OPTIMAL CORROSION PREVENTION IN CRUDE OIL REFINERIES WITH SURROGATE MODELING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE OF
MIDDLE EAST TECHNICAL UNIVERSITY
BY

YÜCELEN BAHADIR YANDIK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2020

Approval of the thesis:

OPTIMIAL CORROSION PREVENTION IN CRUDE OIL REFINERIES WITH SURROGATE MODELING

Submitted by Yücelen Bahadır YANDIK in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics** _____

Prof. Dr. Sevgi Özkan Yıldırım
Head of Department, **Information Systems** _____

Assoc. Prof. Dr. Altan Koçyiğit
Supervisor, **Information Systems, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Pekin Erhan Eren
Information Systems, METU _____

Assoc. Prof. Dr. Altan Koçyiğit
Information Systems, METU _____

Assist. Prof. Dr. Bilgin Avenoğlu
Software Engineering, TED University _____

Assist. Prof. Dr. Serhat Peker
Management Information Systems, İzmir Bakırçay University _____

Assoc. Prof. Dr. Tuğba Taşkaya Temizel
Information Systems, METU _____

Date: ***24.09.2020***

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Yücelen Bahadır Yandık

Signature : _____

ABSTRACT

OPTIMAL CORROSION PREVENTION IN CRUDE OIL REFINERIES WITH SURROGATE MODELING

Yandık, Yücelen Bahadır

MSc., Department of Information Systems

Supervisor: Assoc. Prof. Dr. Altan Koçyiğit

September 2020, 93 pages

Energy demand in the world is increasing day by day, which makes energy markets extremely competitive. Crude oil refineries have to adapt to this competition like other players in the energy field. Corrosion is a common problem in crude oil refineries. Production may need to be stopped for maintenance to fix problems caused by corrosion. These stops cause businesses to miss their production target and lose their competitive advantage. Today, it is known that the salts in crude oil play an important role in corrosion. Even though there are several methods to remove salts in the crude oil, these methods are not perfect. For this reason, chemicals such as neutralizers, corrosion inhibitors, and caustic soda are used to reduce the corrosive effect of salts. However, using these chemicals in the inappropriate amounts can increase the corrosion or create a coke in the tanks that affects the production and needs to be cleaned. Therefore, the amount of chemicals to inject should be determined carefully. If this is performed manually by field operation staff based on heuristic approaches, it may lead to failures. Carrying out the decision-making process with a data-driven analytical method may provide more successful results and enable optimizations. Yet, developing analytical methods is seen as a costly and challenging way due to the complex nature of crude oil refineries. Using surrogate models instead of theoretical models can reduce costs and make the development process more manageable. To this end, we propose a method that optimizes the amount of chemicals added to prevent corrosion in crude oil refineries using an analytical method that relies on surrogate models. In order to evaluate the applicability and performance of the proposed method, an application was carried out in a refinery, and positive results were observed in the short term.

Keywords: Surrogate modeling, machine learning, optimization, crude oil distillation, corrosion

ÖZ

HAM PETROL RAFİNERİLERİNDE VEKİL MODEL KULLANILARAK ENİYİLENMİŞ AŞINMA ENGELLEME

Yandık, Yücelen Bahadır
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Doç. Dr. Altan Koçyiğit

Eylül 2020, 93 sayfa

Dünya'daki enerji talebinin gün geçtikçe artmaktadır ve bu durum enerji piyasalarını aşırı rekabetçi bir hale getirmektedir. Ham petrol rafinerileri de enerji alanındaki diğer oyuncular gibi bu rekabete uyum sağlamalıdır. Ham petrol rafinerilerinde aşınma sıkça rastlanan bir problemdir. Aşınmadan kaynaklanan sorunların giderilmesi için üretimin durması gerekebilmektedir. Bu duruşlar işletmelerin üretim hedeflerini tutturamamasına ve rekabet avantajlarını kaybetmelerine yol açabilmektedir. Günümüzde ham petrolün içerisinde bulunan tuzların aşınmada önemli rol oynadığı bilinmektedir. Ham petrolü içerisindeki tuzlardan arındırmak için bazı yöntemler olsa da kullanılan yöntemler mükemmel değildir. Bu nedenle, tuzların aşındırıcı etkisini azaltmak için nötrleştiriciler, aşınma engelleyiciler ve kostik soda gibi kimyasallar kullanılmaktadır. Fakat bu kimyasalların uygun olmayan miktarlarda kullanılması aşınmayı arttırabilmekte ya da tankların içerisinde üretimi etkileyen ve temizlenmesi gereken kok oluşturabilmektedir. Bu yüzden eklenecek kimyasalların miktarlarının dikkatlice belirlenmesi gereklidir. Eğer bu işlem sezgisel yöntemler temel alınarak saha personeli tarafından elle yapılırsa hatalara yol açabilir. Karar verme işleminin veri güdümlü analitik bir yöntemle yapılması daha başarılı sonuçlar alınmasını ve eniyileme yapılabilmesini sağlayabilir. Fakat ham petrol rafinerilerinin karmaşık doğası nedeniyle analitik yöntemlerin geliştirmesi masraflı ve zor bir yol olarak görülmektedir. Analitik yöntemlerin geliştirilmesinde teorik modeller yerine vekil modeller kullanılması masrafları azaltabilir ve geliştirme sürecini daha kolaylaştırabilir. Bu amaçla, ham petrol rafinerilerinde aşınmayı önleme amacıyla eklenecek kimyasalların miktarlarını eniyileyen vekil modellere dayalı olarak geliştirilmiş bir analitik yöntem önermekteyiz. Önerilen yöntemin uygulanabilirliğini ve başarımını değerlendirmek amacıyla bir rafineride uygulama gerçekleştirilmiştir ve kısa vadede olumlu sonuçlar gözlemlenmiştir.

Anahtar Sözcükler: Vekil modelleme, makine öğrenmesi, eniyileme, ham petrolün damıtılması, aşınma

To my parents,

Seniha and Nizamettin YANDIK

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere appreciation to my supervisor, Assoc. Prof. Dr. Altan Koçyiğit. He has given me warm supports, priceless guidance, generous advice, continuous support, and shown great patience through emails and phone calls than I ever would expect from an advisor.

I would like to thank Assoc. Prof. Dr. Pekin Erhan Eren for his suggestions, supports, encouragement, and for sharing his experiences and knowledge through this research.

I would like to thank Assist. Prof. Dr. Bilgin Avenoğlu, Assist. Prof. Dr. Serhat Peker, Assoc. Prof. Dr. Tuğba Taşkaya Temizel for reviewing my work.

I would like to thank my mother, Seniha Yandık, and my father, Nizamettin Yandık, for their guidance and advice throughout my life.

Special thanks go to my colleagues Mert Onuralp Gökalp and Kerem Kayabay, for their guidance and support since the beginning of this research.

I also would like to thank my precious friends and previous colleagues Okan Bilge Özdemir and Sibel Gülnar, for their friendship, support, and valuable guidance throughout life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ARIMA	Autoregressive Integrated Moving Average
BI&A	Business Intelligence and Analytics
CSV	Comma Separated Values
DSS	Decision Support System
EM	Expectation Maximization
ERL	Eastern Refinery Limited
IoT	Internet of Things
KNN	K-Nearest Neighbour
LASSO	Least Absolute Shrinkage and Selection Operator
ML	Machine Learning
OLS	Ordinary Least Squares
PCA	Principle Component Analysis
PPM	Parts Per Million
RBF	Radial Basis Function
RF	Random Forest
RMSE	Root Mean Squared Error
RRSE	Root Relative Squared Error
RSE	Relative Squared Error
SOM	Self Organizing Maps
SVM	Support Vector Machine
TSVM	Transductive Support Vector Machines
VLSI	Very Large Scale Integration
VPPM	Volumetric Parts Per Million

CHAPTER 1

INTRODUCTION

The processes and operations that companies perform have been getting more complex with improved technology. As businesses embrace newer methods to deal with the increasing complexity, companies understand the importance of using data to gain a competitive advantage. In a relevant study, 70% of senior executives stated that conducting data-oriented analytics is very or extremely important for their businesses to sustain competitive advantage [1]. Accordingly, companies are more focused on becoming data-driven to gain a competitive advantage in the market by using information technologies and IoT devices in the era of big data [2].

IoT devices and big data analytics play a crucial role in building data-driven companies. A recent study conducted by Statista states that there were 22 billion IoT devices connected to the Internet in 2018, and this number is expected to increase to 38.6 billion in 2025 and 50 billion in 2030 [3]. This large number of connected devices have the potential to generate an enormous amount of data. Even though there are many big data tools to process the data generated by these devices, extracting useful information by using available data is not an easy task. Because of that, businesses need to invest in business intelligence and analytics tools, which rely on big data technologies. Chen et al. [4] state that Business Intelligence & Analytics (BI&A) tools showed significant development in the past two decades, and business analytics became one of the four major technology trends in the 2010s. According to Chen et al. [4], five domains take advantage of BI&A tools to be data-driven. One of these domains is science and technology. However, in this domain, the development of a BI&A tool, considering the scale and complexity of the processes and the amount of data, is a time-consuming task that generally does not pay-off.

Optimization of processes is vital for companies to maintain a competitive advantage. This is justified by the increasing number of studies in the area in recent years. Such studies are targeting businesses operating in different areas such as textile [5], food processing [6], welding [7], and so on. Besides, there are studies focused on the optimization of more general resources, such as improving labor productivity [8] and increasing stakeholder satisfaction [9]. However, businesses are having trouble to optimize their complicated processes. They make some simplifying assumptions that lead them to ignore some subtle points in their operations to miss optimal solutions. At this point, surrogate modeling offers more straightforward solutions instead of explicitly representing complex relations in such applications. Surrogate modeling uses a simplified model that serves as a proxy to describe events that are too complicated to model, or the outcome of them cannot be measured directly. The successful use of surrogate models helps to save both computational time and resources in general. Bhosekar and Ierapetritou [10] state that surrogate modeling has gained popularity over the past three decades, which can be considered as proof of surrogate modeling is advantageous for businesses.

The use of surrogate modeling instead of making simplifying assumptions may enable companies not to ignore certain aspects of their processes. However, executives are acting cautiously to surrogate modeling despite its advantages. There are two main

reasons for this. The first one is that surrogate modeling is still an unfamiliar method in the industry. The second one is that surrogate modeling has not proven its competence yet. As a result, executives do not want to use their companies as a test object ambitiously, relying on a new and unproven technology [11].

In this study, we focused on the optimization of corrosion control and prevention in chemical plants. Slavcheva et al. [12] state that the reasons behind the corrosion in chemical plants are known. However, modeling the reactions that create a corrosive environment is not an easy task due to the complicated nature of the problem. Therefore, using surrogate modeling helps to reduce the complexity of the problem and eases the modeling of the corrosive environment inside the tanks. Such an approach to the problem is helpful for us to use the surrogates obtained in surrogate-based optimization to minimize the costs of chemicals used against corrosion problems.

In this thesis, we investigated the processes of a crude oil refinery located in Turkey. According to our observations, we decided to focus on improving the chemical injection process, which is commonly performed to prevent corrosion in crude oil refineries. We observed that the field operations staff of the company decide the amount of chemicals to inject manually by considering the specifications of the crude oil to be refined and the success of the recent process progress. However, this decision process causes several problems. First of all, they can easily miss a critical point that affects the process. As a result, they end up either injecting more chemicals than needed, which causes an unwanted accumulation of chemicals in the tanks that requires maintenance to clean or injecting fewer chemicals than needed, which causes faulty production and faster rate of corrosion in the production tanks. Even if the effect of one of the injected chemicals is insufficient, other chemicals are capable of compensating for the inadequacy of that chemical. Therefore, even if the process is progressing successfully, the mixture of the chemicals may not be optimum from the purchase cost or maintenance cost points of view. Lastly, we observed that newer employees of the company do not know the process with every aspect. For this reason, an extra workload of experienced employees becomes necessary.

In this study, we developed a software that helps to optimize the chemical injection process. Then, we tested and validated the applicability of the software in an operational crude oil refinery. The software we develop consists of two components: machine learning and optimization. The trained machine learning models are used as the constraints of the optimization problem, which aims to find optimal mixture of chemicals to prevent corrosion in crude oil refineries. In the development of the machine learning component, we followed the methodology suggested by Ge et al. [13]. In the optimization part, we employed two well-known globally convergent optimization problem solution methods, namely, genetic algorithm and simulated annealing. Although the scope of this study is limited to the optimization of the chemical injection process, our methodology used in the study can be easily adapted to the optimization of similar processes.

This thesis consists of five chapters. After this introductory chapter, Chapter 2 aims to give background information regarding machine learning, optimization, and surrogate modeling and the tools utilized in this thesis study. Chapter 3 introduces related work in the area of optimization in chemical plants and preventing corrosion in crude oil refineries. Chapter 4 presents our approach and the pertinent studies which we have

conducted in the domain of corrosion control for our partner in the crude oil refinery business. In Chapter 5, discussion about the proposed approach, findings of the study, our contributions, and possible future research areas are provided.

CHAPTER 2

BACKGROUND

This chapter provides an overview of the background information required to grasp the study performed in the context of the thesis. This background information consists of four headlines: machine learning, surrogate modeling, optimization, and the tools used in this thesis study. In section 2.1, machine learning is reviewed. In section 2.2, surrogate modeling is investigated. The purpose of using surrogate modeling; its use in real life are explained. In section 2.3, optimization is described. Besides, the relation between machine learning, surrogate modeling, and optimization is investigated. In section 2.4, tools used in the works in the content of the study are explained.

2.1. Machine Learning

Machine learning is a subdomain of artificial intelligence (AI). AI is a field of computer science based on imitating the signs of intelligence displayed by living things on machines. Copeland [14] states that constructing complex machines that resemble the same characteristics of intelligence of human beings with every aspect is the dream of AI pioneers from 1956. Yet, so far, this dream is partially achieved with limited success. Hall et al. [15] defined machine learning as a branch of artificial intelligence that focuses on constructing systems that require minimum intervention from humans in order to learn from data and make accurate predictions.

Holdaway and Ball [16] categorized the machine learning algorithms under three headlines: supervised, unsupervised, and semi-supervised. Each of these also has several sub-categories. This classification is delineated in Figure 1.

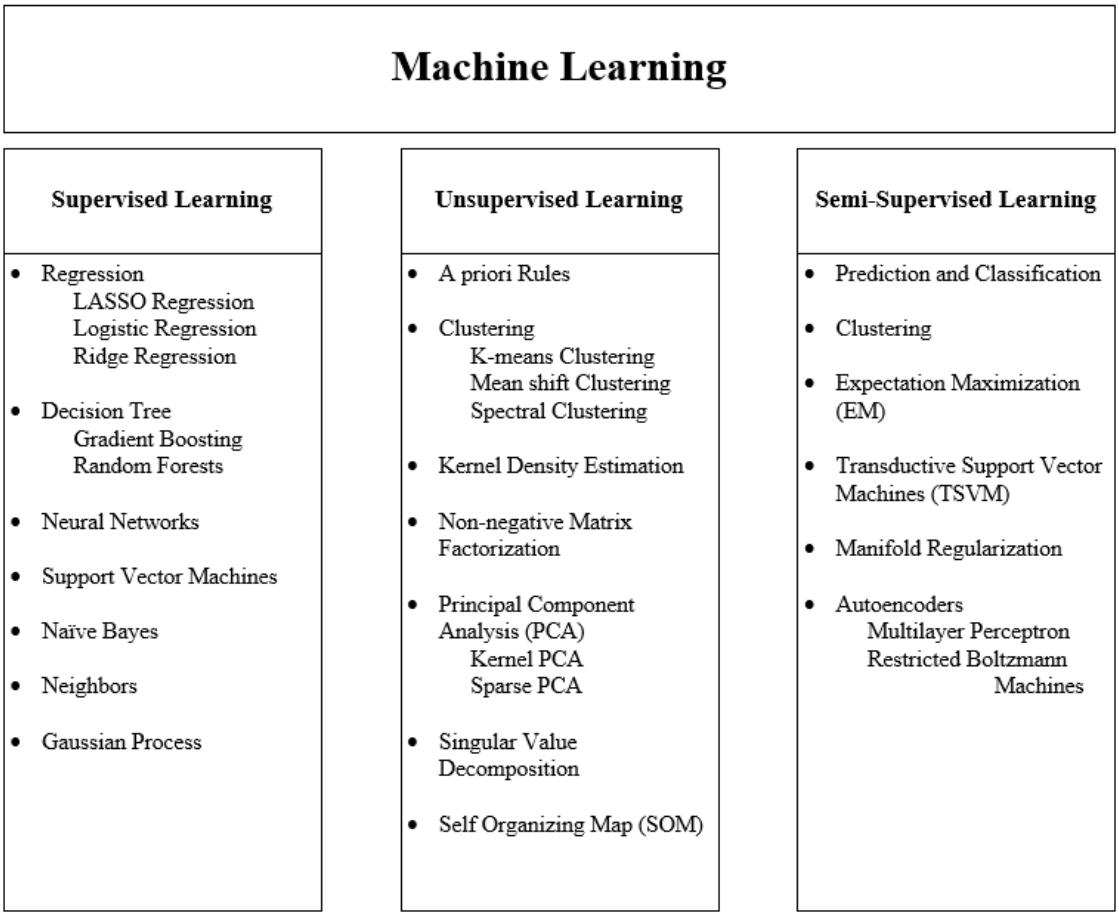

Figure 1: Detailed Machine Learning Taxonomy with the most widely used ML Algorithms (adapted from [16])

2.1.1. Supervised Learning

Supervised learning corresponds to a type of machine learning method that utilizes training data with each input has an associated output [17]. Several machine learning algorithms can be categorized under supervised learning [16]. In the content of this study, we are going to review linear (together with lasso and ridge) regressions, support vector machines (SVM), K-nearest neighbors (KNN), and random forests (RF) since they have widespread use in the literature.

