Surrogate Gradient Methods for Data-Driven Scrap Procurement Optimization

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Maschinenbau und Verfahrenstechnik der

Universität Duisburg-Essen

zur Erlangung des akademischen Grades

eines

Doktors der Ingenieurwissenschaften

(Dr.-Ing.)

genehmigte Dissertation

von

Shikun Chen

aus

Fujian, China

Tag der Disputation: 26.04.2024

Erstgutachter: Prof. Dr. rer. nat. Robert Martin

Zweitgutachter: Prof. Dr. Ing. Rüdiger Deike

DOI: 10.17185/duepublico/81909
URN: urn:nbn:de:hbz:465-20240503-075110-9

Acknowledgments

Pursuing a Doctor of Engineering Sciences degree is a long and challenging journey. I couldn't have done it without the strong help and support from my teachers, family, and friends.

I'm really thankful to my main teacher, Prof. Dr. rer. nat. Robert Martin from the University of Duisburg-Essen's Chair of Math for Engineers. Throughout my research, he was always there to guide, inspire, and give great advice. His knowledge and how he supports new researchers like me can't be simply put into words.

I also want to say a big thank you to, Prof. Dr. rer. nat. Johannes Gottschling. His skills and advice were so important in making this research a success. His helpful tips and encouragement were very important in my studies. Also, a shoutout to my colleagues in the Chair of Math for Engineers, including Dr. Sebastian Tewes, Saad Alvi, Torben Disselhoff, Philine Kerst, Annika Tonnius, and Tim Kaufmann.

I also want to mention Prof. Dr. Ing. Hartmann Dierk, who was my mentor when I first started researching at Kempten University of Applied Sciences.

Lastly, a big thank you goes to my parents, Chen QingLiang and Lin LiLing, my dear sisters, and my wife, Chen XiaoHong. Their love, constant support, and understanding were always there for me during this academic journey. All the things they did to support my dream are the main reasons I could finish this thesis. I know how important their role has been in helping me grow personally and professionally. I couldn't have done it without their belief in me.

Chen Shikun
2023

Abstract

The foundry industry, a fundamental element of modern industrialization, faces significant challenges in scrap procurement and energy consumption, impacting both operational costs and environmental sustainability. This work presents a threefold approach to address these challenges, aiming to bridge the theoretical concepts of machine learning (ML) and numerical optimization with practical applications in the foundry sector. Initially, two user-friendly ML software tools, EidoData desktop and EidoData web, are developed to democratize ML applications, enabling individuals with limited ML expertise to leverage data-driven insights for informed decision-making. Furthermore, a novel surrogate gradient method is introduced to simplify the optimization process of scrap purchasing, making it more efficient and practical. This method uses smooth surrogate functions to approximate non-smooth cost functions, enhancing computational efficiency in deriving optimal scrap purchasing strategies more effectively. Lastly, an integration of the developed software tools and the optimization method is employed to simulate real-world scrap purchasing scenarios. Through a virtual environment mirroring real-world scrap trading dynamics, this integrated framework provides a powerful tool for decision-makers in the foundry industry, facilitating a detailed understanding of the complex interplay between scrap procurement, energy consumption, and overall operational cost. By employing user-friendly software, introducing a new optimization method, and demonstrating their practical application in real-world scenario simulations, the presented work contributes to the development of cost-effective and environmentally sustainable practices in the foundry industry.

Zusammenfassung

Die Gießereiindustrie, ein grundlegendes Element der modernen Industrialisierung, steht vor erheblichen Herausforderungen bei der Beschaffung von Schrott und dem Energieverbrauch, die sowohl die Betriebskosten als auch die Umweltverträglichkeit beeinflussen. Diese Arbeit stellt einen dreigliedrigen Ansatz zur Bewältigung dieser Herausforderungen vor, mit dem Ziel, die theoretischen Konzepte des maschinellen Lernens (ML) und der numerischen Optimierung mit praktischen Anwendungen im Gießereisektor zu verbinden. Zunächst werden zwei benutzerfreundliche ML-Softwaretools, EidoData desktop und EidoData web, entwickelt, um ML-Anwendungen zu demokratisieren und Personen mit begrenzten ML-Kenntnissen zu ermöglichen, datengetriebene Erkenntnisse für informierte Entscheidungen zu nutzen. Darüber hinaus wird eine neuartige Surrogate-Gradient-Methode vorgestellt, um den Optimierungsprozess des Schrottkaufs zu vereinfachen und praktikabler zu gestalten. Diese Methode erstellt glatte Ersatzfunktionen, um nicht-differenzierbare Kostenfunktionen anzunähern und die Recheneffizienz bei der Ermittlung optimaler Schrottkaufstrategien zu verbessern. Schließlich wird eine Integration der entwickelten Softwaretools und der Optimierungsmethode verwendet, um reale Szenarien für den Schrottkauf zu simulieren. Durch eine virtuelle Umgebung, die die realen Dynamiken des Schrotthandels widerspiegelt, bietet dieses integrierte Framework ein leistungsstarkes Tool für Entscheidungsträger in der Gießereiindustrie und erleichtert ein detailliertes Verständnis der komplexen Wechselwirkungen zwischen Schrottbeschaffung, Energieverbrauch und den gesamten Betriebskosten. Durch den Einsatz benutzerfreundlicher Software, die Einführung einer neuen Optimierungsmethode und die Demonstration ihrer praktischen Anwendung in Echtzeitszenariosimulationen trägt diese Arbeit zur Entwicklung kosteneffizienter und umweltverträglicher Praktiken in der Gießereiindustrie bei.

