | 7.57 | 210 | 11.83 |