2.1.1.1. Linear Regression

Linear regression is a straightforward approach to statistical learning [18]. It models the linear relationship between a given set of independent variables and a dependent variable. The linear regression method used in this study is the ordinary least squares (OLS) linear regression. OLS selects the parameters of a linear function of an input variable set by the principle of least squares. The least-squares method aims to minimize the sum of the squares of differences between the predicted output variable and the observed output variable. The linear regression model is called simple linear regression if the number of independent variables is equal to one. If the number of independent variables is more than one, it is called multiple linear regression [19]. The cost function for an OLS linear model is given in the following equation:

$$\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}\left(y_i - \sum_{j=0}^{M} w_j x_{ij}\right)^2 \quad (1)$$

Where the variables indicate:

- i : index of the samples,
- N : total number of samples,
- y_i : i^{th} element of the real values,
- $\hat{y}_i$: value of the i^{th} predicted element by the regression model,
- j : index number of features,
- M : total number of features,
- w_j : j^{th} element of the regression coefficient,
- x_{ij} : j^{th} element of i^{th} element of the input vector,

2.1.1.2. Lasso Regression

The linear regression models are subject to overfitting in many cases. The regularization process can be used to prevent overfitting. Lasso regression (least absolute shrinkage and selection operator) is a type of regularization in regression analysis models. It is able to perform L1 norm regularization, which is performed by adding a penalty term to the loss function. The penalty term is the absolute value of the magnitude of the coefficient [20]. Specifically, Lasso is also able to perform feature selection. The reduced set of features may improve the comprehension of models and provide higher accuracy. The cost function of Lasso regression is given in the following equation:

$$\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}\left(y_i - \sum_{j=0}^{M} w_j x_{ij}\right)^2 + \lambda \sum_{j=0}^{M} |w_j| \quad (2)$$

The equation is an extended version of the OLS linear model, which includes the penalty term to the OLS linear model. The function and, therefore, the variables are the same with the OLS linear model except for the penalty term. The only different variable in the penalty term is the regularization parameter, which indicates:

- λ : penalty term that helps to regularize the coefficients

2.1.1.3. Ridge Regression

Ridge regression is another method for regularization in regression analysis models. Ridge regression also uses a penalty term in the cost function to prevent overfitting, as Lasso regression does. It performs L2 norm regularization differently than Lasso regression. That is, the squared magnitude of coefficients is included instead of the absolute value of coefficient as penalty terms. The cost function of Ridge regression is given in the following equation:

$$\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}\left(y_i - \sum_{j=0}^{M} w_j x_{ij}\right)^2 + \lambda \sum_{j=0}^{M} w_j^2 \quad (3)$$

The equation is an extended version of the OLS linear model, which includes the penalty term to the OLS linear model, like the Lasso regression. The function and, therefore, the variables are the same with the OLS linear model except for the penalty term. The only different variable in the penalty term is the regularization parameter, which indicates:

- λ : penalty term that helps to regularize the coefficients

2.1.1.4. Support Vector Machine (SVM)

Support vector machines (SVMs) are supervised learning models. An SVM can be used for classification, regression, and other learning tasks [21]. SVM-regression is used for regression analysis, and SVM-classification is used for classification. SVMs are based on separating hyperplanes. The optimal separating hyperplane maximizes the distance of the closest point between two hyperplanes [22]. SVMs can utilize kernels concerning the optimal separating hyperplanes such as linear, polynomial, and radial basis function (RBF) [23]. These kernels are illustrated in Figure 2, Figure 3, and Figure 4.

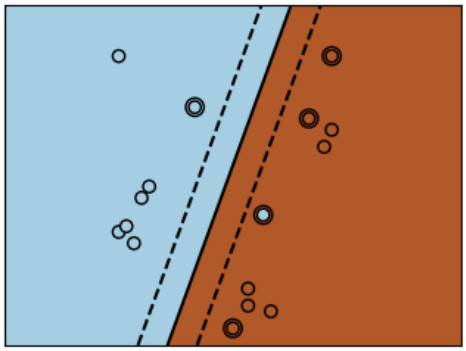

Figure 2: Linear SVM Kernel (adapted from [23])

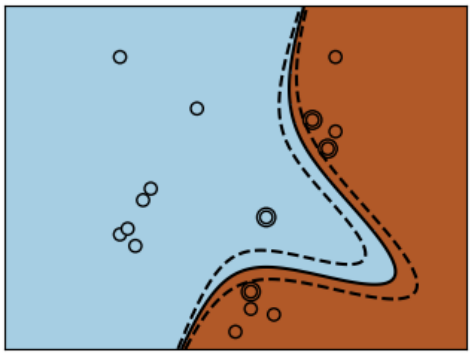

Figure 3: Polynomial SVM Kernel (adapted from [23])

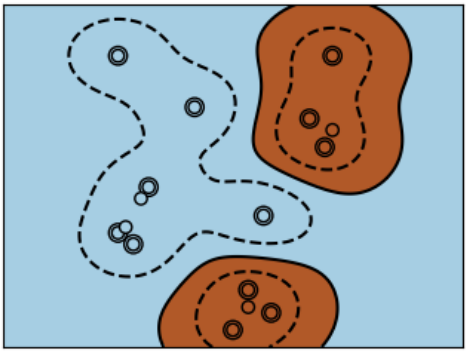

Figure 4: Radial Basis Function (RBF) SVM Kernel (adapted from [23])

2.1.1.5. K-Nearest Neighbors (KNN)

K-nearest neighbors algorithm is a nonparametric estimation technique [24]. It can be used for both regression and classification purposes. The algorithm defines the group or value of the new input with respect to the group or value of the k nearest element. Euclidian distance is the most commonly used distance metric while identifying the k nearest elements.

An example case of KNN is as follows: A classification problem where there are only two classes exists represented in Figure 5. When an unclassified element comes to the classifier, it identifies the k nearest neighbors (3 nearest neighbors for our example). It defines the group of the new element with respect to the types of neighbors. This process is represented in Figure 6. In the example, the unclassified new member belongs to class 2 since two of its nearest neighbors belong to class 2.

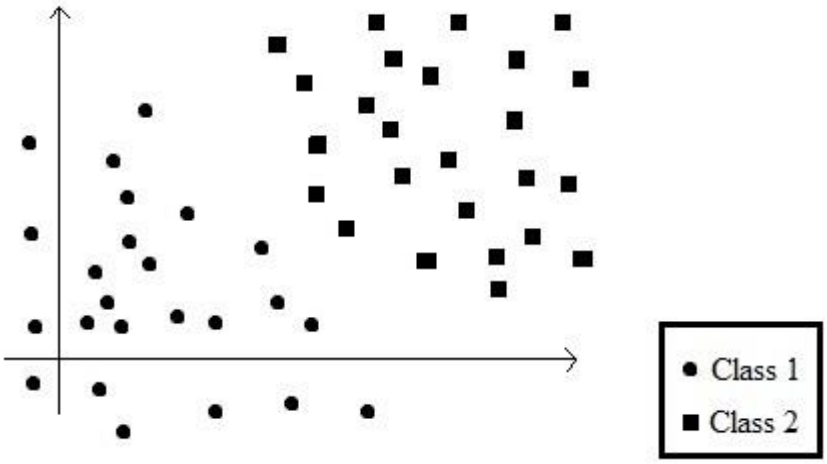

Figure 5: A classification example with two classes

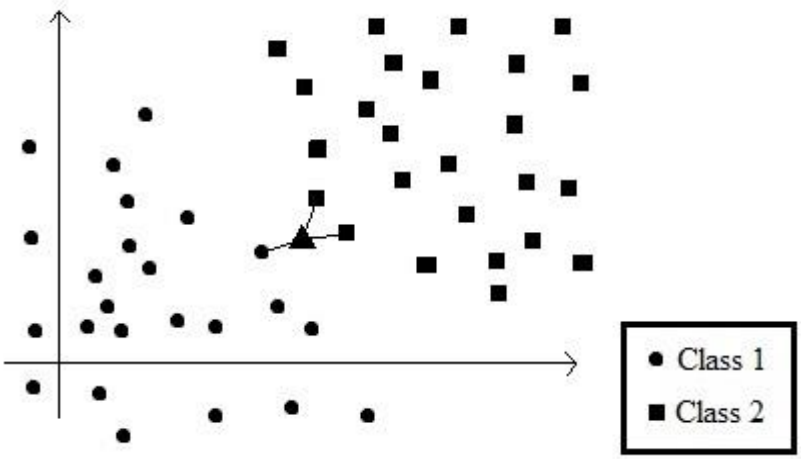

Figure 6: Identification of k nearest neighbors of the unclassified new member

2.1.1.6. Random Forests (RF)

Random forests are classified as an ensemble learning method. Ensemble learning means using multiple learning algorithms to provide better results. Classification, regression, and other learning tasks can be performed by using random forests. Multiple decision trees are trained at training time to construct a forest. Mode of the individual trees used for classification or regression purposes [25].

2.1.2. Machine Learning Process

Ge et al. [13] define the methodology of data mining and analytics as follows:

- *Data preparation:* It is the initial step of machine learning model development. This step aims to have an overview of the process data and define the most suitable samples. The main tasks of this step are obtaining the data from the database, examining the structure of the dataset obtained, and selecting the relevant data by using methods that include variable selection.
- *Data pre-processing:* The purpose of this step to improve the quality of the dataset prepared in the previous step by performing data pre-processing and some appropriate data transformations. This step consists of four sub-steps. Firstly, when working with temporal data, the time axis of the dataset needs to be checked and corrected if any inconsistent record exists. Secondly, outliers and corrupt records should be removed from the dataset since they can affect the performance of the ML model. Thirdly, missing values should be detached or filled by various estimation techniques. Lastly, process variables should be scaled if they have scale differences between themselves.
- *Model selection, training, and performance evaluation:* The first objective to achieve after the preparation of the dataset is to the choice of a proper ML model. The model parameters are estimated after the model selection by implementing the algorithm on the training dataset. Performance evaluation of the model is necessary to perform before using the model by using model validation and performance evaluation methods such as cross-validation, model stability analysis, model robust stability, etc.

- *Data mining and analytics:* The ML model is ready to use in data mining and analytics, such as data clustering, data visualization, dimensionality reduction, trend analysis, and so on, after the ML model has been trained and validated.

Overview of the data mining and analytics methodology Ga et al. suggested is illustrated in Figure 7.

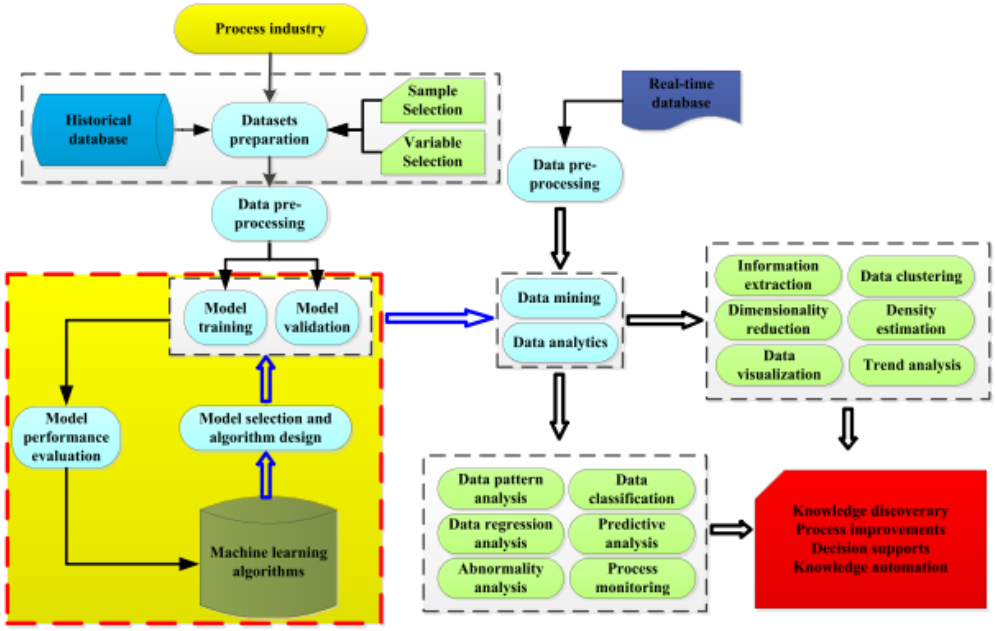

Figure 7: Overview of the data mining and analytics methodology (adapted from [13])

2.2. Surrogate Modeling

Kim and Boukouvala [26] state that the solution for most of the real-life engineering problems requires complex computer simulations. These simulations may help to obtain more accurate and precise information regarding complex, multi-scale, multiphase, or distributed systems. It is essential to model the problems during the simulations. On the other hand, it is not always possible to express these models explicitly by using physics-based algebraic equations. In cases where the models cannot be expressed explicitly, the environment to be simulated will be ambiguous. To solve this ambiguity, developing successful and accurate models required. However, developing these models requires complicated and costly calculations. As a result, obtaining an analytical form of the objective function and derivatives of the objective function becomes difficult [10]. Vu et al. [27] define such functions as black-box.

Surrogate modeling is an engineering technique that simplifies modeling of events or processes that cannot be directly measured or quantified, with the help of proxy models. It is a method that can be used to simplify modeling black-boxes. Developed surrogates enable modeling of events or processes to a certain extent and make them observable. As a result, they become eligible to make measurements. It often relies on other tools like machine learning to approximate the relationships between the variables by using the measured data. McBride and Sundmacher [28] suggest using surrogate models instead of dealing with the black-boxes and model them promises remarkable success.

The main challenge of surrogate modeling is to make the surrogates as accurate as possible while using the minimum amount of computational resources. Having such surrogates requires to have a good sample selection, proper model, and model parameters selection, evaluation of the accuracy. The accuracy of the model increases as the number of samples increases in general. After the model is produced, its accuracy is tested. Nevertheless, the model may not always be very successful. Sources of the error are tried to be removed in unsuccessful cases. The source of the error might be noise on the data or erroneous measurements in the dataset. Reproducing the surrogate model using a different model can also be used as a solution.

There are several occasions where using a surrogate model is advantageous compared to the modeling. First of all, we can obtain disastrous results in modeling by ignoring a significant factor while making approximations and assumptions used. On the other hand, all factors affecting the results can be reflected up to some point if the data is taken correctly from the right places. Secondly, if companies have a sufficient amount of collected data or installed data infrastructure, the surrogate modeling process can start immediately. On the contrary, in cases where the whole system is modeled, the system must stop to take precise measurements. Also, complex operations and calculations are required for modeling after measurements are taken. Lastly, if there are changes in system operation, surrogate modeling can quickly adapt to the new situation if the data infrastructure is ready. However, the process starts from scratch in the modeling of the whole system.

Several studies use surrogate modeling, which cover different specialties. Alley [29] stated that placing the purifiers in the process of installing underground water cleaning systems is a complicated problem. Therefore, Alley simplified the structure using the surrogate model instead of solving the complex differential equations used to locate the purifiers. In another previously mentioned study, Sacks et al. [30] suggest that to reduce the complexity of computer simulations, they use surrogate modeling instead of solving differential equations. These simulations cover a wide range from electronic circuits to estimation of fire direction, even a ring oscillator. In another study, Fialho et al. [31] suggest using surrogate modeling to design buildings in the Smart Building project of Microsoft Research. They tried to minimize energy consumption, construction costs, and so on. As can be seen, surrogate modeling has extensive use in many different areas.

2.3. Optimization

Optimization can be defined as finding the best solution among the many feasible solutions that we have [32]. The best solution could suggest things such as the cost that minimizes a process or the maximum efficiency of a system [32]. Optimization problems generally consist of two main elements: an objective function and constraints. However, constraints are not a must element and do not exist in all optimization problems.

According to NEOS [33], the taxonomy of optimization problems varies concerning several aspects of the issues. First, they are classified according to whether the variables are discrete or continuous. Secondly, they are classified according to whether the variables are subject to certain constraints. Thirdly, they are classified according to how many objective functions the problem is trying to optimize at the same time.

Finally, they are classified as deterministic or stochastic, depending on whether the data used in the issue is fully known.

According to Hussain et al. [34], optimization has a widespread application with a lot of known methods categorized under conventional, non-conventional, and hybrid approaches. Even though traditional methods provide accurate results, they are more time consuming than non-conventional ones. Besides, non-conventional methods give better solutions than conventional ones in general. As a result, using a non-conventional optimization method is more advantageous. Genetic algorithms and simulated annealing are two well-known non-conventional optimization techniques.

2.3.1. Genetic Algorithm

The genetic algorithm (GA) is a solver for both optimization and search problems. GAs are a subset of a computation method called evolutionary computation. It provides sufficiently reasonable solutions for problems in relatively faster computation time than the traditional methods. Laws of natural selection and genetics are the bases of GA [35]. GA consists of three operations, namely selection, genetic operation, and replacement. The cycle of GA is illustrated in Figure 8.

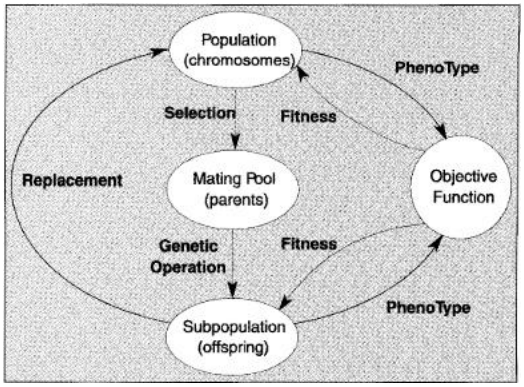

Figure 8: The cycle of GA (adapted from [35])

GAs have a population or a pool that consists of possible solutions to the given problem. Members of this solution pool go through processes like recombination and mutation, selection of best parents, and producing new offspring. This process is repeated over and over again for numerous generations. Each member of the pool has a fitness value that indicates how the members are capable of solving the problem. The members have more chance to reproduce new children if their fitness value is higher. Therefore, the survival of the fittest theory of Darwin holds. In this way, the members of the population keep evolving to become better solver for the problem over generations.

GAs have some advantages and disadvantages with respect to their competitors. Firstly, they are advantageous since they do not require any derivative information. Secondly, they are more efficient and faster than traditional methods. Thirdly, they are capable of providing several reasonable solutions instead of one solution. Fourthly, they offer an answer all the time, which gets better over time. Lastly, they are useful when the search space and the number of parameters to optimize are large. However, they have some disadvantages, too. First, they are not suitable for solving every problem. Second, the calculation of the fitness function might be costlier than solving

the problem with traditional methods if the problem is simple. Third, new solutions are only better than the previous ones, which requires to have proper stop criteria. Otherwise, it might not be converging to an optimal solution.

2.3.2. Simulated Annealing

Simulated annealing (SA) is a probabilistic optimization method. It approximates the global optimum solution of a given function all the time. SA is more useful when finding an approximate result to a global minimum than finding the exact global minimum in a definite amount of time is more critical. SA is generally used in solving discrete but enormous problems such as the traveling salesman problem or VLSI routing.