Contents

Chapter 1

Introduction

1.1 Motivation

The foundry industry, fundamentally instrumental to our contemporary living standards and the advancement of industrialization, has a far-reaching impact that affects many facets of our life. The scope of its application extends to a broad spectrum of sectors, including, but not limited to, the automotive industry, mechanical manufacturing, and agricultural equipment production, serving as a cornerstone for these integral industries.

At the core of foundry operations lies the process of metal casting, wherein metals are subjected to high temperatures until they transition into a molten state. This liquefied metal is subsequently poured into molds of varying materials, such as sand, ceramic, or metal. The molds, crafted with intricate geometric complexity, facilitate the formation of uniquely shaped parts. Following the pouring, the metal is given time to cool, leading to the solidification of the previously liquid metal. The resultant shaped metal, referred to as a casting, can be utilized in the manufacturing of products or parts exhibiting complex designs and unique shapes that would otherwise be challenging to produce.

However, the operations of foundries are not without their environmental consequences. They stand as significant contributors to environmental degradation, impacting the environment in two primary ways: through the extensive consumption of energy and the emission of harmful air pollutants. The substantial demand for energy in foundry operations and the resultant environmental implications highlight the pressing need for energy conservation and efficiency improvement. It presents a overwhelming challenge to the foundry industry, necessitating concerted efforts towards reducing energy consumption and enhancing energy efficiency.

Parallel to these operations, the foundry processes generate a voluminous amount of data. This data, when subjected to rigorous analysis and interpretation, holds the potential to unravel valuable insights. Turning this basic data into useful information is key for making good decisions. It could potentially guide optimal decision-making processes, thus playing a vital role in addressing the challenges faced by the foundry industry.

Machine learning (ML) represents a dynamic collection of algorithms that recognize patterns, describe rules, and integrate knowledge, thereby facilitating automation and optimizing the manufacturing process [4]. In data-driven applications and Artificial Intelligence (AI) services, ML algorithms help improve and often replace old rule-based methods [145]. They boost work results and make outcomes better. With more computer power and big model development, ML has made considerable progress in numerous fields, such as robotics [76], healthcare [57], and autonomous driving [22].

On the other hand, using ML in the foundry industry is a complex and challenging task. This is mainly because of several issues [139, 155]:

- Many foundries have been collecting data without a clear strategy over the years. The common method starts with a plan to link many data sources and then pull out new insights. Buying decisions are often based on whether equipment can be connected or fitted with a sensor. As a result, data is collected without much thought and organizing it becomes difficult. This leads to a lot of data being collected, but most of it isn't used because it's hard to combine and understand.

- Secondly, the data collection process often suffers from the negative effects of unstructured methods or spatial constraints. This frequently leads to the early termination or overwriting of the data collection process, or worse, the incorrect recording of information. The data collected in these conditions often lacks usefulness for later steps, making it mostly useless.

- Third, to make an effective ML solution work, several steps need to be completed correctly. This includes data preprocessing, feature engineering, model selection, hyperparameter tuning, model evaluation, and model deployment. Absent a robust understanding of this complicated ML workflow, it becomes a near impossible task to leverage ML effectively within the foundry industry.

To overcome these challenges, the foundry industry must adopt a more structured approach to data collection and management, invest in building ML expertise, and develop a comprehensive understanding of the ML workflow. This can open up new avenues for optimizing operations and driving innovation in the industry.

The production of high-quality castings necessitates the investment of multiple resources, each of which acquires a distinct cost. The primary cost determinant in this process is typically the expenditure related to the procurement of materials. The material cost is inherently dependent on the price at which scrap metal is purchased from external sources for integration into the casting process.

The cost per kilogram of castable metal, or the unit spout cost, is consequently a cumulative measure of multiple constituent costs. These include the aforementioned material cost (which is subject to the cost of scrap), the energy cost required for the operation of casting machinery, the demand for electricity which is a significant factor in determining overall energy cost, the cost of labor in terms of wages for employees engaged in the casting process, and the cost of consumable material that is expended in the production process.

To explain further, the cost of the material is influenced by the fluctuations in the global metal market, availability of quality scrap metal, and the costs associated with transporting and storing the scrap metal. The energy cost accounts for the expenditure on power required for melting the scrap metal, maintaining the operation of the casting machinery, and other associated processes.

Consequently, it is crucial to optimize the procurement of scrap material while simultaneously trying to reduce the overall cost. The consumption of energy, a significant component of the overall cost structure, can be predicted through the application of trained ML models. These ML models primarily utilize the quantity of scrap material and specific furnace parameters as input features, effectively mapping the relationship between scrap quantity and energy usage.

Numerical optimization techniques can play a pivotal role in fine-tuning this relationship. By employing these mathematical strategies, it becomes possible to devise an optimal plan for scrap procurement. This can potentially lead to a reduction in energy consumption, thereby contributing to cost reduction. Importantly, this optimization process must be carefully managed to ensure that the quality of the final production is not compromised. To elaborate, numerical

optimization techniques can be used to find the optimal selection of scrap that minimizes energy consumption. These techniques can explore the solution space defined by the ML models and find the point that yields the lowest energy consumption.

However, the complication associated with scrap optimization is a direct consequence of the numerous of process parameters, control parameters, and control variables. Furthermore, it is challenging to optimize data-driven models to address such complex issues, in large part because the majority of existing models exhibit strong irregularity, which hinders the application of conventional optimization techniques, such as gradient descent. For instance, ML models that are frequently utilized, like gradient boosting and random forests, may not possess differentiability, and might even show discontinuities.

Since traditional optimization methods, such as gradient descent, operate on the assumption of smooth, differentiable functions, and leverage the gradient or derivative information to iteratively reach the function's minimum, they are not immediately applicable to such ML models. This dilemma necessitates the use of alternative strategies, specifically derivative-free optimization techniques. These techniques do not rely on derivative information but rather employ direct search, stochastic, or evolutionary strategies to find the optimal solution. By doing so, derivative-free optimization techniques offer a viable route to tackle the irregularities and non-differentiability inherent in many modern machine learning models, thereby enabling the optimization of these models for complex tasks such as scrap optimization. For many applications, however, the performance offered by derivative-free optimization is simply insufficient.

Overall, the main problem is to optimize the total cost of the foundry process, with a particular focus on reducing energy consumption, which is a significant factor in the cost structure. The total cost function, denoted as $f(x) = p(x) + m(x) + l(x)$, depends on the scrap procurement strategy x and is composed of three main components:

1. Scrap purchasing cost ($p(x)$): The cost incurred in acquiring scrap metal, which is a key raw material for the casting process.

2. Energy consumption cost ($m(x)$): This cost is directly influenced by the amount and type of scrap metal used. The energy consumption for melting and processing the scrap is modeled using a machine learning approach, specifically a gradient boosting model, which is inherently non-differentiable.

3. Transport cost ($l(x)$): The expenses associated with transporting the scrap metal to the foundry.

The challenge lies in the fact that the energy consumption component of the cost function ($m(x)$) is predicted by a non-differentiable ML model (gradient boosting). This characteristic of the model complicates the application of traditional optimization methods that rely on derivatives, such as gradient descent.

The research aims to investigate the effectiveness of surrogate optimization as an alternative to derivative-free optimization techniques in this context. Surrogate optimization involves creating a differentiable model that approximates the behavior of the non-differentiable ML model. This surrogate model can then be optimized using traditional methods. The investigation will assess whether this approach offers advantages, particularly in terms of accuracy and efficiency, over standard derivative-free methods in optimizing the total cost function in the foundry industry, considering the constraints and characteristics of the ML model used.

1.2 Goals and contributions

A major contribution of this thesis is the systematic investigation of differentiable surrogates functions that serve as viable approximations for non-smooth functions, preserving their core properties while enhancing their tractability for optimization purposes.

Differentiable surrogates act as mathematical stand-ins for the original non-smooth functions in the optimization process. This surrogate-based method simplifies the optimization, allowing for the application of conventional optimization algorithms, which are typically well-suited for smooth, differentiable functions.

The steps in the optimization process using differentiable surrogates are as follows: Firstly, a differentiable surrogate function that approximates the original non-smooth function is constructed. This surrogate function is then subjected to optimization utilizing standard algorithms designed for differentiable functions. The optimized surrogate function subsequently serves as a means to extract an approximate solution for the initial non-smooth function.

While these surrogate functions might not perfectly represent the actual process due to the inherent simplifications introduced during their construction, they provide substantial benefits when derivative-based optimization techniques are employed. The primary advantage is the marked improvement in computational efficiency, as these techniques generally perform better with smooth, differentiable functions. This advantage can significantly reduce the time and computational resources required for the optimization process, thus making the surrogate-based approach a practical choice for large-scale or complex optimization problems.

Based on the aforementioned methodology, a specialized simulation software has been developed. This software is designed to encapsulate the comprehensive process of scrap trading, specifically from the viewpoint of the foundry industry. It seeks to provide a virtual environment that closely mirrors the complexities and dynamics of real-world scrap trading.

This simulation system capitalizes on the solution provided by differentiable surrogate models to ascertain the optimal combination of scrap purchases. The optimality here refers to minimizing the total cost incurred in the scrap procurement process. By utilizing differentiable surrogates, the software can efficiently navigate the search space of possible purchasing combinations. It does so by approximating the cost function, which might be non-smooth due to the employment of ML models such as gradient boosting, with a smooth surrogate function. This surrogate function can then be optimized using standard optimization algorithms, resulting in an efficient solution for the original cost function.

The incorporation of differentiable surrogates into the simulation software allows for a streamlined and efficient approach to the complex task of scrap procurement optimization. The software can handle a variety of scenarios and constraints, adapting to changes in the market conditions, availability of scrap types, and other variables. This makes it a valuable tool for decision-makers in the foundry industry, enabling them to make informed and cost-effective decisions about scrap procurement.

Together with the development of the simulation software, two additional ML software tools have been created to speed up the process of ML model training and deployment. These tools are designed to lower the barriers to entry for ML applications in the foundry industry, enabling non-experts to leverage the power of ML for their specific needs.