Its name comes from the metallurgy term annealing. Annealing is a heat treatment process that changes the physical and chemical properties of a material to increase its flexibility and reduce its hardness, which makes it more workable. During the annealing process, the material is heated until the temperature is above its recrystallization point. Later, the temperature is maintained for some time. Finally, it is left to cool slowly in time. The material recreates the crystalline lattices in the cooling process while it tries to reach minimal energy between the atoms or molecules.

SA resembles the physical annealing process by which molten materials cool to crystalline lattices of minimal energy. Instead of minimizing the energy in the annealing process, the objective function is optimized in SA. As a result, it approximates the global optimum value of a given objective function [36].

In SA, we randomly select a solution first. Then, we check if there is any neighbor solution that provides a better solution than our solution. If neighbors do not have a better solution, we review two things: how bad neighboring solutions are from our solution and how high the current temperature of the system is. If the temperature of the system is higher, the solution is more likely to be better. Simply, we try to find solutions that provide smaller energy changes with high temperatures. In the SA algorithm, first, we set an initial temperature and define a random initial solution. Later on, we try to resemble the cooling process. In this step, we decrease the temperature gradually until we find a suitable solution, which is the stop criterion of the process. When the algorithm stops, we make a small change in the obtained solution to make the modified solution to pretend as a neighbor solution. Later on, we decide to go on with the neighbor solution or not by computing the energy function. In the end, we decrease the temperature and redo the process.

2.4. Tools

2.4.1. Programming Language

Python and R are the most common programming languages for data science applications, according to a blog written by Ridpath published in Toward Data Science [37]. Python promoted to be used as a programming language of the pipeline since it is widely used by the community and widespread support for tools.

2.4.1.1. Programming Language Libraries

During the development of the software and the validation studies, we used Python as the primary programming language. Python provides many libraries for data science and many other applications, which makes it the most used programming language in data science, as stated previously. The libraries used in this study are briefly explained in this section.

Numpy is the fundamental package for scientific computing for Python Programming Language. It eases to use N-dimensional array objects, complicated functions. It is also useful for linear algebra and Fourier transforms computations [38].

Pandas is started to be developed in 2008. It enables quick and efficient manipulation of data frame objects in Python. It can also read and write CSV, text, and Microsoft Excel files. It makes data alignment, handling of missing data, manipulating data, and reshaping it easier. It also eases to work with time-series data [39].

Scikit-learn is a machine learning library for Python which supports supervised and unsupervised learning. Besides, it provides tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities. It is an entirely open-source library [40].

SciPy is an open-source software collection for scientific computing in Python. It is built on libraries like Numpy, Pandas, Scikit-learn, and Matplotlib. It gathers several numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics, and so on [41].

MlxTend is an open-source data science library for Python, which provides useful tools for the day-to-day data science tasks [42].

CHAPTER 3

PROBLEM STATEMENT AND SOLUTION APPROACH

The use of decision support systems (DSSs) and BI&A tools in chemical plants are widespread. These tools are used for several different purposes, such as planning human resources [43], deciding the amount of raw material to be purchased and processed [44], observing, controlling, and making the necessary diagnoses in production lines and distillation units [45] in chemical plants.

3.1. Crude Oil and Distillation

Crude oil is a complex chemical compound of several hydrocarbons with small amounts of sulfur, nitrogen, salt, and metals such as nickel, vanadium, and copper [46]. These hydrocarbons are gasoline, diesel, jet fuel, heating oil, fuel oil, lubricants, asphalt, coke, wax, and chemical feedstocks [46]. These substances differ from each other with molecule types. Such substances constitute the products of the refineries, and the products refined from crude oil should comply with the standards set by the industry.

Crude oil refineries separate these hydrocarbons from each other to produce end products. In addition, they are able to eliminate the parts of the hydrocarbons, which does not meet the industry standards. However, this is not the limit of what crude oil refineries can do. Sulfur and nitrogen have a detrimental impact on the environment. Besides, salts inside the crude oil are harmful to the production environment. Crude oil refineries can also separate these harmful and unwanted compounds, too [46].

3.2. Corrosion in Crude Oil Refineries

One of the significant challenges chemical plants faced is corrosion. Especially, crude oil refineries are prone to corrosion in the production tanks. Slavcheva et al. [12] stated that even though we know the reasons behind the corrosion in chemical plants, we cannot offer the right solution to this problem since we cannot thoroughly model the corrosive behaviors of reactions due to the complexity of the problem. Bhowmik et al. [47] identified the primary source of corrosion in chemical plants as hydrochloric acid, hydrogen sulfide, organic and inorganic chlorine, and sulfur compounds. Besides, these compounds lead to the production of end products that do not meet the specifications, as well as creating a corrosion-prone environment [48]. Therefore, it is crucial to separate unwanted and harmful substances such as sulfur, nitrogen, and salts in crude oil.

Crude oil contains inorganic salts that contain chlorine (Cl), such as NaCl, $CaCl_2$, and $MgCl_2$. Factors like high temperature, water inside crude oil, water added in desalination, and pressure accelerate hydrolysis and cause the generation of hydrochloric acid (HCl), which quickens corrosion [47]. Chemical reactions regarding these processes are given below.

$$NaCl + H_2O \rightarrow NaOH + HCl$$
$$CaCl_2 + H_2O \rightarrow Ca(OH)_2 + 2HCl$$
$$MgCl_2 + H_2O \rightarrow Mg(OH)_2 + 2HCl$$

3.2.1. Corrosion Prevention in Crude Oil Refineries

There are several methods to prevent corrosion in the crude oil refineries. The measures taken to solve the corrosion problem in crude oil refineries can be listed as desalination, injecting chemicals, and choosing the right building material [47]. Desalination is performed in desalters, and it is the most effective way of reducing corrosion in chemical plants. It is because the salts inside crude oil are the main reason for corrosion. However, desalination does not entirely eliminate those salts.

Chemical injection methods are used to reduce the corrosive effects of salt remained after the desalination process. There are four different types of chemical injection to prohibit corrosion in chemical plants: demulsifiers, caustic soda, neutralizer, and corrosion inhibitor injections [47]. Even though these injected chemicals are beneficial to reduce the effects of corrosive elements, they have several adverse effects, such as leaving residuals behind or damaging copper alloys in case of excessive use.

In the desalination process, crude oil is mixed with water so that the salts inside the crude oil dissolved in the water. Demulsifiers are applied before the desalination process and assist the desalination operation. Demulsifiers help to separate salty water from desalted crude oil. They also assist residence time, improve solids removal, minimize water carryover or oil under carrying, and reduce the emulsion layer. Recommended dose limit of demulsifiers is between a few ppm to about 100 ppm. For example, a crude oil refinery located in Bangladesh, Eastern Refinery Limited (ERL), uses 6 kg of separol per 1000 metric tons of crude oil [47].

Caustic soda is used to regulate the amount of chlorine in the water. As a result, it reduces the formation of hydrochloric acid, which causes corrosion in chemical plants. Even though caustic soda regulates the chlorine to prevent the formation of hydrochloric acid most of the time, it is not wholly successful. The establishment of hydrochloric acid is observed even if it is in small amounts. Caustic soda also causes some accumulation of unwanted leftovers from chemical reactions in the tanks. In general, the maximum amount of injected caustic soda is limited to 2.27 kg per 100 barrels of crude oil. ERL uses caustic soda dose 5-10 ppm of crude oil which is equal to 0.66-0.32 kg of caustic soda per 1000 barrels of crude oil [47].

Neutralizer or neutralizer amine (NH_3) is used to neutralize the effects of the hydrochloric acid formed despite the use of caustic soda. However, extensive use of neutralizer amine causes severe corrosion because they react with copper-base alloys if pH is above 8.0 [47] and create useless residual products in the tanks, which requires cleaning. Chemical reactions regarding this process are given below.

$$NH_3 + HCl \rightarrow NH_4Cl$$

Lastly, a corrosion inhibitor is used to eliminate the adverse effects of any acidic environment left after these three processes. Corrosion inhibitor creates a thin film layer around the tanks. This layer prevents hydrochloric acid from damaging the tanks and formation of corrosion in the tanks. In general, corrosion inhibitor injection rate is

3-5 vppm. However, the rate may be temporarily increased to 12 vppm to help to form the protective layer during the start or unit upsets. In ERL, the dose of corrosion inhibitor is between 0.6 to 1.7 kg of total gasoline per 1000 barrels of crude oil [47].

Professionals use different measurement techniques to observe the success of corrosion preventive methods employed in the process. Reports suggest that regular laboratory analysis of iron and chlorine amount in boot water and pH of boot water is one of the indicators of corrosion rate [49]. These analyses are usually performed on a shift or daily basis. Under normal operating conditions, the results of these analyses should meet values between a range given in Table 1.

Table 1: The allowed range of laboratory analyses of boot water to measure the corrosion rate

Measurement Type	Minimum Value	Maximum Value
pH of Boot Water	5.5	6.5
Iron in Boot Water (ppm)	0.2	1
Chlorine in Boot Water (ppm)	10	40

3.2.2. Optimal Chemical Injection for Corrosion Prevention

Although there is extensive knowledge of the use and effects of chemicals and how to measure the success of the chemical injection process, it is not possible to propose a definitive conclusion about the optimum mixture of chemicals to inject. This uncertainty occurs because of two reasons. First, we have a range of values in which the process is assumed to be acceptable. Second, several different mixtures of chemicals may confine the process in the acceptable range. These two reasons guide us to approach the chemical injection process as an optimization problem. Optimization of the process has the potential to provide reduced costs, increased durability of production tanks, and better quality of end products.

The objective function of the chemical injection process optimization can be set as minimizing the costs. These costs should include both the purchasing costs of chemicals and the maintenance costs of the production tanks. Determination of the purchasing costs is easy since the unit prices are readily available, and the total cost is proportional to the unit price and amount of chemicals used.

However, the decision of maintenance costs is not as straightforward as determining the purchasing costs. It has several reasons behind it. First, the injected chemicals cause an unwanted accumulation in some cases. However, there is no well known method to predict the amount of accumulation. To model the accumulation, frequent and long term measurements of accumulated leftovers is needed and this is not usually done in the regular operation of the refinery. In addition, source of accumulated leftovers may affect the process quality in different ways. For example, one chemical leftover may require heating the furnaces more than the other chemicals. Second, the maintenance processes require stopping the process for some time. During the maintenance, company loses money for not producing. Therefore, refinery cannot estimate how much it will lose money since the duration of maintenance is not known

and prices of end products sare volatile. In addition, during the maintenance process, there is a need for maintenance personnel, special equipment and replacement materials. These are also varying and it is not easy to estimate the costs of these elements. By considering these difficulties on the determination of maintenance costs, in this study, we are going to focus on purchasing costs only. However, our objective function can easily be extended to incorporate maintenance costs. In the end, our solution will be shaped around the following optimization problem formulation:

$$minimize: (Cost_{ci} * \ \ Amount_{ci}) + (Cost_{na} * Amount_{na}) + (Cost_{cs} * Amount_{cs})$$

$$Subject\ to:$$

$$5.5 \leq pH\ of\ Boot\ Water \leq 6.5$$

$$0.2 \leq Iron\ in\ Boot\ Water \leq 1$$

$$10 \ \leq Chlorine\ in\ Boot\ Water \leq 40$$

where:

- $Cost_{ci}$: unit cost of corrosion inhibitor
- $Amount_{ci}$: amount of injected corrosion inhibitor
- $Cost_{na}$: unit cost of neutralizer amine
- $Amount_{na}$: amount of injected neutralizer amine
- $Cost_{cs}$: unit cost of caustic soda
- $Amount_{cs}$: amount of injected caustic soda

Identifying the constraints is not as simple as the determination of costs. The lower and upper limits of constraints are known and given in Table 1. Modeling the entire system can help to determine the factors affecting these constraints. These factors include specifications of crude oil, momentary process parameters, and amount of injected chemicals. However, the complex feedback mechanism of the crude oil refineries and complex interactions between these factors makes it hard to model the entire system. Besides, all parts of the system need to be investigated element-wise to find their transfer functions. As a result, these make the modeling of the entire system too complex and costly for the crude oil refinery companies.

Instead of modeling the entire system, we suggest using surrogate modeling to solve the optimization of the chemical injection process. Surrogate modeling basically corresponds to modeling a system by using measured data. In our case, surrogate modeling is used to determine constraints that cannot be derived theoretically. Instead, with the help of training machine learning models with historical data, we obtain the expressions needed in the constraints. That is, we apply machine learning to approximate the following functions:

$$pH\ of\ Boot\ Water = \ p\left(x_{petroleum}\ , Amount_{ci}\ , Amount_{na}\ , Amount_{cs}\right)$$

$$Iron\ in\ Boot\ Water = \ f\left(x_{petroleum}\ , Amount_{ci}\ , Amount_{na}\ , Amount_{cs}\right)$$

$$Chlorine\ in\ Boot\ Water = c\left(x_{petroleum}\ , Amount_{ci}\ , Amount_{na}\ , Amount_{cs}\right)$$

where:

- $Amount_{ci}$: amount of injected corrosion inhibitor
- $Amount_{na}$: amount of injected neutralizer amine
- $Amount_{cs}$: amount of injected caustic soda
- $x_{petroleum}$: set of specifications of crude oil – available as sensor readings

In this study, we aim to develop a software that can be categorized as BI&A software. The software optimizes the chemical injection process to prevent corrosion in crude oil refineries by using surrogate modeling. This tool is composed of two parts: surrogate modeling and optimization. We are going to follow the machine learning methodology given in section 2.1.2 during the surrogate modeling part. Because it is necessary to train high accuracy machine learning models to represent the system with every aspect, and having a well-defined methodology might be a key to it. In the second part, we are going to use a genetic algorithm and simulating annealing techniques for solving the optimization part that comprises the surrogate models generated in the first part. In the following chapter, we are going to demonstrate the application of the proposed method in a field study and present the results.

CHAPTER 4

FIELD APPLICATION AND RESULTS

In this study, we worked with a crude oil refinery operating in Turkey. The operators in the company are currently determining an appropriate mixture of chemicals to confine the process in an acceptable operating region by means of heuristic approaches. They basically employ an educated guess method based on their observations to determine the appropriate amounts of chemicals to inject. However, this application may make the company miss three advantages that using an analytical approach may provide. These advantages are to reduce the adverse effects coming from the following situations:

- The process does not always end with success if heuristic approaches are used and may lead to the use of an insufficient amount of chemicals. This results in the production of end products that do not meet industry standards. In addition, corrosion in production tanks occurs faster.
- There may be situations where the inexperienced employees of the company do not have enough insight to make these educated guesses. This may create an extra workload for experienced employees.
- Over-injected chemicals due to the unsuccessful estimates lead to an accumulation in production tanks. These accumulated chemical leftovers reduce the quality of the products and need to be cleaned. This cleaning may require maintenance that necessitates stopping production temporarily.

In this study, we focused on developing a new approach to determine the amount of chemicals to be injected into the system to prevent corrosion in crude oil refineries. In contrast to the currently employed method, our approach offers cost optimization as well as reducing the adverse effects while injecting the chemicals. To this end, we formulate an optimization problem by using surrogate modeling. A set of ML models form the basis of the optimization constraints.

Our study consists of two steps. The first step is to train ML models with high accuracy. These ML models are used as surrogates that model certain aspects of the process. Therefore, having more accurate ML models means analyzing the system more reliably. We employ the ML methodology proposed by Ge et al. [13] with minor adaptations. The second step is to formulate an optimization problem by involving these ML models to incorporate constraints on the solution. We solve the optimization problem by using the genetic algorithm and simulated annealing methods.

The training of ML models consists of several steps. The first step is to determine the processes to be modeled, the variables to be estimated, and the variables to be used in the estimations. The field operation staff needs to decide the amount of the injected chemicals and to determine the success of the injecting chemicals after the process is finished. Deciding the amount of injected chemicals requires three different ML models since there are three different chemicals that can be used. These models can be derived in terms of thirteen different features (listed in Section 4.1) obtained from laboratory and sensor measurements. Determining the success of chemical injection also requires three ML models since the success of the process is controlled with three

variables. These models can be formed in terms of the amount of chemicals used as well as the thirteen features (listed in Section 4.1) used in the first set of ML models since the success of the process is dependent on the amount of three different chemicals injected. Therefore, sixteen features are used in total while estimating the three variables that indicate the success of the process.

The dataset that is used in training the ML models is extracted from the process history database of the company. Pre-processing operations are applied to the dataset. ML models which are used as surrogates are trained by using the pre-processed dataset. Trained models are used both to assist the operators in the company to control the process and in the optimization process.

In the second step of the study, we formulate the optimization problem and solve it. The optimization problem formulation consists of the following steps.

- Determining the objective function
- Determining the constraints
- Solving the optimization problem by
    - Genetic algorithm
    - Simulated annealing

In this section, we explain the procedures given above and provide the results.

4.1. Machine Learning Processes of Our Suggested Methodology

In this study, we need to predict six features. Three of these features are the amount of chemicals to be injected. These variables are called “set variables.” Prediction of the set variables is useful to train inexperienced employees. Therefore, we generated prediction models for the set variables, which are the amount of the following chemicals:

- Corrosion inhibitor
- Neutralizer amine
- Caustic soda

The other three features are to check whether the chemical injection is successful or not. These variables are called “observed variables.” This naming convention is chosen because we are setting the amount of the first three variables and observe the other three to determine the result of the process.

The models trained to predict observed variables are used to determine the constraints of the optimization problem. Therefore, they help to reduce the adverse effects due to the amount of chemicals. Observed variables are as follows:

- pH of boot water
- Iron in boot water
- Chlorine in boot water

Six different ML algorithms have been tried to train these models, which are linear, lasso, ridge, random forest (RF) regressions, SVM-regression, and k-nearest neighbors (KNN) regression. The ML models for prediction of chemicals use the following thirteen sensor readings and laboratory measurements, which are called as input variables given below.

- Crude oil flow rate
- Crude oil API
- Desalter input temperature
- Desalter mixture valve 1
- Desalter mixture valve 2
- Flow rate of washing water
- Temperature of the top of the tower
- Reflux flow at the top of the tower
- Furnace entrance temperature
- Crude oil entrance salt
- Crude oil salt at the exit of desalter
- pH of wastewater from desalter
- Oil at sewer from desalter

The latter three ML models which predict pH, iron, and chlorine levels, use these thirteen variables given above together with the amounts of three chemicals injected. Therefore, these models use sixteen different features during training as input.