The first tool is the EidoData desktop version, which is designed with user-friendliness at its core. EidoData desktop offers a straightforward graphical user interface (GUI) that simplifies the process of ML for non-ML experts. Users can easily import data for ML training and preprocessing. It extends a wide array of state-of-the-art ML algorithms for users to select from, thereby offering flexibility and customization based on the specific task at hand. EidoData desktop has been designed with a focus on accessibility and ease of use, allowing users without a

deep understanding of ML to train and deploy ML models effectively. This tool has the potential to accelerate the adoption of ML solutions in the foundry industry, as it bridges the gap between complex ML algorithms and practical industry applications.

In addition to the desktop version, an EidoData web version has been developed to provide even greater accessibility. This web-based software eliminates the need for local installation, allowing users to access ML services directly through their browser from anywhere with internet access. The EidoData web version further streamlines the ML process by providing Automated Machine Learning (AutoML) services. AutoML represents a significant advancement in the democratization of ML, as it enables users with limited ML expertise to train high-quality models that are tailored to their specific needs. Through the use of AutoML, the EidoData web version automates various stages of the ML process, including feature selection, model selection, and hyperparameter tuning. This allows users to focus more on their specific business problems and less on the intricacies of ML model development.

These software tools—EidoData desktop and EidoData web—represent significant strides in bringing ML solutions to the foundry industry. By making ML more accessible and user-friendly, they have the potential to significantly impact the industry, enabling businesses to harness the power of ML for their specific needs.

In conclusion, in conjunction with numerical optimization, these programs can serve as powerful tools for decision-making and strategy formulation in the foundry industry. They enable a more detailed understanding of the complex interplay between scrap procurement, energy consumption, and production quality, leading to more efficient and cost-effective operations.

1.3 Thesis structure

This thesis addresses challenges in the foundry industry, emphasizing the optimization of scrap procurement and energy consumption using ML and numerical optimization. Chapter 2 delves into the foundry and metal recycling sectors, pinpointing the critical challenge of scrap purchasing optimization and introducing ML and numerical optimization as potential solutions. Chapter 3 presents a scrap procurement platform based on differentiable surrogates ML models, simplifying the optimization process for cost-effective scrap purchasing strategies. In chapter 4, two software tools, EidoData Desktop and EidoData Web, are introduced which make ML accessible to individuals in the foundry industry, facilitating data import, ML algorithm selection, and model deployment. Chapter 5 explains the mechanics of differentiable surrogates optimization, validating its effectiveness through numerical experiments and paving the way for real-world applications. Chapter 6 extends the validation to real-world scenarios, showcasing how the optimization method simulates real-world scrap purchasing behavior, thus providing a practical tool for decision-makers in the foundry industry. The final chapter 7 concludes the thesis, summarizing the contributions and discussing future work, highlighting the potential for more efficient and environmentally responsible operations in the industry through the developed methods and tools.

Chapter 2

Background

Climate policies and sustainability requirements have an immense effect on the current industrial landscape. The most visible aspect of this shift is the requirement to reduce atmospheric carbon dioxide (CO2) emissions, a major contributor to global climate change. Considering that metal castings form the backbone of approximately 90% of all manufactured goods [144], this development has profound implications for the foundry industry. This suggests that foundries have a growing obligation to improve the sustainability of their operations, laying the groundwork for the manufacturing industry's sustainable growth. The foundry industry has historically made significant contributions to sustainability, particularly through the recycling of scrap iron and steel [68]. However, when it comes to energy efficiency and resource utilization, there is adequate scope for further enhancements.

The following introduction provides some basic background of the foundry industry and related subjects. Initially, sections 2.1 and 2.2 explain the foundational aspects of foundry operations and metal recycling, respectively, laying the groundwork for the subsequent discussion. Section 2.3 then depicts the complexities involved in scrap procurement. Afterwards, section 2.4 introduces the underlying principles of machine learning (ML), with a particular emphasis on supervised learning and AutoML. Subsequently, section 2.5 describes the core principles of the numerical optimization methods used in this study. Finally, section 2.6 provides a thorough examination of cloud computing concepts and their applications relevant to this research.

2.1 Foundations of the metal casting process

A foundry is an industrial setting created exclusively to produce metal castings, which are created by melting a certain metal, pouring the molten metal into a precisely designed mold, and then enabling the solidified metal to take on a brand-new, custom geometric form. In case of iron and steel casting, the molds are typically made out of sand, binder and other additives. This intricate process, known as foundry work, results in the production of a casting through a complex, multi-stage workflow, which includes pattern-making, molding, melting, pouring, solidification, removal of the casting from the mould, cleaning, fettling, and inspection. This process sequence is depicted in Figure 2.1.

The eventual shape of the casting is a direct replica of the mold cavity into which the liquid metal was poured, thus necessitating the careful crafting of the mold. Molds are created based on a pattern, which is a replica of the object to be cast, typically constructed from materials such as wood or metal. The production of a precise pattern forms an integral step in the casting process,

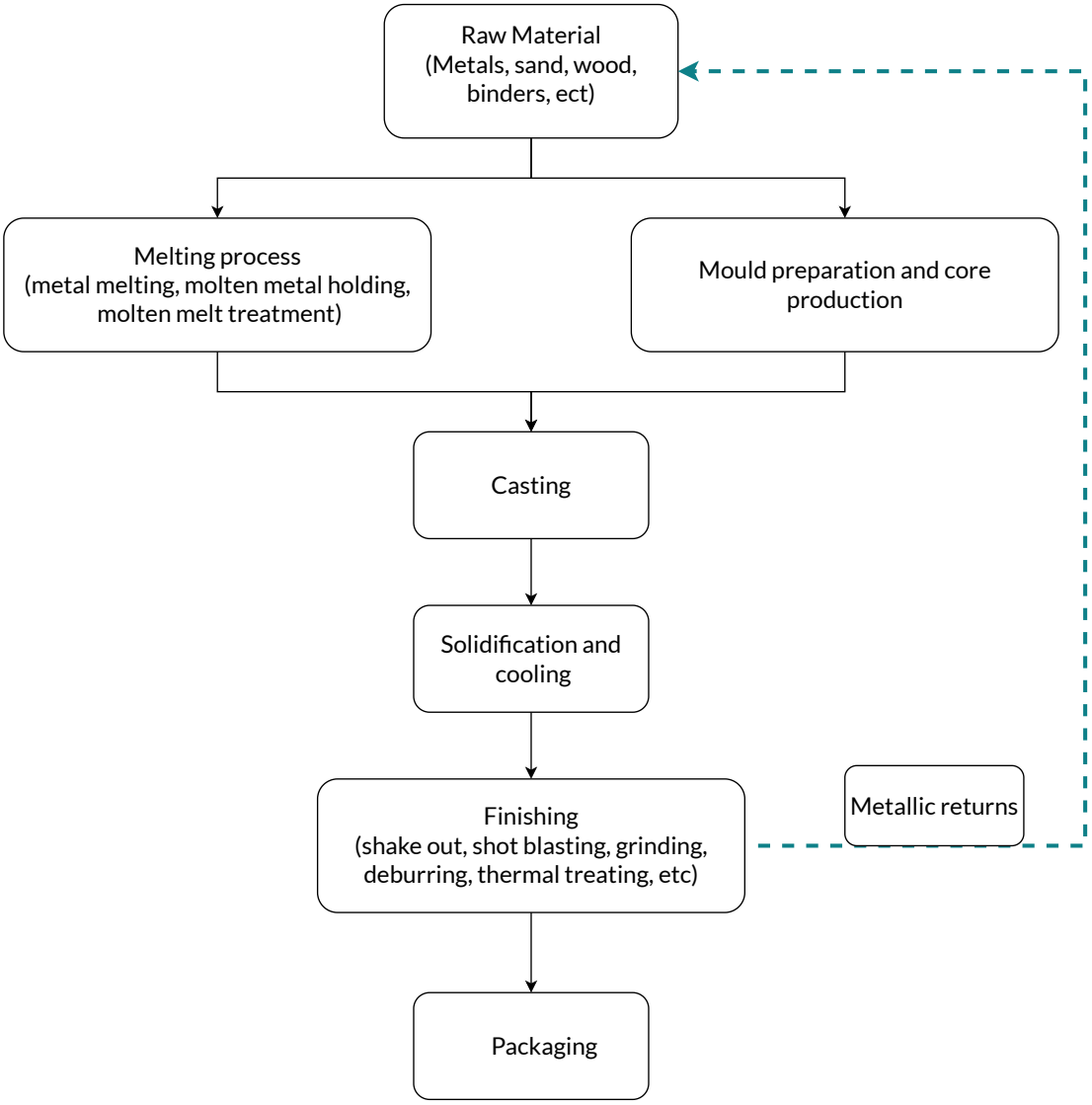

Figure 2.1: Process flow diagram of a foundry industry.

as it sets the foundation for the subsequent manufacturing steps and ultimately determines the final form of the casting.

Predominantly, silica sand is used as the material for molds, owing to its high refractory properties and cost-effectiveness. However, the choice of mold material is not limited to silica sand and can be varied based on several factors including the type of metal being cast, the specific casting method employed, and the unique requirements of the casting process. For example, in certain high-precision casting methods, alternative materials such as ceramics or metals might be used for the construction of molds. The decision regarding the selection of an appropriate mold material plays a critical role in the foundry process and has significant implications for the quality of the final casting, process efficiency, and overall economic viability of the operation.

Metal melting is the key procedure in the foundry and is frequently referred to as the core of the business. This procedure turns solid metals into liquids by applying heat above the melting point of the metal in a machine fittingly called the melting furnace. The process of melting metal is characterized by several key steps, including the determination of metal mixture ratio, preparation and loading of the metal, melting, refining, and treatment of the molten metal, and the transportation of molten metal [30].

Typically, the metal mixture, comprising predominantly of ingots and scrap metal, is carefully calculated to yield the desired mechanical properties of the final product. This mixture often contains additives such as alloys of Zinc (Zn), Manganese (Mn), Copper (Cu), Silicon (Si), etc., to enhance certain properties [116]. The furnace is charged with this mixture and a fuel source, which could be charcoal, natural gas, or electricity. The metal is loaded continuously throughout the heating process to optimize energy consumption and work efficiency [115].

Once the molten metal has reached a certain temperature, it is transported from the melting furnace to the moulding line, where it is manually or automatically poured into the moulds. The metal then cools and solidifies, and the casting is removed from the mold and cleaned [37]. The entire process significantly influences the physical and chemical characteristics of the final casting products and consumes a considerable amount of energy, accounting for about 55% of the energy consumption in the metalworking industry[37].

Foundries are generally distinguished by whether they work with ferrous or non-ferrous metals. Ferrous metals contain iron, while non-ferrous metals do not. The global production of metal is dominated by ferrous metallurgy, with gray iron being the most commonly cast metal in foundries [13]. The exact type of furnace utilized in these foundries is contingent on the metal they work with. For instance, electric arc furnaces (EAF) are optimal for steel processing, while foundries specializing in copper are more likely to employ an induction furnace [101].