4.1.1. Data preparation

The company has a big data infrastructure that stores process data, such as sensor readings and laboratory analysis results. Sensor readings are taken periodically, and periods vary from one second to one minute. Also, product samples are taken and analyzed in the laboratory periodically at the start of each shift, and results are written into the database. These data have longer periods, which can go up to eight hours. The variables are stored in the database with measurement value and time. We merged the measurements in the same dataset with a single time value. Even though the dataset contains the time value, we are not going to mention them since we are not using them except data preprocessing. The dataset consists of records of operation between August 25th, 2016 and November 5th, 2019.

There are more than a thousand different data collection points in a plant. However, all of them are not directly affecting the chemical injection process. Because of that, there is a need for feature selection to determine the relevant features if the whole dataset available with no background information. In this study, relevant features are suggested by the domain experts such as the field operation staff of the company. The corresponding sensor readings and sampling intervals are provided in Table 2.

Table 2: Sampling rate of the relevant variables in the database

Variable Name	Sampling Rate
Crude Oil Flow Rate	1 sample/30 seconds
Crude Oil API	1 sample/30 seconds
Desalter Input Temperature	1 sample/1 minute
Desalter Mixture Valve 1	1 sample/1 minute
Desalter Mixture Valve 2	1 sample/1 minute
Washing Water Flow Rate	1 sample/1 minute
Tower Top Temperature	1 sample/30 seconds
Tower Top Reflux Flow	1 sample/30 seconds
Furnace Entrance Temperature	1 sample/1 minute
Crude Oil Entrance Salt	1 sample/1 minute
Crude Oil Salt at Exit of Desalter	1 sample/1 minute
pH of Waste Water from Desalter	1 sample/1 minute
Oil at Waste Water from Desalter	1 sample/1 minute
Corrosion Inhibitor Rate	1 sample/8 hours
Neutralizer Amine Rate	1 sample/8 hours
Caustic Soda Rate	1 sample/8 hours
pH of Boot Water	1 sample/8 hours
Chloride in Boot Water	1 sample/8 hours
Iron in Boot Water	1 sample/8 hours

The descriptive statistics of the dataset, such as the number of records, mean value, standard deviation, and maximum and minimum values before preprocessing, are given in Table 3 for input variables and in Table 4 for set and observed variables.

Table 3: Descriptive statistics for the unprocessed input variables

Variable Name	Number of Records	Mean	Standard Deviation	Max	Min	Median
Crude Oil Flow Rate	59856	94.24	31.66	137.15	-1.40	109.35
Crude Oil API	538	19.80	0.44	22.60	18.90	19.80
Desalter Input Temperature	59856	98.48	27.16	139.48	23.14	108.05
Desalter Mixture Valve 1	59856	0.53	0.24	1.79	-0.06	0.61
Desalter Mixture Valve 2	59856	0.55	0.25	0.88	-0.06	0.65
Washing Water Flow Rate	59856	6.20	2.92	16.43	-0.40	7.48
Tower Top Temperature	59856	116.44	33.31	190.42	25.34	129.85
Tower Top Reflux Flow	59856	29.73	12.51	40.71	-0.51	35.59
Furnace Entrance Temperature	59856	228.29	84.75	283.25	22.41	267.46
Crude Oil Entrance Salt	518	214.32	85.64	944.30	56.20	211.35
Crude Oil Salt at Exit of Desalter	500	9.15	10.3	191.50	5.30	7.90
pH of Waste Water from Desalter	502	6.54	0.23	8.00	5.80	6.50
Oil at Waste Water from Desalter	497	13.85	4.10	37.20	5.40	13.00

Table 4: Descriptive statistics for the set and observed variables

Variable Name	Number of Records	Mean	Standard Deviation	Max	Min	Median
Corrosion Inhibitor Rate	357	9.03	1.08	10.93	6.98	9.04
Neutralizer Amine Rate	357	13.94	1.74	18.61	11.09	13.75
Caustic Soda Rate	357	572.80	85.87	800.00	450.00	580.00
pH of Boot Water	616	5.69	0.26	6.60	3.80	5.70
Chloride in Boot Water	608	9.68	5.63	47.50	2.00	8.40
Iron in Boot Water	615	0.54	1.21	15.40	0.02	0.32

After investigating the refinery processes and the nature of the data available, we decided to work with data that correspond to five-minute intervals. The reason behind this is that there is not much difference between the two successive measurements in the variables in the short term. That is, even the sensor reading periods are small, the values do not change rapidly, and analysis carried out based on five-minute sampling intervals capture the process dynamics sufficiently.

4.1.2. Data pre-processing

Even though this study follows the methodology presented in section 2.1.2, some adaptations were necessary for the data pre-processing step of the method. In this section, applied pre-processing procedures and the adjustments performed are explained. After the preprocessing, we are left with a dataset consisting of 59856 records of 19 variables. Thirteen of these variables are input variables, three of these are set variables, and three of these are observed variables.

4.1.2.1. Resampling and Time Correction

The sampling periods of the features in the input dataset are different from each other. Thus, a frequency resampling process becomes necessary to set the period of each variable to five minutes. Even though the frequency resampling process is not given as a time axis correction method in the followed ML methodology, we need such a preprocessing in our case.

We apply upsampling on variables having a period higher than five minutes and downsampling on variables having a period smaller than five minutes in resampling. We employ forward filling and backfilling for upsampling. We complete the missing values of the variables with a period of eight-hour. First, we determine the measurement and analysis times. Afterward, we fill the missing values four hours ahead and four hours backward using the values at the determined times. Resampling also helps to complete all missing values and synchronize all timestamps in the dataset. The illustration of this process is given in Figure 9. In the figure, M_1 and M_2 are the actual values measured at the start of the shifts. We calculate the mean of the values

in a five-minute window if the periods of the variables are lower than five minutes. After the resampling, we are left with a dataset consisting of 59856 records of 19 variables.

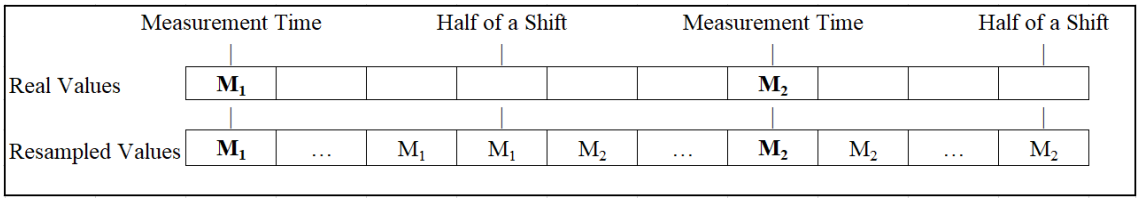

Figure 9: Resampling process

4.1.2.2. Outlier Elimination and Correction of Corrupt Records

The crude oil refinery plant where this study was carried out was not continuously operational. Nevertheless, the sensors located at different places in the plant keep reading the values. We identify these values by considering the times when measurements and analyzes were not performed. Later on, these records are deleted from the dataset. Besides, there is a timespan that the sensor reading of crude oil flow rate was broken. The sensor readings are negative at that period, which makes these readings incorrect. We also delete these values from the dataset.

In the second step, we can apply outlier elimination. At the beginning of this study, the records with values that are not in three standard deviations away from the mean of the corresponding feature are detected and removed from the dataset for outlier cleaning. However, in the later steps of this study, we discovered that the outlier deletion is reducing our success. In fact, these outliers represent rare events, and it is crucial to find rare events to increase success in this study. Therefore, we decided not to remove records containing outlier values.

4.1.2.3. Removal or Completion of Missing Values

We used resampling in the earlier steps of the data pre-processing. Therefore, all the missing values were completed by filling earlier. However, these filled values do not represent actual values. On top of that, two other reasons can negatively affect the results. The first one is since the crude oil is processed in a flow, a delay occurs between the sensor measurements itself and even between laboratory measurements and analysis. That is, crude oil, which starts to be processed at time t, passes through the first sensor at a time of uncertain t+x, and passes through the last sensor at the time of another unknown t+y. Therefore, it is not appropriate to use the values of the first and the latest sensor measured at the same time due to this time lag. This lag in the process is illustrated in Figure 10. The second one is that the results of laboratory analyses and measurements are written to the database manually at the time of the beginning of the shift. However, the collection time of the samples is different from the beginning of the shift. To neutralize these adverse effects, we included two additional steps at the end of the data pre-processing different than the ML methodology suggested, which are:

- Use of moving window to eliminate the harmful effects of resampling.
- Calculation of a polynomial features of the variables of the higher degree to find non-linear correlations and also interactions between features.

These steps are explained in the following sections.

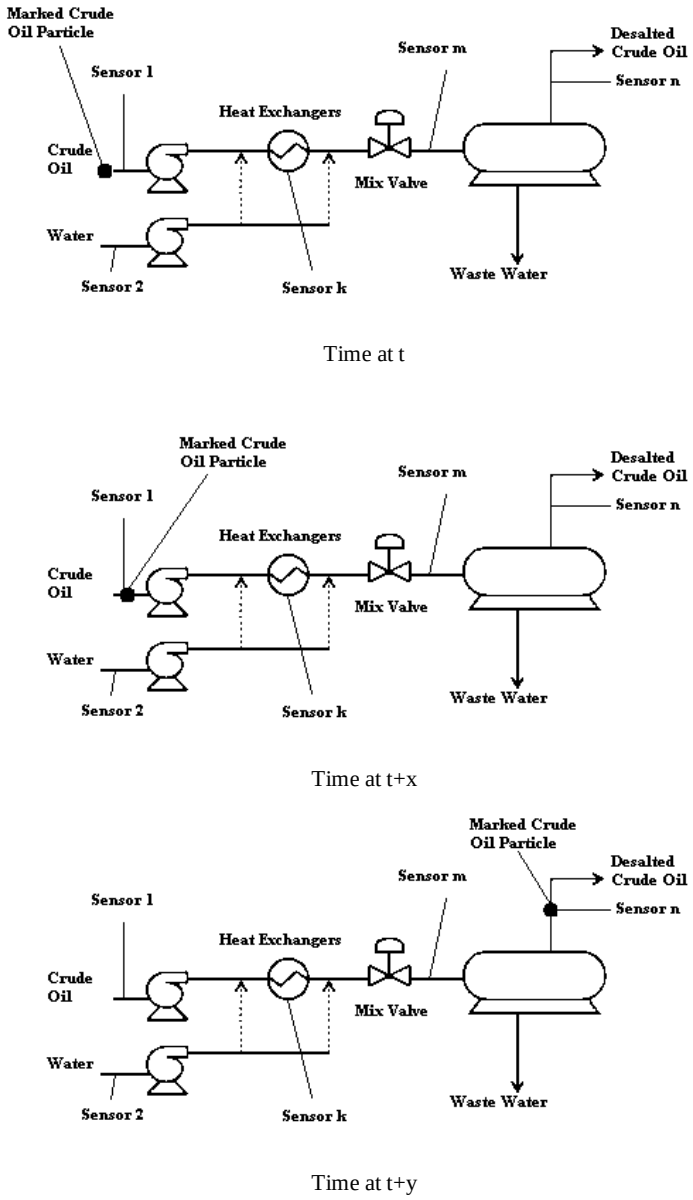

Figure 10: Time lag illustration of desalter

4.1.2.4. Scaling the Values of the Variables

Since the difference between the values of different variables is not excessively high, a scaling operation on the variables is not required.

4.1.2.5. Moving Window

A moving window is employed to reduce the harmful effects of resampling. The field operation staff of the oil refinery company provided information that the whole process takes place in one hour. Based on this information, we set the broadest range of our moving window as one hour. The limit of the end part of the one-hour window is adjusted to the point where the measurement is taken. Later, it starts to move forward with five-minute steps until the edge at the start comes to the point where the measurement is taken. After the first run was completed, the length of the window

shortened five minutes until the window length became five minutes. The windows moved forward each time. In each run, ML models are trained to detect the best model. During the training, each record inside the window is given as input to the ML models. The working principle of the rolling window method is illustrated in Figure 11. The most accurate ML models were found with the window length was fifty minutes and located between fifty-five minutes before and five minutes before the shift change. As a result, we are left with a dataset consisting of 7944 records of 19 variables after the moving window is applied.

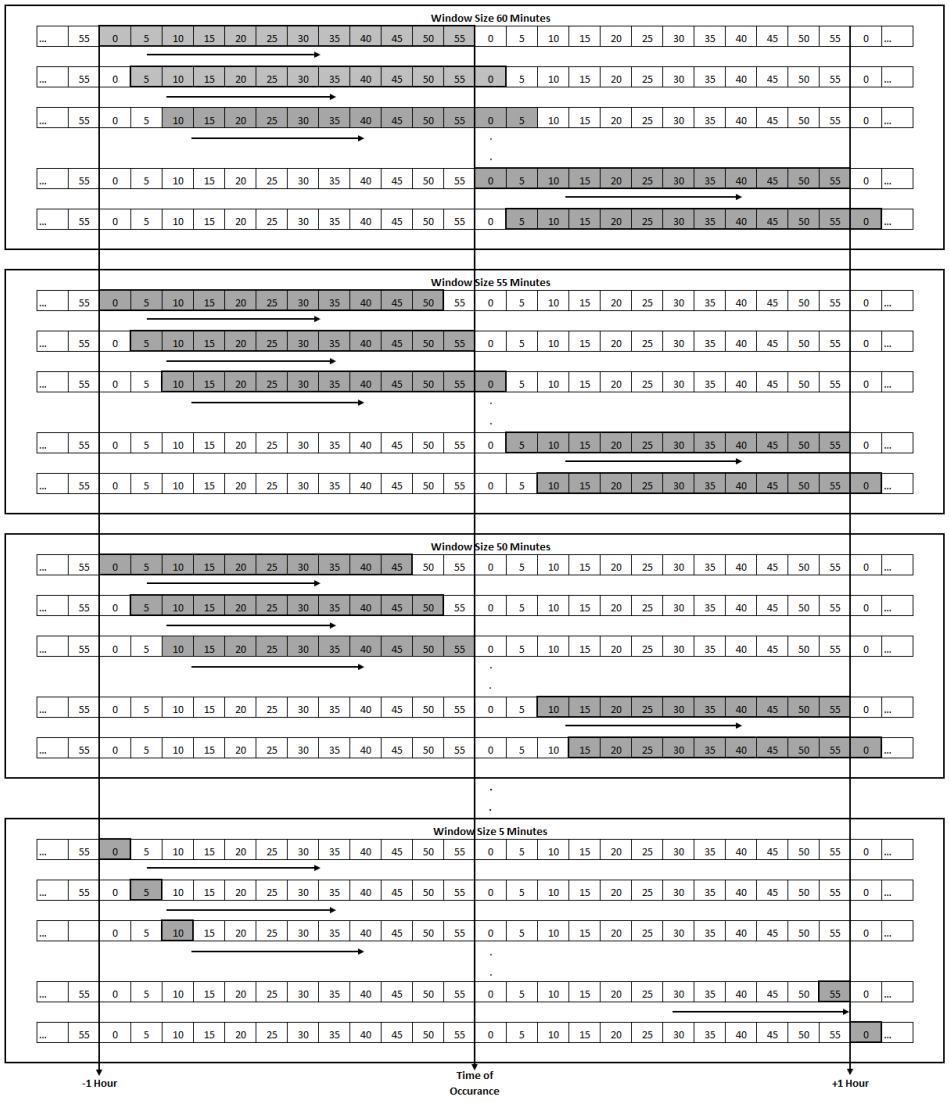

Figure 11: Moving window process

4.1.2.6. Polynomial Features

There are non-linear relations between variables, and input features may interact with each other due to the complex nature of the process. Therefore, we add new features by using polynomial features to extract existing non-linear relations and interactions. Polynomial feature calculation gives a zero power of all terms, which is a single one value, real values of features, interaction terms, and n^{th} power of features where n represents the degree of polynomial calculation. The polynomial expansion is performed by using the PolynomialFeatures method of the scikit-learn library. We find

the polynomial features of second-degree polynomials and interaction terms for linear, lasso, ridge, RF, KNN, and SVM-regression models.

After the polynomial expansion, the number of features used to train the ML models is 105 for the set variables and 153 for the prediction of the observed variables.

4.1.2.7. The Resulting Dataset after Preprocessing

After completing the data pre-processing techniques described in this section until now, we obtain a data set that we can train our ML models. The dataset consists of 7944 records and 19 variables. These variables, their types/use, mean value, standard deviation, and maximum and minimum values of these variables are given in Table 5 for the input variables and Table 6 for the set and observed variables.

Table 5: Descriptive statistics for the input variables after pre-processing

Variable Type	Variable Name	Mean	Standard Deviation	Max	Min	Median
Input Variables	Crude Oil Flow Rate	106.29	8.62	124.92	87.35	109.48
	Crude Oil API	19.68	0.39	20.80	18.90	19.70
	Desalter Input Temperature	109.99	3.18	120.74	102.94	109.93
	Desalter Mixture Valve 1	0.63	0.07	0.88	0.49	0.63
	Desalter Mixture Valve 2	0.66	0.07	0.85	-0.03	0.67
	Washing Water Flow Rate	7.37	0.87	8.21	-0.40	7.50
	Tower Top Temperature	130.25	2.77	162.14	121.79	130.19
	Tower Top Reflux Flow	34.76	3.62	40.12	21.42	35.80
	Furnace Entrance Temperature	264.53	6.20	272.62	248.45	267.30
	Crude Oil Entrance Salt	213.80	84.52	944.30	63.60	211.85
	Crude Oil Salt at Exit of Desalter	9.42	11.49	191.50	5.30	7.93
	pH of Waste Water from Desalter	6.55	0.24	8.00	5.80	6.50
	Oil at Waste Water from Desalter	14.34	4.16	37.20	5.40	13.40

Table 6: Descriptive statistics for the set and observed variables after pre-processing

Variable Type	Variable Name	Mean	Standard Deviation	Max	Min	Median
Set Variables	Corrosion Inhibitor Rate	9.00	1.07	10.93	6.98	8.98
	Neutralizer Rate	13.91	1.70	18.61	11.09	13.75
	Caustic Soda Rate	571.47	85.23	800.00	450.00	580.00
Observed Variables	pH of Boot Water	5.71	0.20	6.50	4.20	5.70
	Chloride in Boot Water	9.33	4.15	47.50	2.20	8.70
	Iron in Boot Water	0.38	0.33	5.70	0.03	0.31

The resulting dataset is randomly split into two datasets by taking 80% of the records for training and 20% of the records for testing purposes after the data pre-processing procedure is completed. As a result, we have a dataset with 6355 records for training and 1589 records for testing purposes. Besides, we carried out cross-validation on the training set in order to choose the best hyperparameters during training.

We also considered the train-test split with respect to the time axis. However, this did not provide better performance. Performances of trained models with the corresponding datasets are given in Appendices. In Appendix A, we provide the results for the estimators trained with earlier 80% of the records and tested by the last 20% of the records. In Appendix B, we provide the results for the models trained with the last 80% of the records and tested by the earliest 20% of the records. The poor performance provided by these models is possibly caused by the two main reasons.

First is the chemical amount determination strategy employed by the field operation staff. They started to use higher rates at a certain operating condition and reduce the amount of chemicals injected gradually in time to find the most appropriate operating point. When we split the dataset with respect to time by considering this, the trained models become biased due to the gradual decrease in amounts. The amount of injected chemicals are given as time series in Figure 12 which helps to visualize the gradual decrease. Amount of caustic soda injection is scaled by 1/100. In addition, Figure 13 provides the observed variables as time series which may help to investigate the variables in more detail. Iron amount in the boot water is scaled by 10 in the Figure 13. There is no data available due to the refinery stop which causes a gap in both figures.

Second, eight-hour period of measurements of set and observed variables is not frequent enough to represent the time-series relationship between the variables. Because of that, it seems more logical to approach to the data as it is containing process values instead of recurrent event values.

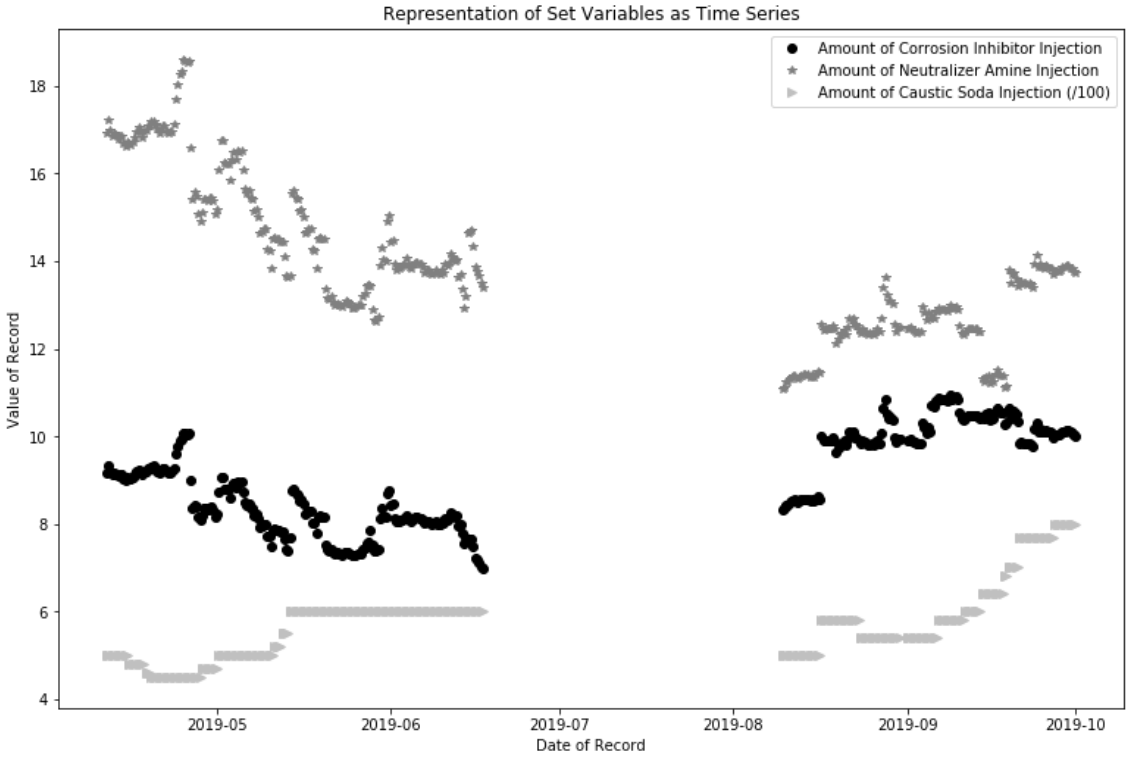

Figure 12: Representation of Amount of Injected Chemicals as Time Series

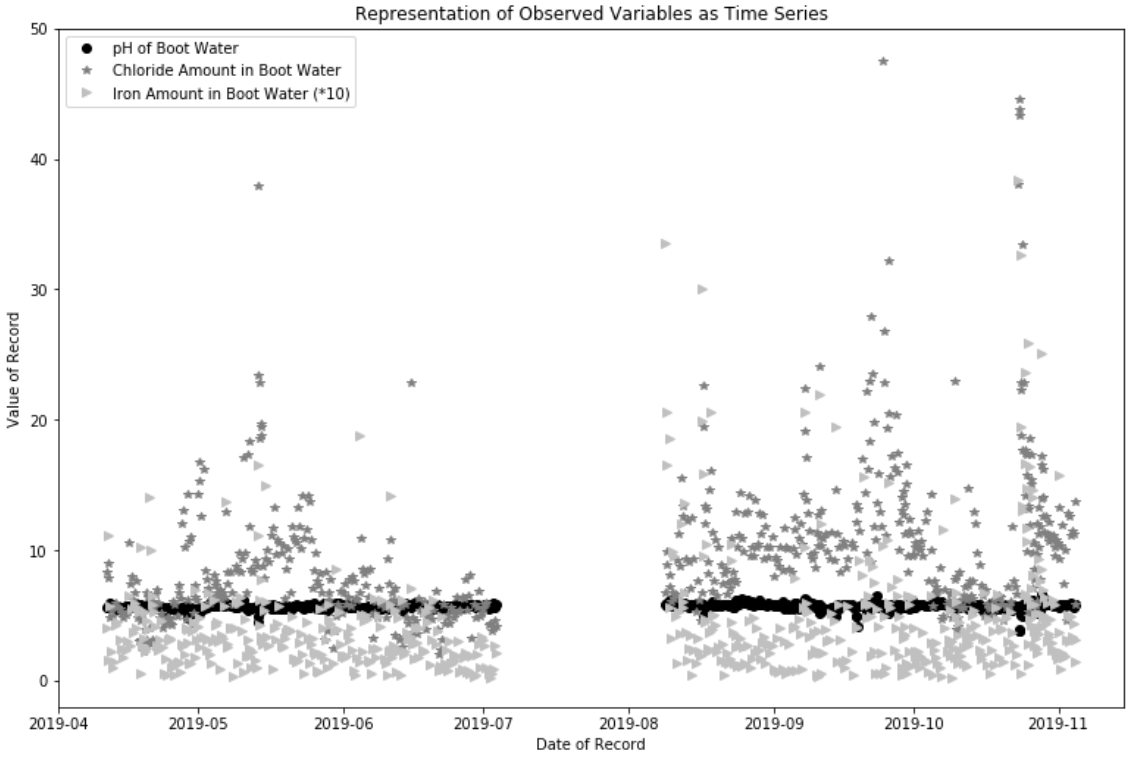

Figure 13: Representation of Observed Variables as Time Series

4.1.3. Model Selection, Training, and Performance Evaluation

In this section, we are going to fit ML models that predict six different variables. The first three variables are corrosion inhibitor, neutralizer amine, and caustic soda. The

models that are generated for these variables can be used to predict the amount of chemicals that might be injected by the domain experts if they stick to the currently employed strategy. That is, the ML models predicting these variables can be used to emulate the current approach taken by field operators for determining the amount of chemicals to inject. The other three variables to be predicted are pH of boot water, iron amount in boot water, and chlorine amount in boot water. These variables can be used to predict the performance of the chemical injection process. The ML models predicting these variables are going to be used to define the constraints of our optimization problem for minimization of cost.

There are points where different ML models are superior to each other. Therefore, a model selection was necessary to find the best fit ML model for a case-specific scenario. In this section, we present the training results of linear, lasso, ridge, random forest, KNN, and SVM regressions that can be used to predict six variables. These variables are corrosion inhibitor rate, neutralizer amine rate, caustic soda rate, pH of boot water, the iron amount in boot water, and the chlorine amount in boot water.

4.1.3.1. Model Selection and Training

We used linear, lasso, ridge, random forest (RF), KNN, and SVM regressions to find the best model. The models are trained by performing a hyperparameter tuning except for linear regression. In this section, we provide the performances on the training and test datasets of these models with the corresponding hyperparameters.

The first regression type is linear regression. There is no parameter to tune for linear regression, but tuning of the regularization parameter is necessary for lasso and ridge regressions. The regularization parameter represents the coefficient of L1 norm regularization term in lasso regression and L2 norm regularization term in ridge regression and will be represented with alpha (α). Tested regularization values are given in Table 7 for lasso regression and Table 8 for ridge regression.

Table 7: Tested regularization parameter (α) values for lasso regression during hyper-parameter tuning

Regression Type	**Regularization Parameter (α) Values**
Lasso Regression	0.00001, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20

Table 8: Tested regularization parameter (α) values for ridge regression during hyper-parameter tuning

Regression Type	**Regularization Parameter (α) Values**
Ridge Regression	0.00001, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20

Scikit-learn library of Python offers fourteen parameters to adjust for the RF regression model. Changing these parameter values may affect the performance of a model considerably. We choose six of the parameters which might affect the results the most

to be tested during hyperparameter tuning. These are the number of trees in the forest (n), maximum depth of the tree (d), the minimum number of samples to split a node (ms), the minimum number of samples required to be at a leaf node (minl), the maximum number of leaf nodes (maxl), and the minimum weighted fraction of the total of weights (mfw).

We create a grid that contains the values to be tested in hyperparameter tuning. The start values, the end values, steps between values, and the total number of values for each parameter are given in Table 6. The values start at the start value, increase with the step value until they reach the stop value. In the end, the total number of test values for a parameter represented at the '#values' column of Table 9. As a result, 2304 different parameters are tested for the hyperparameter tuning of RF regression.

Table 9: Tested values of parameters for random forest regression during hyper-parameter tuning

Parameter	Start	Stop	Step	# values
Number of trees in the forest (n)	25	200	25	8
Maximum depth of the tree (d)	2	8	2	4
Minimum number of samples to split a node (ms)	2	8	3	3
Minimum number of samples required to be at a leaf node (minl)	1	5	2	3
Maximum number of leaf nodes (maxl)	0	6	2	4
Minimum weighted fraction of the total of weights (mfw)	0	0.2	0.1	3

Scikit-learn library of Python offers seven parameters to adjust for the KNN regression model. We only set the number of neighbors for the KNN regression among these parameters since we believe it is the most important among them. Tested regularization parameter values for KNN regression are given in Table 10.

Table 10: Tested values of parameters for KNN regression during hyper-parameter tuning

Parameter	Values
Number of neighbors (n)	3,4,5

The SVM-Regression module of the scikit-learn library of Python has eleven parameters available for adjusting. We choose four of them to tune since their effect is more important than others. These parameters are kernel type of SVM, degree of polynomials, kernel coefficient (gamma), and regularization parameter (C). The degree of the polynomial parameter only exists if the kernel type is polynomial, and the kernel coefficient (gamma) parameter exists if the kernel types are RBF, polynomial or sigmoid. Regularization parameter (C) applies a squared L2 norm

penalty. Tests are performed by using linear, polynomial, and RBF kernels. Since each parameter is available for different types of kernels, the parameters are given in separate tables. Tested regularization parameter values for the linear kernel are given in Table 11. As a result, seven different tests are performed for the linear kernel. The tested degree of the polynomial, kernel coefficient, and regularization parameter values for the polynomial kernel is given in Table 12. As a result, 245 different tests are performed for the polynomial kernel. Tested kernel coefficient and regularization parameter values for RBF kernel are given in Table 13. As a result, forty-nine different tests are performed for the RBF kernel.

Table 11: Tested values of regularization parameter (C) values for SVM-regression with Linear Kernel during hyper-parameter tuning

Parameter	**Values**
Regularization Parameter (C)	0.001, 0.01, 0.1, 1

Table 12: Tested values of degree the of polynomial, kernel coefficient (gamma) and regularization parameter (C) values for SVM-regression with Polynomial Kernel during hyper-parameter tuning

Parameter	**Values**
Degree of polynomial	1, 2, 3, 4, 5
Kernel Coefficient (gamma)	‘scale’
Regularization Parameter (C)	0.001, 0.01, 0.1, 1

Table 13: Tested values of kernel coefficient (gamma) and regularization parameter (C) values for SVM-regression with RBF Kernel during hyper-parameter tuning

Parameter	**Values**
Kernel Coefficient (gamma)	‘scale’
Regularization Parameter (C)	0.001, 0.01, 0.1, 1

We performed model fitting and hyperparameter tuning by using cross-validation. LassoCV and RidgeCV of the scikit-learn library were used for lasso and ridge regressions. On the other hand, an exhaustive grid search with a cross-validation method of the scikit-learn library is used for RF regression and SVM-regression. This method is called as GridSearchCV. LassoCV, RidgeCV, and GridSearchCV use 5-fold cross-validation. 5-fold cross-validation means the training dataset is divided into five datasets, as illustrated in Figure 14. Four of the components are used for training, while the other is used for validation. Cross-validation is repeated five times to perform validation with each component. The average accuracy of these five models provides accuracy for us to decide on the best hyperparameters. In the end, we used the parameter set, which provides the highest accuracy.

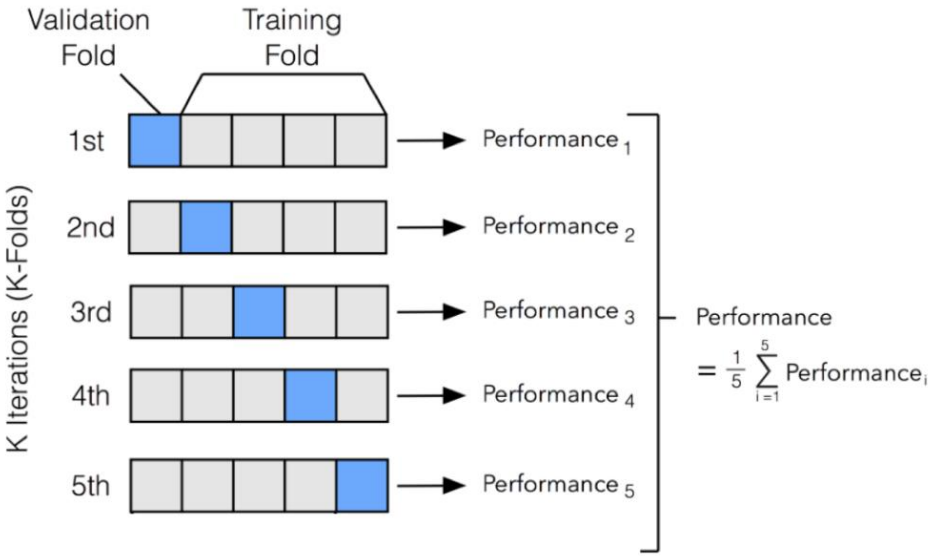

Figure 14: Explanation of 5-Fold Cross-Validation

The training of each ML model is completed with hyperparameter tuning. GridSearchCV method trains ML models by using cross-validation. Then, it gives the parameters of the ML model with the best performance after training of all ML models are completed. However, it cannot compare different types of estimators. Therefore, we choose the best among these trained ML models manually. During this selection process, the root mean squared error (RMSE) values of training and test datasets, and the ratios of the average value of the feature and the RMSE value are taken into account. RMSE values for train and test datasets for each ML model, their relative squared errors (RSEs), and the most successful hyper-parameter values for each ML model will be provided separately for each predicted value later on in this section.

4.1.3.1.1. Corrosion Inhibitor

We are predicting the corrosion inhibitor first. We use only input variables to train the estimators to predict the corrosion inhibitor feature. The model will use input variables as input only.

The performances provided and the best hyperparameters selected for each of the machine learning algorithms applied on datasets without and with polynomial expansion are given in Table 14 and Table 15, respectively. The regression types with the highest accuracy are marked with a star in both tables.

The linear, lasso, and ridge regressions provided the same result with the dataset without polynomial expansion since there is no overfitting to the train dataset. Therefore, the regularization that lasso and ridge regression and the feature selection that lasso regression applied do not create a remarkable difference. The small regularization parameter for both estimators proves that regularization is not remarkably effective. However, the positive effects of the regularization and the feature selection can be observed in training with the dataset with polynomial expansion.

KNN regression showed promising results, but it is not as good as RF regression did on both datasets. SVM regression with a linear kernel is more accurate with the dataset without polynomial expansion. However, SVM regression with RBF kernel is more accurate with the dataset with polynomial expansion. RF regression provided the best results on both datasets. The polynomial expansion increased accuracy. Having an RMSE nearly ten times smaller than the standard deviation is an acceptable accuracy for our model. Therefore, we go on with the RF regression model trained with the dataset with polynomial expansion to predict the amount of corrosion inhibitor typically injected by the field operators. The root relative squared error (RRSE) of the best model is 0.088 for the train dataset and 0.114 for the test dataset.

Table 14: Corrosion Inhibitor: The performances on the training/test datasets (without polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	1.070	0.584	0.567	54.58%	52.99%
Lasso	$\alpha = 0.00001$	1.070	0.584	0.567	54.58%	52.99%
Ridge	$\alpha = 0.00001$	1.070	0.584	0.567	54.58%	52.99%
RF(*)	n=25, d=8, ms=5, minl=1, maxl=∞, mfw=0	1.070	0.136	0.137	12.71%	12.80%
KNN	n=3	1.070	0.146	0.224	13.64%	20.93%
SVM	Kernel='linear'	1.070	0.586	0.568	54.76%	53.08%

Table 15: Corrosion Inhibitor: The performances on the training/test datasets (with polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	1.070	0.347	0.345	32.43%	32.24%
Lasso	$\alpha = 0.00001$	1.070	0.320	0.315	29.91%	29.44%
Ridge	$\alpha = 0.00001$	1.070	0.289	0.930	27.01%	86.92%
RF(*)	n=75, d=8, ms=8, minl=1, maxl=∞, mfw=0	1.070	0.095	0.122	8.98%	11.40%
KNN	n=3	1.070	0.171	0.27	15.98%	25.23%
SVM	Kernel='rbf', C=1	1.070	0.919	0.906	85.89%	84.67%

4.1.3.1.2. Neutralizer Amine

The second variable we are predicting is the neutralizer amine. The model will use input variables as input only.