In addition to the type of metal being melted, the choice of melting furnace technology can also influence the process. Traditional methods include the use of crucible and cupola furnaces, while modern advanced melting technologies employ induction and EAF. These different technologies have a direct impact on the effectiveness of the melting process.

Foundries indeed leverage metal recycling as a sustainable and cost-effective source of raw material. The inherent ability of metals, including but not limited to steel, iron, aluminum, and copper, to withstand recycling cycles without loss of metallic properties makes them ideal candidates for repeated melting and casting.

Broadly, foundries purchase raw materials from two main sources. Firstly, there is home scrap, or Kreislaufschrott in German, which includes trimmings, pigged metal, and rejects originating from mill or foundry production. The known chemical properties of this type of scrap allow for easy integration with other raw materials and reprocessing on site. Secondly, foundries acquire external scrap, or Fremdschrott in German, from the scrap metal recycling industry. This industry refines both obsolete and prompt scrap into commodity-grade material. Obsolete scrap, collected from diverse sources such as scrap metal dealers, auto salvage facilities, and industrial manufacturers, is typically processed at large-scale recycling facilities. Prompt scrap, on the other hand, is generated during the fabrication of metal products and, akin to home scrap, is quickly returned to the mill for reprocessing due to its known chemical properties.

Following procurement, the scrap metal is classified into ferrous, non-ferrous, and non-metallic materials. This necessitates shredding the scrap metal into smaller pieces for efficient and thorough separation.

The core challenge of this research is the optimization of the charge material for melting. One of the most crucial steps in preparing for the production of liquid casting alloys in foundry furnaces is the burden calculation. These calculations, often complex due to the multitude of input materials and chemical elements, are typically realized through numerical methods imple-

mented in spreadsheets, universal mathematical programs, or specialized programs for foundry engineering [144].

2.2 Metal recycling

Globally, each day witnesses the generation of substantial volumes of waste materials, often in the guise of scrap from production processes or demolition activities [21]. Despite first appearances, this waste material is actually a resource of great worth that can be repeatedly reintroduced into the material economy without losing any of its quality. Within the realm of recycling, a distinction is drawn among various categories of metals, namely industrial metals, minor metals, and noble metals [66].

Industrial metals, owing to the sheer volume involved in global trade, command a position of exceptional importance. This category encompasses a wide range of metals, including iron (Fe), aluminum (Al), copper, zinc, nickel (Ni), and tin (Sn), among others. These metals are often used in large-scale industrial applications, such as construction, manufacturing, and transportation, and their recycling plays a critical role in resource conservation and sustainable development.

Minor metals, on the other hand, constitute a group that includes elements traded in quantities significantly lower than industrial metals. Despite their relatively smaller trade volumes, these metals often command much higher prices, attributable to their specialized applications and often limited supply. Minor metals are frequently used as alloying elements. An alloy is a mixture of two or more chemical elements, at least one of which is metal. For instance, steel is compound of iron, and carbon (C) and other elements. Important alloying elements in steel and minor metals are manganese, silicon, chromium (Cr), molybdenum (Mo), tungsten (W), vanadium (V), cobalt (Co), titanium (Ti), niobium (Nb), boron (B) and bismuth (Bi).

Noble metals, such as gold (Au), silver (Ag), and platinum (Pt), represent a distinct group, characterized by their resistance to corrosion and oxidation. While their use in jewelry is well-known, they also find important applications in various industries, such as electronics, catalysis, and medicine [103]. The recycling of noble metals is particularly valuable, given their high market prices and significant environmental impact associated with their mining and refining.

The first industrial metal to be highlighted in this discussion is iron, a versatile element that serves as the foundation for two main materials: steel and cast iron, each of which has unique properties. Steel is a general term for a variety of alloys that are mostly made of iron, displaying high deformability, and have a maximum carbon content of 2.06% [111]. This highlights the complex interaction between iron and carbon, which results in the production of an alloy with special qualities that make it a vital component in several industrial applications.

As per the statistical data released by World Steel Association [160], the global production of crude steel was estimated to be 1,878 million metric tons in the year 2020. Focusing on Germany, a significant player in the global steel industry, it is noteworthy that the nation produced 35.7 million metric tons of crude steel in the same year. Interestingly, the production process incorporated 19.76 million tons of scrap, thereby indicating that, on average, each ton of steel was constituted by approximately 55.3% of recycled material [14]. This suggests a considerable level of metal recycling, effectively reintegrating iron back into the material cycle in the form of scrap.

Additionally, German steel melt shop reportedly procured an estimated 12.55 million tons of scrap during the year 2020 [14]. The part of the discrepancy of procured and utilized scrap can be attributed to the generation of scrap within the steel mill itself, thereby negating the necessity for external procurement. This observation provides valuable insights into the integral role of recycling within the industrial operations, contributing to resource efficiency and sustainability.

In the domain of industrial metals, aluminum manifests a significant presence, occupying the second rank in terms of volume. There are two unique routes that the aluminum manufacturing process takes: the production of primary aluminum, derived directly from mined ore, and the production of recycled aluminum, a category also known as secondary aluminum. In the year 2020, German aluminum industry's production data revealed an output of 1.1 million tons of raw aluminum, a decrease by ten percent compared to the preceding year [93]. This decrease underscores the cyclicality and responsiveness of metal production to the prevailing economic conditions and market demands.