The performances provided and the best hyperparameters selected for each of the machine learning algorithms applied on datasets without and with polynomial expansion are given in Table 16 and Table 17, respectively. The regression types with the highest accuracy are marked with a star in both tables.

The linear, lasso, and ridge regressions provided the same result with the dataset without polynomial expansion since there is no overfitting to the train dataset. Therefore, the regularization that lasso and ridge regression and the feature selection that lasso regression applied do not create a remarkable difference. The small regularization parameter for both estimators proves that regularization is not remarkably effective. However, the positive effects of the regularization and the feature selection can be observed in training with the dataset with polynomial expansion.

KNN regression showed promising results, but it is not as good as RF regression did on both datasets. SVM regression with a linear kernel is more accurate with the dataset without polynomial expansion. However, SVM regression with RBF kernel is more accurate with the dataset with polynomial expansion. RF regression provided the best results on both datasets. The polynomial expansion increased accuracy. The RMSE of the best model is nearly thirteen times smaller than the standard deviation is a better accuracy for our model. Therefore, we go on with the RF regression model trained with the dataset with polynomial expansion to predict the amount of neutralizer amine typically injected by the field operators. RRSE of the best model is 0.067 for the train dataset and 0.08 for the test dataset.

Table 16: Neutralizer Amine: The performances on the training/test datasets (without polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	1.700	0.617	0.613	36.29%	36.06%
Lasso	α = 0.00001	1.700	0.617	0.613	36.29%	36.06%
Ridge	α = 0.00001	1.700	0.617	0.612	36.29%	36.00%
RF(*)	n=200, d=8, ms=5, minl=1, maxl=∞, mfw=0	1.700	0.138	0.154	8.12%	9.06%
KNN	n=3	1.700	0.170	0.235	10.00%	13.82%
SVM	Kernel='linear'	1.700	0.629	0.621	37.00%	36.52%

Table 17: Neutralizer Amine: The performances on the training/test datasets (with polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE on Train Dataset	RMSE on Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear	-	1.700	0.395	0.41	23.24%	24.12%
Lasso	$\alpha = 0.00001$	1.700	0.380	0.387	22.35%	22.76%
Ridge	$\alpha = 0.00001$	1.700	0.502	0.726	29.53%	42.71%
RF(*)	n=50, d=8, ms=5, minl=1, maxl=∞, mfw=0	1.700	0.111	0.135	6.53%	7.94%
KNN	n=3	1.700	0.176	0.254	10.35%	14.94%
SVM	Kernel='rbf', C=1	1.700	0.936	0.940	55.05%	55.29%

4.1.3.1.3. Caustic Soda

We are predicting the caustic soda as the last chemical. We use only input variables to train the estimators to predict the caustic soda feature.

The performances provided and the best hyperparameters selected for each of the machine learning algorithms applied on datasets without and with polynomial expansion are given in Table 18 and Table 19, respectively. The regression types with the highest accuracy are marked with a star in both tables.

The linear, lasso, and ridge regressions provided the same result with the dataset without polynomial expansion since there is no overfitting to the train dataset. Therefore, the regularization that lasso and ridge regression and the feature selection that lasso regression applied do not create a remarkable difference. The small regularization parameter for both estimators proves that regularization is not remarkably effective. However, the positive effects of the regularization and the feature selection can be observed in training with the dataset with polynomial expansion.

KNN regression showed promising results, but it is not as good as RF regression did on both datasets. SVM regression with a linear kernel is more accurate with the dataset without polynomial expansion. However, SVM regression with the polynomial kernel is more accurate with the dataset with polynomial expansion. RF regression provided the best results on both datasets. The polynomial expansion increased accuracy. The RMSE of the best model is nearly twenty times smaller than the standard deviation is a better accuracy for our model. Therefore, we go on with the RF regression model trained with the dataset with polynomial expansion to predict the amount of caustic soda typically injected by the field operators. RRSE of the best model is 0.036 for the train dataset and 0.047 for the test dataset.

Table 18: Caustic Soda: The performances on the training/test datasets (without polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	85.230	52.040	52.030	61.06%	61.05%
Lasso	$\alpha = 0.00001$	85.230	52.040	52.030	61.06%	61.05%
Ridge	$\alpha = 0.00001$	85.230	52.040	52.030	61.06%	61.05%
RF(*)	n=75, d=8, ms=2, minl=1, maxl=∞, mfw=0	85.230	3.840	5.570	4.51%	6.54%
KNN	n=3	85.230	8.980	13.870	10.54%	16.27%
SVM	Kernel='linear'	85.230	60.910	61.010	71.47%	71.58%

Table 19: Caustic Soda: The performances on the training/test datasets (with polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	85.230	26.260	26.400	30.81%	30.98%
Lasso	$\alpha = 0.00001$	85.230	23.340	23.630	27.38%	27.73%
Ridge	$\alpha = 0.00001$	85.230	28.270	53.060	33.17%	62.26%
RF(*)	n=125, d=8, ms=2, minl=1, maxl=∞, mfw=0	85.230	3.050	4.200	3.58%	4.93%
KNN	n=3	85.230	9.330	13.540	10.95%	15.89%
SVM	Kernel='poly', degree=3, C=1	85.230	75.610	76.50	88.71%	89.76%

4.1.3.1.4. pH of Boot Water

After we predicted the amount of chemicals to use based on the currently employed approach of the experts, we are going to predict the control variables. These three models are going to be used to determine the constraints in the optimization problem that we will be formulating in the following sections. We are going to predict the pH of boot water first. We use both input and set variables to train the estimators to predict the pH of the boot water feature.

The performances provided and the best hyperparameters selected for each of the machine learning algorithms applied on datasets without and with polynomial expansion are given in Table 20 and Table 21, respectively. The regression types with the highest accuracy are marked with a star in both tables.

The linear, lasso, and ridge regressions provided almost the same result with the dataset without polynomial expansion since there is no overfitting to the train dataset. Therefore, the regularization that lasso and ridge regression and the feature selection that lasso regression applied do not create a remarkable difference. The small regularization parameter for both estimators proves that regularization is not remarkably effective. However, the positive effects of the regularization and the feature selection can be observed in training with the dataset with polynomial expansion.

SVM regression with a linear kernel is more accurate with the dataset without polynomial expansion. However, SVM regression with RBF kernel is more accurate with the dataset with polynomial expansion. Even though RF regression provided good results, KNN regression provided the best results for both datasets this time. The RMSE of the best estimator is nearly one-third of the standard deviation. The results are not the best, but it still works for our study. Therefore, we select the KNN regression model trained with the dataset without polynomial expansion to predict the pH of boot water. RRSE of the best model is 0.179 for the train dataset and 0.277 for the test dataset.

Table 20: pH of the Boot Water: The performances on the training/test datasets (without polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.200	0.170	0.176	85.00%	88.00%
Lasso	$\alpha = 0.00001$	0.200	0.170	0.176	85.00%	88.00%
Ridge	$\alpha = 0.00001$	0.200	0.170	0.176	85.00%	88.00%
RF	n=100, d=8, ms=5, minl=1, maxl=∞, mfw=0	0.200	0.073	0.079	36.50%	39.50%
KNN (*)	n=3	0.200	0.035	0.055	17.50%	27.50%
SVM	Kernel='rbf', C=1	0.200	0.182	0.186	91.00%	93.00%

Table 21: pH of the Boot Water: The performances on the training/test datasets (with polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.200	0.120	0.133	60.00%	66.50%
Lasso	$\alpha = 0.00001$	0.200	0.110	0.118	55.00%	59.00%
Ridge	$\alpha = 0.00001$	0.200	0.080	0.310	40.00%	155.00%
RF	n=50, d=8, ms=5, minl=1, maxl=∞, mfw=0	0.200	0.092	0.099	46.00%	49.50%
KNN (*)	n=3	0.200	0.036	0.058	18.00%	29.00%
SVM	Kernel='rbf', C=1	0.200	0.175	0.179	87.50%	89.50%

4.1.3.1.5. Iron in Boot Water

After predicting the pH of boot water, we are going to predict the iron amount in boot water. We use both input and set variables to train the estimators to predict the iron amount in the boot water feature.

The performances provided and the best hyperparameters selected for each of the machine learning algorithms applied on datasets without and with polynomial expansion are given in Table 22 and Table 23, respectively. The regression types with the highest accuracy are marked with a star in both tables.

The linear, lasso, and ridge regressions provided almost the same result with the dataset without polynomial expansion since there is no overfitting to the train dataset. Therefore, the regularization that lasso and ridge regression and the feature selection that lasso regression applied do not create a remarkable difference. The small regularization parameter for both estimators proves that regularization is not remarkably effective. However, the positive effects of the regularization and the feature selection can be observed in training with the dataset with polynomial expansion.

SVM regression with the polynomial kernel is more accurate with the dataset without polynomial expansion. However, SVM regression with RBF kernel is more accurate with the dataset with polynomial expansion. Even though RF regression provided good results, KNN regression provided the best results for both datasets this time. The RMSE of the best estimator is nearly half of the standard deviation. The results are not perfect, but it still works. Therefore, we select the KNN regression model trained with the dataset without polynomial expansion to predict the amount of iron in the boot water. RRSE of the best model is 0.261 for the train dataset and 0.341 for the test dataset.

Table 22: Iron Amount in the Boot Water: The performances on the training/test datasets (without polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.330	0.301	0.350	91.21%	106.06%
Lasso	$\alpha = 0.00001$	0.330	0.301	0.350	91.21%	106.06%
Ridge	$\alpha = 0.00001$	0.330	0.301	0.350	91.21%	106.06%
RF	n=25, d=8, ms=2, minl=1, maxl=∞, mfw=0	0.330	0.167	0.187	50.61%	56.67%
KNN (*)	n=3	0.330	0.085	0.126	25.76%	38.18%
SVM	Kernel='poly', degree=5, C=1	0.330	0.309	0.360	93.64%	109.09%

Table 23: Iron Amount in the Boot Water: The performances on the training/test datasets (with polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.330	0.239	0.287	72.42%	86.97%
Lasso	$\alpha = 0.00001$	0.330	0.227	0.234	68.79%	70.91%
Ridge	$\alpha = 0.00001$	0.330	0.148	0.214	44.85%	64.85%
RF	n=25, d=8, ms=2, minl=1, maxl=∞, mfw=0	0.330	0.168	0.185	50.91%	56.06%
KNN (*)	-	0.330	0.086	0.132	26.06%	40.00%
SVM	Kernel='rbf', C=1	0.330	0.306	0.358	92.73%	108.48%

4.1.3.1.6. Chlorine in Boot Water

We are going to predict the amount of chlorine in boot water lastly. We use both input and set variables to train the estimators to predict the amount of chlorine in the boot water feature.

The performances provided and the best hyperparameters selected for each of the machine learning algorithms applied on datasets without and with polynomial expansion are given in Table 24 and Table 25, respectively. The regression types with the highest accuracy are marked with a star in both tables.

The linear, lasso, and ridge regressions provided almost the same result with the dataset without polynomial expansion since there is no overfitting to the train dataset. Therefore, the regularization that lasso and ridge regression and the feature selection that lasso regression applied do not create a remarkable difference. The small regularization parameter for both estimators proves that regularization is not remarkably effective. However, the positive effects of the regularization and the feature selection can be observed in training with the dataset with polynomial expansion.

SVM regression with the polynomial kernel is more accurate with the dataset without polynomial expansion. However, SVM regression with RBF kernel is more accurate with the dataset with polynomial expansion. Even though RF regression provided good results, KNN regression provided the best results for both datasets again. The RMSE of the best estimator is nearly a quarter of standard deviation. This result is applicable, according to us. Therefore, we select the KNN regression model trained with the dataset without polynomial expansion to predict the amount of chlorine in the boot water. RRSE of the best model is 0.164 for the train dataset and 0.218 for the test dataset.

Table 24: Chlorine Amount in the Boot Water: The performances on the training/test datasets (without polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	4.150	3.120	3.060	75.18%	73.73%
Lasso	$\alpha = 0.00001$	4.150	3.120	3.060	75.18%	73.73%
Ridge	$\alpha = 0.00001$	4.150	3.120	3.060	75.18%	73.73%
RF	n=75, d=8, ms=2, minl=1, maxl=∞, mfw=0	4.150	1.170	1.280	28.19%	30.84%
KNN (*)	-	4.150	0.680	0.890	16.39%	21.45%
SVM	Kernel='poly', degree=3, C=1	4.150	3.560	3.450	85.78%	83.13%

Table 25: Chlorine Amount in the Boot Water: The performances on the training/test datasets (with polynomial expansion) and the best hyperparameters of applied ML algorithms.

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE on Train Dataset**	**RMSE on Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	4.150	2.220	2.340	53.49%	56.39%
Lasso	$\alpha = 0.00001$	4.150	2.070	2.140	49.88%	51.57%
Ridge	$\alpha = 0.00001$	4.150	1.170	1.480	28.19%	35.66%
RF	n=125, d=8, ms=2, minl=1, maxl=∞, mfw=0	4.150	0.950	1.060	22.89%	25.54%
KNN (*)	-	4.150	0.700	0.990	16.87%	23.86%
SVM	Kernel='rbf', C=1	4.150	3.550	3.460	85.54%	83.37%

4.1.3.1.7. Summary of Model Selection

We trained six ML models for each variable with datasets without and with polynomial expansion. Therefore, we have trained 72 different ML models to predict six variables in total. After applying the grid-search with cross-validation, we determine the best estimator for each variable. The best estimators are selected concerning the RSE on the train and test datasets. The summary of the most accurate models is given in Table 26.

Table 26: Regression types and order of polynomial expansion for each variable that provides the most accurate result

Variable Name	Polynomial Expansion on Dataset	Regression Type
Corrosion Inhibitor	Second-order (with)	Random forest regression
Neutralizer Amine	Second-order (with)	Random forest regression
Caustic Soda	Second-order (with)	Random forest regression
pH of Boot Water	Not applied (without)	KNN regression
Iron in Boot Water	Not applied (without)	KNN regression
Chlorine in Boot Water	Not applied (without)	KNN regression

4.1.4. Data Mining and Analytics

After the determination of the most accurate models for all of the six variables, we store the model coefficients by using the pickle library of the Python for analytics purposes. The analytics performed with these ML models have three aims:

- First, three models provide the amount of chemicals to be injected in real-time. It is useful because the system can help even the experienced employees if they miss some points, which may affect the product quality because of the inappropriate amount of chemicals injected.
- Second, transfer the knowledge of the chemical injection method of experienced employees to inexperienced or future employees of the company by some test cases. In these cases, employees see the real-time input variables coming from the sensor readings and offer the amount of the chemicals to be injected to prevent corrosion in a digital environment. Therefore, they can observe whether the amount of corrosion preventive chemicals is successful or not by following the prediction results of the properties of boot water, i.e., observed variables.
- Third, models trained to predict observed variables are going to be used to determine the constraints in optimization.

4.2. Optimization

The main focus of this study is to optimize the use of injected chemicals to reduce corrosive effects in production tanks of crude oil refineries. The objective function of the optimization problem in this study is to minimize the costs of using these chemicals. In this thesis, we focused on minimizing the total purchase cost of chemicals to use. Nevertheless, it is possible to include the maintenance costs if the cost estimations for the unit amount of chemicals are available. However, it is not easy to estimate the maintenance costs since the maintenance costs increase cumulatively on the contrary to purchase costs. For example, excessive use of caustic soda leaves unusable residue behind. However, we do not have any well-defined model for the amount of residue. Besides, cleaning the residues cannot be performed while the plant is operational. Consequently, a stop that is not known how long it will take is necessary to clean the tanks. During the stop, the company loses money since they are not producing any goods, paying for employees, paying extra money to clean tanks. Besides, there is no pre-defined formula for estimation of maintenance costs. Because of these reasons, we only focused on minimizing the purchase costs of chemicals. We set several meetings with the field operation staff to determine the purchase costs of three chemicals, also stated as output variables for that reason. The costs are determined with units/lt due to confidentiality concerns. Total costs determined as follows:

- Corrosion inhibitor : 24 units/lt
- Neutralizer amine : 4.8 units/lt
- Caustic soda : 1 units/lt

Defining the constraints of the problem is the second step. In Chapter 3, it is stated that our limitations come from the fact that iron and chlorine in boot water and pH of boot water should be in certain ranges, which is defined as the operating region in the literature. In Chapter 3, these values ranges are given as follows:

- the pH of boot water should be between 5.5 and 6.5
- Iron in boot water should be between 0.2 ppm and 1 ppm
- Chlorine in boot water should be between 10 ppm and 40 ppm

Therefore, our optimization problem is formulated as follows:

$$minimize: (24 * \ Amount_{ci}) + (4.8 * Amount_{na}) + (1 * Amount_{cs})$$

$$Subject\ to:$$

$$5.5 \leq pH\ of\ Boot\ Water \leq 6.5$$

$$0.2 \leq Iron\ in\ Boot\ Water \leq 1$$

$$10 \ \leq Chlorine\ in\ Boot\ Water \leq 40$$

where the variables indicate:

- $Amount_{ci}$: the amount of injected corrosion inhibitor in terms of a liter
- $Amount_{na}$: the amount of injected neutralizer amine in terms of a liter
- $Amount_{cs}$: the amount of injected caustic soda in terms of a liter

There are six different ML models trained in the first step. Three of these models are to predict the iron and chlorine in boot water and pH of boot water, which are useful to set constraints of the optimization problem. These models use both input variables coming from the specifications of crude oil and the rate of chemicals to be injected as the input. Therefore, the predictions arising from these models can be used to define the constraints in the optimization problem.

Two different optimization solvers are used to optimize the objective function after the problem definition is completed. These solvers were using genetic algorithm and simulated annealing. Both of these solvers require the minimum and maximum values to be searched manually. We inspected the dataset, and we identified the minimum and maximum values that injected chemicals took. They are given in Table 27.

Table 27: Minimum and maximum values of chemicals in the dataset

Variable Name	Minimum Value (ppm)	Maximum Value (ppm)
Corrosion Inhibitor	6.98	10.93
Neutralizer Amine	11.09	18.61
Caustic Soda	450.00	800.00

We think that experts might have not been considering feasible combinations beyond these ranges. Therefore, we decided to set lower and upper bounds of the values that optimizers can search, as stated in Table 28.

Table 28: Minimum and maximum bounds of the chemical rates that optimizers can search

Chemical	Minimum Value	Maximum Value
Corrosion Inhibitor	6	12
Neutralizer Rate	10	20
Caustic Soda	400	900

After these values are set, we tested both optimizers individually. Solutions were performed on the same desktop PC with the specifications given in Table 29.