Upon dissecting the composition of raw aluminum production, it was found that 529,000 tons constituted primary aluminum, while 548,000 tons were contributed by recycled aluminum. This data implies that in the year 2020, secondary aluminum accounted for approximately 51 percent of the total raw aluminum output, hinting at the considerable role of recycling in the aluminum industry [93]. This proportion not only reflects the industry's emphasis on sustainable practices, but also underlines the economical and environmental benefits derived from metal recycling.

In the hierarchy of industrial metals in Germany, copper is positioned alongside iron and aluminum as one of the top three metals, highlighting its critical importance in the industrial sphere. The data for the year 2020 reveals that Germany produced 639,000 metric tons of refined copper and copper alloys [93].This fact not only highlights how important copper is to the German industrial environment, but it also illustrates how important Germany is to the worldwide copper supply chain. It is important to consider these figures in light of the various applications of copper in diverse industries, which include but are not limited to electrical wiring, construction, and electronics. Such extensive utilization of copper underlines the significance of its production data, as it provides valuable insights into the state of these industries and the economy at large .

Within the industrial sector of Germany, each transaction involving scrap metal is subject to individual negotiation between the scrap dealer and the procurement officer of the steel mill. This process of price determination, however, is typically not as arbitrary as it may seem, being influenced by an array of diverse factors. On a broad level, scrap metal, irrespective of its composition, can be categorized into three distinct groups.

The first category, commonly referred to as home scrap, comprises waste material that is generated directly within the industrial plant or smelter. This scrap is typically an offshoot of the manufacturing process itself, a byproduct of the primary production activities. Such scrap can often be reintroduced into the production cycle, given its relatively uncontaminated state and proximity to the manufacturing process. The second category, often termed new scrap or post-production scrap, is the waste material that accumulates in industries involved in processing metal. This category encompasses the scrap generated during the shaping, forming, and finishing of metal products, for instance, the offcuts, shavings, and trimmings that result from machining and fabricating operations. The third and final category, designated as collective scrap, is produced after the end of use of goods. This category includes a wide range of metal garbage, from outdated infrastructure and construction rubbish to abandoned consumer goods and worn-out machinery. Unlike the other two categories, collective scrap is typically more heterogeneous and contaminated, necessitating additional processing before it can be recycled.

These categorizations are crucial for understanding the diverse origins and properties of scrap metal, providing a framework for its evaluation and pricing. The German Steel Association (Wirtschaftsvereinigung Stahl WV Steel) [148], for instance, releases the average purchase prices of steel scrap delivered to the plants on a monthly basis, classified into seven distinct grades. The basis for pricing between foundries and scrap traders often employs the price publication of WV Steel for grade 2, with an additional surcharge [148].

Usually, the extra for particular grades results from special delivery requirements set by the

foundries. These requirements may limit the types or sizes of scrap that are accepted, demand that the scrap be delivered completely dry, or specify the analytical requirements that the scrap must meet. As a result, the purity of the scrap as well as its physical and analytical composition determine its value to a certain consumer and, consequently, the compensation the scrap dealer can demand. However, the price of scrap is not solely determined by its intrinsic value but also by a host of external factors. These include the costs associated with logistics, such as transportation and handling, the costs related to financing, and the costs incurred in storing the scrap. Thus, the price of scrap metal is a complex interplay of its inherent properties, the specific requirements of the customer, and the various costs involved in its collection, storage, and delivery.

The successful introduction and sustenance of a product in the commercial marketplace necessitates the deployment of cost-effective manufacturing strategies. The selection of an appropriate manufacturing process is therefore critically influenced by considerations of economic viability. For a myriad of commodities, the expense associated with procuring raw materials constitutes a substantial proportion of the overall production costs. The concept of recycling gains prominence in this context, wherein the transformation of waste or discarded materials into secondary raw materials presents a cost-efficient alternative, provided the newly generated materials match, or closely approximate, the mechanical and analytical properties of primary raw materials.

The operational flexibility of an industrial installation in terms of raw material utilization directly impacts the procurement of resources and scrap. This connection is best illustrated by the oxygen steelmaking process used in the manufacture of steel. In the converter, scrap is combined with pig iron, which is obtained from a blast furnace and is characterised by its relatively high purity. The presence of accompanying materials is diminished as a result of the addition of scrap. However, it is crucial to note that steel produced via the oxygen steelmaking process necessitates strict adherence to defined limits on trace element contents [153].

Conversely, the electric steelmaking route, characterized by the capacity to utilize up to one hundred percent scrap, dispenses with the dilution effect associated with pig iron [26]. This process, however, restricts the potential use of oxygen solely to the metallurgical stages within the furnace, given the absence of a distinct refining process analogous to the converter. Subsequent metallurgical operations are conducted in the realm of secondary metallurgy.

Despite the inherent flexibility offered by the electric steelmaking process, particularly with respect to the precise regulation of chemical composition through the coordination of furnace and secondary metallurgy, the fabrication of specific steel grades necessitates the targeted selection of appropriate high-quality scrap grades. This selection may be complemented by potential dilution effects achieved through the use of solid, pure pig iron or sponge iron. Therefore, the strategic selection and utilization of raw materials and manufacturing processes, underpinned by the principles of recycling and resource efficiency, play an instrumental role in optimizing the economic viability of product manufacturing.