Table 29: Specification of the computer solving optimization problems

Component	Brand	Model	Capacity
Processor	Intel	Core i7-8700K	3.7 GHz (12 CPUs)
RAM	Corsair	CMK16GX4M1A2666C16	48 GB (3*16GB) DDR4 - 2666 MHz
Storage	Samsung	970 EVO NVMe M.2	500 GB
Graphics Card	Asus	1070TI Turbo	8 GB GDDR5 RAM

4.2.1. Genetic Algorithm

The genetic algorithm relies on a start with a population randomly determined to fit in the constraints. Therefore, we created a population with eight members satisfying the constraints. The fitness of each chromosome in the population is calculated later, which is the result of values when replaced with the cost function given above in our case. Then, the population must go on with evolving to find the best member. The production of the new population is started with selection. The selection is based on picking the best members, which provides the best solution for the problem. The best four members of the population chosen for breeding. The next step is the crossover, where the decision process of how the new offspring get the genes of their parents. In our case, the new generations have half of the genes from the first parent and the rest from the second parent. The breeding process goes on with the mutation where the genes of new offspring have a chance to mutate the genes coming from their parents. In our case, different mutations were also allowed if they satisfy the constraints that the population restricted to, which is given in Table 1. The mutations occurred in one variable in each generation, which means the mutation probability of a variable is 1/3. In the last step, the new generation of the family is generated. The total number of members in the population is eight again. This process goes on for a thousand generations. However, the process took around three minutes and fifty seconds in the average of fifty trials, which in our test device is so long to offer real-time predictions. Also, the solver did not converge to the global minimum in all the cases, which means it requires more generations and more runtime to converge. As a result, we agreed that the genetic algorithm did not yield as good results as expected.

4.2.2. Simulated Annealing

As the results of the genetic algorithm were not satisfying for us, we decided to use a more robust optimization solver that can guarantee a global solution. We chose to use simulated annealing at this point. It is because that the simulated annealing also provides global solutions, and Python has robust libraries to solve it. We used the SciPy library of Python to solve the problem by using simulated annealing. The results were consistent across all different trials. The average runtime of fifty test cases was around one minute and forty-five seconds for simulated annealing. As a result, simulated annealing runs faster than twice when compared to a genetic algorithm. Besides, it was able to find the global minimum for each case.

4.3. Outcomes

We delivered software to the employees of the company. This software, first of all, offers its users the chemical rates which will be added to the system by users considering the characteristics of the crude oil and the current status of the tanks via sensor measurements. This specification is the replication of the ongoing process at the crude oil plant. Secondly, it shows the characteristics of the crude oil and the current status of the tanks to its users by using the measurements from the sensors. It requests the rate of chemicals to be added from users as input. It presents the predictions of the variables that we control the success of the process according to the added chemical amounts afterward. The users can learn the process by using the following specification. Thirdly, it suggests the optimum mixture of chemical injection rates, which minimizes the purchase costs. Users of the software do not have to use the recommended amounts. Suggestions are intended to support users' decisions. From this point of view, it can be said that the software offered to users is a mixture of DSS and BI&A tools. In this section, we are going to explain these outcomes.

4.3.1. Modeling the Approach of Field Operation Staff for Chemical Injection

One of the demanded outputs of this study was to model the approach of field operation staff. This function is to help the field operation staff in cases where the experienced ones are away. It is also helpful to transfer the knowledge of experienced ones to others.

The ML models trained with only input variables can be used for predictions as an intermediate outcome. We use models trained with the dataset with polynomial expansion to predict the amount of chemicals since their results are better than others.

The software pulls current values of the input variables from the company database in real-time and passes these values to the models to predict the rate of chemicals to inject. Predicted values are stored in another database.

Field operation staff stated that the software provides results better than they imagine. The two users of the system said that our system could observe the specifications of the petroleum and nicely decide to increase or decrease the amount of the chemicals. Suggestions that the system provide did not cause any erroneous production for three months of use. In several cases, the BI&A tool detected process upsets even experienced field operation staff did not foresee.

4.3.2. Predicting the Success of a Mock-up Chemical Injection

The second demanded output of this study was to predict the success of a process before it has happened. The prediction should take place in a digital environment. Such specification is used to train employees about the chemical injection process, and they can control how a process will end if they want to. Therefore, we added this specification to the software.

4.3.3. Providing the Most Cost-Effective Mixture of Chemicals for Chemical Injection

The third demanded output of this study was to find the most cost-effective mixture of chemicals to be injected into the system. Such specification is useful to reduce their

costs as well as to reduce faulty productions. The software delivers chemical injection rates with the lowest cost to the field operations staff by performing the optimization processes explained previously. The results are shown in a couple of minutes, depending on the state of servers of the company that hosts the software.

4.4. Results

The use of the software developed in this thesis study aims to reduce costs and to help field experts to replicate the ongoing process, and transfer knowledge to future employee generations. When the analysis of the results was planned, it was realized that a long-term test period was needed to evaluate the success of the tool both in transferring the information to the next generations and estimation of the optimal amount of chemicals. In addition, the company does not provide information regarding the duration of the test period and their transition process to use the tool. Since it is thought that the process will take a long time to observe its results, we met with the domain experts who used our software and asked their opinions regarding the software and its results. Their feedback was positive for both software and its results.

In addition, we planned to prepare a test-case scenario with the available data to estimate how much the software could reduce the purchase costs by means of the provided optimization method. We planned to calculate the approximate costs of chemical purchases according to the dataset by using the determined prices given in section 4.2. First, we calculated the costs of chemicals used in reality. Later on, we calculated the costs of the suggested amount of chemicals. Then, we compared these results according to the plan.

During the test, we used the dataset that had been used to train our ML models. The dataset consists of records between the dates 25th August 2016 to 30th September 2019. Then, we calculated the purchase costs of the chemicals used during the time interval by using the prices given in section 4.2. The costs are calculated as 72069 units. Later, we calculated the costs of optimized values by our software. The costs are calculated as 62124 units. As a result, the use of our tool promises 9945 units savings in nearly three years of use, which means roughly 14% of the reduction in costs. Besides, 76.8% of the time, the used amount of chemicals caused faulty production according to the data of iron and chlorine in boot water and pH of boot water. We expect that using the optimized values is going to reduce the recurrence of faulty production.

CHAPTER 5

CONCLUSION

In this thesis study, a DSS software that can optimize the amount of injected chemicals in order to prevent the corrosion problem in the production tanks of crude oil refineries is developed. The methods used previously to determine the injected chemical quantities of the refinery we worked with were based on heuristic approaches. However, when the success of the production using the amounts presented as a result of the heuristic approaches in the refinery was measured, it was seen that 76.8% of the processes did not meet the industry standards and qualities suggested by academic studies. Also, it is seen in the literature that these faulty productions cause earlier rusting in the tanks or the accumulation of unwanted chemical residues [47], [48]. As a result, the refinery performs maintenance more frequently, which results in company losses. On top of that, suggesting an educated guess requires an experience and domain knowledge that the newer employees may not have. Therefore, it is not likely that the heuristic approach is used in an environment where recruitment and departures of employees frequently occur since the learning curve of the recruited employees.

In real-life applications, there are some proposals to determine the amount of the injected chemicals. However, as mentioned in Chapter 4, these studies have several shortcomings, such as they are based on heuristic methods, the learning curve of suggesting the amounts in the heuristic way is steep, and the performance of such practices is not optimal all the time. No study aims to flatten the learning curve, to suggest more consistent amounts of chemicals, or to optimize the costs and the amounts of injected chemicals. In this study, we proposed an approach that provides all of these crucial features.

Thus, the main contributions of this thesis study are as follows:

- Transfer of the knowledge of the experienced company employees to the future company employees,
- Ensuring the production of products that do meet the industry standards by injecting a sufficient amount of chemicals,
- The decrease in both the purchase cost of chemicals and the maintenance cost of the production tanks.

We have formulated an optimization problem based on surrogate modeling since the modeling of the entire process is complicated and may not even be feasible. In this method, we train ML models to capture relationships between different process parameters. These ML models are used as surrogates and serve as basic formulas that can be used to define the constraints of the optimization problem. In addition, these ML models are also used in the training of employees.

We could not find any previous research in our literature review on the topic of the optimization of the chemical injection in crude oil refineries. Therefore, we believe it is a new approach to optimize the chemical injection process, and the approach that we propose in this thesis study contributes to the industry as well. According to the study carried on existing data for three years, the approach promises 9945 units

savings, which means nearly 14% of the reduction in purchase costs in three years. Besides, around 76.8% of the time, the used amount of chemicals caused faulty production according to the data of iron and chlorine in boot water and pH of boot water. We expect that using the optimized values is going to reduce the recurrence of faulty production. Moreover, the field operation staff verbally stated that the software could detect the changes and trends in changes correctly, which is another indicator that the software works successfully.

There exist several limitations regarding the prototype implementation. The most significant one is that the study only focuses on purchasing costs. However, our objective function can be extended to incorporate maintenance costs if one can quantify the maintenance costs caused by the inappropriate use of chemicals. Another limitation is the dataset used to train ML models is not large enough in terms of the number of records. This lack of records limits us to capture different operating conditions to model the relationships between variables. The last limitation is that the data used in this study is not collected and stored correctlyimproper data collection methodologies in the refinery, causing inconsistencies. These data collection methodologies are not proper since they do not consider time lag and very prone to human error. First of all, there is a time lag between the variables because the process occurs in a flow, but the data collection is performed with respect to time. As a result, the values on the same record are not measured for the same crude oil particles. Secondly, some of the variables are not measured by the sensors. Specialists collect some samples from the different sections of the system and analyze them in the laboratory. Then, they enter the collection time of the samples and the analysis results of the samples to the database manually. This approach may lead to two problems, such as errors in analysis results and the incorrect entrance of collection time and analysis results. The last limitation may reduce the accuracy of the ML models and, consequently, the surrogates.

We are planning to extend our work by:

- Including maintenance costs to optimization
- Including deep learning based models to have more complicated models which may help to reveal undiscovered relations between features
- Exploiting the flexibility of our methodology, convert the surrogate models from regression based models to classification based models
- Applying time-series based approaches by incorporating more granular data
- Adapting our methodology to other sections of the refineries

Including maintenance costs in optimization requires regular long-term measurements as pointed out in Section 3.2.2. In the future, we plan to include maintenance costs by taking these measurements for a considerably long time with the refinery's collaboration.

Using neural networks may help developing models that extract the most complex relationships between the variables. Therefore, developing deep learning based models can enable having better surrogates.

The problem of the study is suitable for regression based estimators. However, it may also be possible to formulate the estimation problems as classification tasks. That is, instead of estimating precise values of the observed variables, the models may predict

whather the target values are between the allowed ranges or not. This can help to reduce the complexity of the problem and allow making better approximations to the optimization constraints.

In this study, we preferred regression models as the sampling periods of some variables are very long. In the future, by decreasing the sampling period, the problem can be solved by using time-series estimators such as autoregressive integrated moving average (ARIMA), Prophet, or recurrent neural network models.

Finally, we are planning to model and optimize other processes in the crude oil refineries, such as desalination, to prevent corrosion. Finally, we can extend the study to optimize operations in different blocks in the refinery, such as crude oil distillation.

REFERENCES

[1] D. Wetherill, "Broken links Why analytics investments have yet to pay off," 2016.

[2] M. O. Gökalp, K. Kayabay, S. Çoban, Y. B. Yandık, and P. E. Eren, "Büyük Veri Çağında İşletmelerde Veri Bilimi," no. March 2019, pp. 94–97, 2018.

[3] S. R. Department, "Number of internet of things (IoT) connected devices worldwide in 2018, 2025 and 2030," 2019. [Online]. Available: https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/. [Accessed: 03-Apr-2020].

[4] H. Chen, R. H. L. Chiang, and V. C. Storey, "Business Intelligence and Analytics: From Big Data to Big Impact," *MIS Q.*, vol. 36, no. 4, pp. 1165–1188, 2012.

[5] N. A. Ibrahim, N. M. A. Moneim, M. A. Ramadan, and M. M. Hosni, "A Multiple-Objective Environmental Rationalization and Optimization for Material Substitution in the Production of Stone-Washed Jeans- Garments," pp. 435–442, 2008.

[6] M. Fosouli, "Explanation of the importance of innovation and government support on profit of Food industry companies listed on the Tehran Stock Exchange," pp. 789–803, 2016.

[7] H. Gangurde and H. A. Chavan, "International Journal of Modern Trends in Engineering and Research CO 2 Welding Process Optimization by using SPM," no. 2349, pp. 28–30, 2016.

[8] . A. N. P., "Labour Productivity Improvement By Work Study Tools of Fiber Composite Company," *Int. J. Res. Eng. Technol.*, vol. 05, no. 09, pp. 351–355, 2016, doi: 10.15623/ijret.2016.0509054.

[9] S. Gupta, P. Sarkar, and E. Singla, "Understanding different stakeholders of sustainable product and service-based systems using genetic algorithm," *Clean Technol. Environ. Policy*, vol. 17, no. 6, pp. 1523–1533, 2015, doi: 10.1007/s10098-014-0880-y.

[10] A. Bhosekar and M. Ierapetritou, "Advances in surrogate based modeling, feasibility analysis, and optimization: A review," *Comput. Chem. Eng.*, vol. 108, pp. 250–267, 2018, doi: 10.1016/j.compchemeng.2017.09.017.

[11] V. Tang, K. Otto, and W. Seering, *Executive Decision Synthesis: A Sociotechnical Systems Paradigm*. Springer, 2018.

[12] E. Slavcheva, B. Shone, and A. Turnbull, "Review of naphthenic acid corrosion in oil refining," *Br. Corros. J.*, vol. 34, no. 2, pp. 125–131, 1999, doi: 10.1179/000705999101500761.

[13] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data Mining and Analytics in the Process Industry: The Role of Machine Learning," *IEEE Access*, vol. 5, pp. 20590–20616, 2017, doi: 10.1109/ACCESS.2017.2756872.

[14] M. Copeland, "What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?," *NVIDIA Blogs*, 2016. [Online]. Available: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/. [Accessed: 09-Mar-2020].

[15] P. Hall, J. Dean, I. K. Kabul, and J. Silva, "An Overview of Machine Learning with SAS ® Enterprise Miner ™," *An Overv. Mach. Learn. with SAS® Enterp. Miner™*, no. Rosenblatt 1958, pp. 1–24, 2014.

[16] K. R. Holdaway and K. Ball, "Derived Seismic Attributes Underpin Reservoir

Characterization in Data-Driven Methodologies," no. November, 2015, doi: 10.2118/175606-ms.
[17] S. Russell and P. Norvig, *Artificial Intelligence: A modern approach*. New Jersey, 1995.
[18] M. Peixeiro, "Linear Regression — Understanding the Theory," *Towards Data Science*, 2018. .
[19] D. A. Freedman, *Statistical Models: Theory and Practice*. New York: Cambridge University Press, 2009.
[20] A. Nagpal, "L1 and L2 Regularization Methods," *Towards Data Science*, 2017. [Online]. Available: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c. [Accessed: 26-May-2020].
[21] C. C. Chang and C. J. Lin, "LIBSVM: A Library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, 2011, doi: 10.1145/1961189.1961199.
[22] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.
[23] Scikit-learn, "SVM Kernels." [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html. [Accessed: 27-May-2020].
[24] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Am. Stat.*, vol. 46, no. 3, pp. 175–185, 1992, doi: 10.1080/00031305.1992.10475879.
[25] T. K. Ho, "Random decision forests," *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, vol. 1, pp. 278–282, 1995, doi: 10.1109/ICDAR.1995.598994.
[26] S. H. Kim and F. Boukouvala, "Machine learning-based surrogate modeling for data-driven optimization: a comparison of subset selection for regression techniques," *Optim. Lett.*, vol. 14, no. 4, pp. 989–1010, 2019, doi: 10.1007/s11590-019-01428-7.
[27] K. K. Vu, C. D'Ambrosio, Y. Hamadi, and L. Liberti, "Surrogate-based methods for black-box optimization," *Int. Trans. Oper. Res.*, vol. 24, no. 3, pp. 393–424, 2017, doi: 10.1111/itor.12292.
[28] K. McBride and K. Sundmacher, "Overview of Surrogate Modeling in Chemical Process Engineering," *Chemie-Ingenieur-Technik*, vol. 91, no. 3, pp. 228–239, 2019, doi: 10.1002/cite.201800091.
[29] W. M. Alley, "Regression Approximations for Transport Model Constraint Sets in Combined Aquifer Simulation-Optimization Studies," *Water Resour. Res.*, vol. 22, no. 4, pp. 581–586, 1986, doi: 10.1029/WR022i004p00581.
[30] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Stat. Sci.*, vol. 4, no. 4, pp. 409–423, 1989.
[31] Á. Fialho, Y. Hamadi, and M. Schoenauer, "A multi-objective approach to balance buildings construction cost and energy efficiency," *Front. Artif. Intell. Appl.*, vol. 242, pp. 961–966, 2012, doi: 10.3233/978-1-61499-098-7-961.
[32] R. K. Arora, *Optimization: Algorithms and Applications*. CRC Press, 2015.
[33] Wisconsin Institutes for Discovery, "Types of Optimization Problems," 2013. [Online]. Available: https://neos-guide.org/optimization-tree. [Accessed: 07-Jun-2020].
[34] S. Hussain, M. Al-Hitmi, S. Khaliq, A. Hussain, and M. A. Saqib, "Implementation and comparison of particle swarm optimization and genetic algorithm techniques in combined economic emission dispatch of an independent power plant," *Energies*, vol. 12, no. 11, 2019, doi:

10.3390/en12112037.
[35] K. S. Tang, K. F. Man, S. Kwong, and Q. He, “Genetic Algorithms and their Applications,” *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 22–37, 1996, doi: 10.1109/79.543973.
[36] M. Trosset, “What is Simulated Annealing?,” *Optim. Eng.*, vol. 2, no. 2, pp. 201–213, 2001, doi: 10.1023/A:1013193211174.
[37] J. Ridpath, “Training your staff in data science? Here’s how to pick the right programming language,” 2018. [Online]. Available: https://towardsdatascience.com/training-your-staff-in-data-science-heres-how-to-pick-the-right-programming-language-dda349354b18. [Accessed: 03-Apr-2020].
[38] N. D. Team, “About Numpy.” [Online]. Available: https://numpy.org/.
[39] P. Team, “About pandas,” 2015. [Online]. Available: https://pandas.pydata.org/about/. [Accessed: 07-May-2020].
[40] S. Developers, “Getting Started,” 2019. [Online]. Available: https://scikit-learn.org/stable/getting_started.html#getting-started. [Accessed: 16-Apr-2020].
[41] S. Developers, “Scientific computing tools for Python,” 2020. [Online]. Available: https://www.scipy.org/about.html. [Accessed: 19-Jun-2020].
[42] S. Raschka, “MLxtend Documentation,” 2019. [Online]. Available: http://rasbt.github.io/mlxtend/. [Accessed: 16-Apr-2020].
[43] R. P. Mohanty and S. G. Deshmukh, “Evolution of a decision support system for human resource planning in a petroleum company,” *Int. J. Prod. Econ.*, vol. 51, no. 3, pp. 251–261, 1997, doi: 10.1016/S0925-5273(97)00077-7.
[44] F. Oliveira, P. M. Nunes, R. Blajberg, and S. Hamacher, “A framework for crude oil scheduling in an integrated terminal-refinery system under supply uncertainty,” *Eur. J. Oper. Res.*, vol. 252, no. 2, pp. 635–645, 2016, doi: 10.1016/j.ejor.2016.01.034.
[45] C. W. Chan, “An expert decision support system for monitoring and diagnosis of petroleum production and separation processes,” *Expert Syst. Appl.*, vol. 29, no. 1, pp. 131–143, 2005, doi: 10.1016/j.eswa.2005.01.009.
[46] R. E. Young, “Petroleum refining process control and real-time optimization,” *IEEE Control Syst. Mag.*, vol. 26, no. 6, pp. 73–83, 2006, doi: 10.1109/MCS.2006.252833.
[47] P. Bhowmik, M. D. Emam Hossain, and J. Ahmed Shamim, “Corrosion and its Control in Crude Oil Refininig Process,” no. September, pp. 28–29, 2012, doi: 10.13140/2.1.1349.0244.
[48] J. Scarborough, L. Mega-Franca, M. Ibrahim, and B. Hughes, “Minimise system upsets in high oil production facility throughout demulsifier chemical trial,” *Int. Pet. Technol. Conf. 2019, IPTC 2019*, 2019, doi: 10.2523/iptc-19496-ms.
[49] N. P. Hilton, N. Williams, E. W. Vetters, and S. Ronnie, “Refinery reduces crude overhead corrosion through analysis and online control,” in *NACE - International Corrosion Conference Series*, 2014.

APPENDICES

APPENDIX-A RESULTS OF ESTIMATOR TRAININGS WITH DATASET SPLIT EARLIER 80% DATA TO TRAIN AND LATER 20% DATA TO TEST CONCERNING TIME AXIS

Table 30: Corrosion Inhibitor: The performances on the training/test datasets (without polynomial expansion and train-test split with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE of Train Dataset	RMSE of Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear (*)	-	1.070	0.601	0.681	56.17%	63.64%
Lasso (*)	α = 0.00001	1.070	0.601	0.681	56.17%	63.64%
Ridge (*)	α = 0.00001	1.070	0.601	0.681	56.17%	63.64%
RF	n=25, d=4, ms=8, minl=5, maxl=2, mfw=0	1.070	0.754	1.305	70.47%	121.96%
KNN	n=4	1.070	0.163	1.198	15.23%	111.96%
SVM	Kernel='linear'	1.070	0.612	0.737	57.20%	68.88%

Table 31: Corrosion Inhibitor: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	1.070	0.359	0.640	33.55%	59.81%
Lasso (*)	α = 0.00001	1.070	0.459	0.525	42.90%	49.07%
Ridge	α = 0.00001	1.070	0.359	0.631	33.55%	58.97%
RF	n=50, d=6, ms=2, minl=5, maxl=2, mfw=0.1	1.070	0.720	1.315	67.29%	122.90%
KNN	n=5	1.070	0.215	1.242	20.09%	116.07%
SVM	Kernel='rbf', C=1	1.070	0.833	2.179	77.85%	203.64%

Table 32: Neutralizer Amine: The performances on the training/test datasets (without polynomial expansion and train-test split with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear (*)	-	1.700	0.576	0.870	33.88%	51.18%
Lasso (*)	$\alpha = 0.00001$	1.700	0.576	0.870	33.88%	51.18%
Ridge (*)	$\alpha = 0.00001$	1.700	0.576	0.870	33.88%	51.18%
RF	n=50, d=6, ms=8, minl=1, maxl=∞, mfw=0	1.700	0.193	1.102	11.35%	64.82%
KNN	n=3	1.700	0.127	1.011	7.47%	59.47%
SVM	Kernel='linear'	1.700	0.587	1.172	34.53%	68.94%

Table 33: Neutralizer Amine: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	1.700	0.353	1.706	20.76%	100.35%
Lasso	$\alpha = 0.00001$	1.700	0.623	1.014	36.65%	59.65%
Ridge	$\alpha = 0.00001$	1.700	0.369	1.514	21.71%	89.06%
RF	n=25, d=8, ms=5, minl=5, maxl=∞, mfw=0.1	1.700	0.583	1.026	34.29%	60.35%
KNN (*)	n=3	1.700	0.184	0.972	10.82%	57.18%
SVM	Kernel='rbf', C=1	1.700	0.937	1.099	55.12%	64.65%

Table 34: Caustic Soda: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	85.230	22.208	163.658	26.06%	192.02%
Lasso	α = 0.00001	85.230	26.251	154.717	30.80%	181.53%
Ridge	α = 0.00001	85.230	22.208	163.658	26.06%	192.02%
RF (*)	n=25, d=6, ms=8, minl=1, maxl=2, mfw=0	85.230	33.133	138.484	38.87%	162.48%
KNN	n=3	85.230	4.060	149.216	4.76%	175.07%
SVM	Kernel=’poly', degree=3, C=1	85.230	33.767	150.008	39.62%	176.00%

Table 35: Caustic Soda: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE of Train Dataset	RMSE of Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear	-	85.230	13.340	162.871	15.65%	191.10%
Lasso	α = 0.00001	85.230	19.454	157.063	22.83%	184.28%
Ridge	α = 0.00001	85.230	15.213	155.924	17.85%	182.94%
RF	n=25, d=8, ms=8, minl=5, maxl=∞, mfw=0	85.230	20.097	150.998	23.58%	177.17%
KNN	n=3	85.230	4.586	146.517	5.38%	171.90%
SVM (*)	Kernel=’poly', degree=5, C=1	85.230	75.606	76.495	88.71%	89.75%

Table 36: pH of the Boot Water: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.200	0.139	0.274	69.50%	137.00%
Lasso (*)	$\alpha = 0.00001$	0.200	0.177	0.260	88.50%	130.00%
Ridge	$\alpha = 0.00001$	0.200	0.139	0.274	69.50%	137.00%
RF	n=50, d=4, ms=8, minl=1, maxl=∞, mfw=0.1	0.200	0.136	0.288	68.00%	144.00%
KNN	n=5	0.200	0.037	0.261	18.50%	130.50%
SVM	Kernel='rbf', C=1	0.200	0.153	0.622	76.50%	311.00%

Table 37: pH of the Boot Water: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.200	0.096	1.896	48.00%	948.00%
Lasso	$\alpha = 0.00001$	0.200	0.124	0.311	62.00%	155.50%
Ridge	$\alpha = 0.00001$	0.200	0.103	1.337	51.50%	668.50%
RF	n=50, d=8, ms=5, minl=3, maxl=∞, mfw=0	0.200	0.049	0.268	24.50%	134.00%
KNN	n=5	0.200	0.037	0.267	18.50%	133.50%
SVM (*)	Kernel='rbf', C=1	0.200	0.175	0.179	87.50%	89.50%

Table 38: Iron Amount in the Boot Water: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.330	0.238	0.565	72.12%	171.21%
Lasso (*)	$\alpha = 20$	0.330	0.246	0.517	74.55%	156.67%
Ridge	$\alpha = 0.00001$	0.330	0.238	0.565	72.12%	171.21%
RF	n=25, d=8, ms=5, minl=1, maxl=∞, mfw=0	0.330	0.249	0.537	75.45%	162.73%
KNN	n=5	0.330	0.070	0.581	21.21%	176.06%
SVM	Kernel='poly', degree=4, C=1	0.330	0.246	0.527	74.55%	157.70%

Table 39: Iron Amount in the Boot Water: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.330	0.167	1.116	50.61%	338.18%
Lasso	$\alpha = 1$	0.330	0.241	0.531	73.03%	160.91%
Ridge	$\alpha = 0.00001$	0.330	0.176	0.684	53.33%	207.27%
RF (*)	n=25, d=8, ms=5, minl=1, maxl=∞, mfw=0	0.330	0.237	0.503	71.81%	152.42%
KNN	n=5	0.330	0.074	0.553	22.42%	167.58%
SVM	Kernel='rbf', C=1	0.330	0.247	0.542	74.85%	164.24%

Table 40: Chlorine Amount in the Boot Water: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	4.150	2.663	5.541	64.17%	133.52%
Lasso	$\alpha = 0.00001$	4.150	2.663	5.541	64.17%	133.52%
Ridge (*)	$\alpha = 0.00001$	4.150	2.717	5.465	65.47%	131.69%
RF	n=25, d=2, ms=8, minl=1, maxl=2, mfw=0	4.150	2.991	6.549	72.07%	157.81%
KNN	n=5	4.150	0.613	8.012	14.77%	193.06%
SVM	Kernel='rbf', C=1	4.150	3.010	9.727	72.53%	234.39%

Table 41: Chlorine Amount in the Boot Water: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	4.150	1.531	15.814	36.89%	381.06%
Lasso	$\alpha = 0.00001$	4.150	2.454	7.959	59.13%	191.78%
Ridge	$\alpha = 0.00001$	4.150	1.531	15.872	36.89%	382.46%
RF (*)	n=25, d=6, ms=8, minl=3, maxl=2, mfw=0.1	4.150	2.930	7.224	70.60%	174.07%
KNN	n=5	4.150	0.636	9.243	15.33%	222.72%
SVM	Kernel='rbf', C=1	4.150	2.873	15.347	69.23%	369.81%

APPENDIX-B RESUTLS OF ESTIMATOR TRAININGS WITH DATASET SPLIT THE EARLIEST 20% DATA TO TEST AND LATER 80% DATA TO TRAIN CONCERNING TIME AXIS

Table 42: Corrosion Inhibitor: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE of Train Dataset	RMSE of Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear	-	1.070	0.608	0.640	56.82%	59.81%
Lasso (*)	α = 0.00001	1.070	0.881	0.413	82.34%	38.60%
Ridge	α = 0.00001	1.070	0.612	0.608	57.20%	56.82%
RF	n=50, d=2, ms=5, minl=1, maxl=4, mfw=0.1	1.070	0.272	1.094	25.42%	102.24%
KNN	n=5	1.070	0.181	0.911	16.92%	85.14%
SVM	Kernel='poly', degree=3, C=1	1.070	0.824	0.710	77.00%	66.36%

Table 43: Corrosion Inhibitor: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	1.070	0.342	8.278	31.96%	773.64%
Lasso (*)	$\alpha = 0.00001$	1.070	0.470	0.448	43.93%	41.87%
Ridge	$\alpha = 0.00001$	1.070	0.393	1.293	36.73%	120.84%
RF	n=125, d=8, ms=5, minl=5, maxl=∞, mfw=0.1	1.070	0.127	0.968	11.87%	90.47%
KNN	n=5	1.070	0.220	0.886	20.56%	82.80%
SVM	Kernel='poly', degree=3, C=1	1.070	1.014	0.792	94.77%	74.02%

Table 44: Neutralizer Amine: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE of Train Dataset	RMSE of Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear	-	1.700	0.614	1.088	36.12%	64.00%
Lasso	α = 0.00001	1.700	0.614	1.089	36.12%	64.06%
Ridge	α = 0.00001	1.700	0.614	1.089	36.12%	64.06%
RF	n=50, d=2, ms=5, minl=1, maxl=4, mfw=0.1	1.700	0.565	2.485	33.24%	146.18%
KNN (*)	n=5	1.700	0.213	0.266	12.53%	15.65%
SVM	Kernel='linear'	1.700	0.624	1.124	55.52%	66.12%

Table 45: Neutralizer Amine: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE of Train Dataset	RMSE of Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear	-	1.700	0.383	5.463	22.53%	321.35%
Lasso	$\alpha = 0.00001$	1.700	0.587	1.049	34.53%	61.71%
Ridge	$\alpha = 0.00001$	1.700	0.399	1.775	23.47%	104.41%
RF	n=125, d=8, ms=5, minl=5, maxl=∞, mfw=0.1	1.700	0.700	2.227	41.18%	131.00%
KNN (*)	n=5	1.700	0.228	0.293	13.41%	17.24%
SVM	Kernel='rbf', C=1	1.700	0.919	2.815	54.06%	165.59%

Table 46: Caustic Soda: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE of Train Dataset	RMSE of Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear	-	85.230	51.731	100.435	60.70%	117.84%
Lasso (*)	α = 0.00001	85.230	61.303	34.689	71.93%	40.70%
Ridge	α = 0.00001	85.230	54.757	49.786	64.25%	58.41%
RF	n=25, d=2, ms=8, minl=3, maxl=6, mfw=0.1	85.230	37.450	56.571	43.94%	66.37%
KNN	n=3	85.230	9.116	103.183	10.62%	121.06%
SVM	Kernel='linear'	85.230	64.267	60.027	75.40%	70.43%

Table 47: Caustic Soda: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE of Train Dataset	RMSE of Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear	-	85.230	24.347	779.772	28.57%	914.90%
Lasso	α = 0.00001	85.230	41.315	84.919	48.47%	99.64%
Ridge	α = 0.00001	85.230	29.137	158.519	34.19%	185.99%
RF (*)	n=25, d=4, ms=2, minl=5, maxl=4, mfw=0.1	85.230	42.572	56.895	49.95%	66.75%
KNN	n=3	85.230	9.534	100.647	11.19%	118.09%
SVM	Kernel=’poly', degree=3, C=1	85.230	74.613	78.496	87.54%	92.10%

Table 48: pH of the Boot Water: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.200	0.178	0.355	89.00%	177.50%
Lasso (*)	$\alpha = 0.00001$	0.200	0.205	0.168	102.50%	84.00%
Ridge	$\alpha = 0.00001$	0.200	0.178	0.347	89.00%	173.50%
RF	n=25, d=6, ms=5, minl=1, maxl=∞, mfw=0.1	0.200	0.169	0.183	84.50%	91.50%
KNN	n=5	0.200	0.061	0.250	30.50%	125.00%
SVM	Kernel=’poly', degree=3, C=1	0.200	0.184	0.298	92.00%	149.00%

Table 49: pH of the Boot Water: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	Best Parameters	Standard Deviation (Std. Dev.)	RMSE of Train Dataset	RMSE of Test Dataset	RMSE Train / Std.Dev.	RMSE Test / Std.Dev.
Linear	-	0.200	0.119	1.630	59.50%	815.00%
Lasso	$\alpha = 0.00001$	0.200	0.164	0.195	82.00%	97.50%
Ridge	$\alpha = 0.00001$	0.200	0.129	0.670	64.50%	335.00%
RF (*)	n=125, d=6, ms=2, minl=3, maxl=∞, mfw=0.1	0.200	0.160	0.169	80.00%	84.50%
KNN	n=5	0.200	0.048	0.229	24.00%	114.50%
SVM	Kernel=’poly', degree=3, C=1	0.200	0.182	0.258	91.00%	129.00%

Table 50: Iron Amount in the Boot Water: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.330	0.332	0.220	100.61%	66.67%
Lasso (*)	$\alpha = 0.4$	0.330	0.357	0.186	108.18%	56.36%
Ridge	$\alpha = 2$	0.330	0.332	0.220	100.61%	66.67%
RF	n=50, d=2, ms=5, minl=1, maxl=4, mfw=0.1	0.330	0.340	0.230	103.03%	69.70%
KNN	n=5	0.330	0.112	0.304	33.94%	92.12%
SVM	Kernel='poly', degree=3, C=1	0.330	0.344	0.187	113.52%	56.67%

Table 51: Iron Amount in the Boot Water: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	0.330	0.257	1.722	77.88%	521.82%
Lasso	$\alpha = 0.3$	0.330	0.317	0.210	96.06%	63.64%
Ridge	$\alpha = 5$	0.330	0.268	0.788	81.21%	238.79%
RF (*)	n=125, d=8, ms=5, minl=5, maxl=∞, mfw=0.1	0.330	0.350	0.200	106.06%	60.61%
KNN	n=5	0.330	0.116	0.284	35.15%	86.06%
SVM	Kernel='rbf', C=1	0.330	0.341	0.203	112.53%	66.99%

Table 52: Chlorine Amount in the Boot Water: The performances on the training/test datasets (without polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	4.150	3.183	2.974	76.70%	71.66%
Lasso (*)	$\alpha = 0.00001$	4.150	3.745	2.594	90.24%	62.51%
Ridge	$\alpha = 0.00001$	4.150	3.286	2.774	79.18%	66.84%
RF	n=75, d=8, ms=8, minl=3, maxl=∞, mfw=0	4.150	1.038	3.420	25.01%	82.41%
KNN	n=5	4.150	1.206	3.304	29.06%	79.61%
SVM	Kernel='poly', C=1, degree=3	4.150	3.714	2.982	89.49%	71.86%

Table 53: Chlorine Amount in the Boot Water: The performances on the training/test datasets (with polynomial expansion and train-test split performed with respect to time axis) and the best hyperparameters of applied ML algorithms

Regression Model	**Best Parameters**	**Standard Deviation (Std. Dev.)**	**RMSE of Train Dataset**	**RMSE of Test Dataset**	**RMSE Train / Std.Dev.**	**RMSE Test / Std.Dev.**
Linear	-	4.150	2.207	26.413	53.18%	636.46%
Lasso (*)	$\alpha = 0.00001$	4.150	3.105	3.013	74.82%	72.60%
Ridge	$\alpha = 0.00001$	4.150	2.293	18.379	55.25%	442.87%
RF	n=25, d=8, ms=8, minl=5, maxl=2, mfw=0	4.150	3.600	3.700	86.75%	89.16%
KNN	n=4	4.150	1.147	3.299	27.64%	79.49%
SVM	Kernel='rbf', C=1	4.150	3.647	3.611	87.88%	87.01%