Contrarily, the foundry, an entity engaged primarily in the heating of raw materials, possesses a comparatively lower degree of flexibility due to its inherent limitations in the removal of disruptive elements [6]. This limitation is largely attributable to the absence of dedicated metallurgical procedures designed specifically to attenuate trace element concentrations during the melting operations. In the contemporary manufacturing landscape, foundries employ a diverse spectrum of furnace technologies, each tailored to meet specific requirements of the casting process [29]. The primary goal of these foundries is to purchase furnaces that can enable the manufacture of a wide range of metal alloys and additives, providing a variety of casting characteristics to satisfy a wide range of industrial uses.

In the context of prevalent furnace technologies, induction furnaces and cupola furnaces are the predominant choices for foundries [102]. Induction furnaces, leveraging the principles of elec-

tromagnetic induction, harness the power of alternating electric currents to attain the requisite melting temperatures for various metals. These furnaces have gained substantial traction in the foundry sector owing to their high operational quality, intuitive control mechanisms, and energy efficiency. A noteworthy feature of induction furnaces is their adaptability across wide operational scales, possessing the capability to melt quantities from less than 1 kilogram to volumes scaling up to 100 tons, thus accommodating the diverse requirements of different foundries.

On the other hand, cupola furnaces represent a more traditional approach to metal melting, having been an integral part of foundry operations for a considerable length of time. These furnaces are characterized by their towering cylindrical chimneys, which are lined with protective materials such as clay, bricks, and blocks. This lining is essential to shield the interior of the furnace from the intense thermal energy, abrasion, and oxidation that occur during the melting process. The operational methodology of cupola furnaces entails the placement of several layers of ferroalloys, coke, and limestone in the furnace prior to the addition of the metal [157]. This layered configuration initiates a series of chemical reactions which facilitate the segregation of impurities within the furnace, resulting in the impurities ascending to the surface of the molten metal.

The elemental makeup of the scrap, particularly the inclusion of noble elements like copper and nickel, has a considerable impact on the value proposition of scrap utilization in the context of steel manufacturing [104]. The metallurgical techniques implemented within the steel mill are sufficiently resilient to allow a certain degree of compositional diversity, hence permitting a certain amount of analytical flexibility, provided that the scrap is not contaminated with excessive concentrations of these components. This capability is, in part, a function of the comprehensive processing systems deployed within these facilities, which can effectively manage a diverse range of input materials.

This operational flexibility extends to the procurement of scrap material. Given the large-scale nature of steel mill operations, there is an increased capacity to integrate varying scrap dimensions and types within the production process. As a consequence, steel mill purchasers have a broader market from which to source scrap, often translating into more competitive procurement prices.

Contrastingly, purchasers associated with foundries often encounter more strict constraints in their scrap procurement activities. Owing to the smaller scale of foundry operations, the ability to process diverse scrap dimensions is inherently limited. This necessitates the procurement of scrap that aligns with specific dimensional parameters.

Furthermore, the nature of foundry operations, particularly those utilizing induction furnaces, necessitates that procured scrap is devoid of moisture. This requirement, driven by occupational safety considerations, imposes an additional constraint on the types of scrap that can be used within the foundry environment. Consequently, the confluence of these factors – a need for dimensionally specific, dry scrap often results in foundries incurring higher scrap procurement costs relative to their steel mill counterparts.

Scrap metal smelting has potential for sustainability, but the practical implementation is complex due to cost and logistical challenges. Costs arise from handling and segregating scrap metal, ensuring its quality and composition, and storing different types of scrap. However, these costs can be offset by savings from using scrap metal containing desired alloy elements, reducing the need for additional elements during smelting. The economic viability of scrap-based smelting depends on the elemental composition of the scrap metal, with the value of recovered elements needing to outweigh handling and storage costs.

The melting rate in an EAF, or tap-to-tap times, is a key parameter influenced by factors like the density of the raw material [153]. High packing density scrap offers advantages, but other factors like market rates for scrap grades and logistics also play a role. Logistics, includ-

ing transportation distance and mode, are crucial. Within Germany, trucks are preferred for distances under 100 kilometers, while maritime transport is cost-effective for larger batch sizes, given suitable transshipment facilities. Thus, the efficiency of the EAF operation is tied to multiple factors, from scrap metal characteristics to logistics, impacting overall productivity and profitability.

2.3 Challenges of scrap procurement optimization

Most foundries use some sort of scrap mix optimization tool. The goal is to select the most cost-effective scrap mix for each production grade while maintaining steel quality within specified limitations. The complexity of these optimization programs varies by plant. The easiest option is to use a few standard scrap recipes that are based on standard purchase prices for scrap grades, volume restrictions in the scrap buckets and furnace, and quality requirements for the chemical analysis of the liquid steel. To prevent cave-ins, electrode breakage, and damage to the furnace lining and wall panels during charging, the placement of the various scrap grades in the baskets is often also taken into consideration.

A more complex solution is to choose the standard scrap mixtures while taking the energy consumption and metallic yield into account. The "value in use" for each scrap grade must then be determined by considering the precise energy consumption and yield coefficients of the scrap grades. The arrangement of the scrap grades in the furnace also has an impact on the effectiveness of various energy inputs (such as electrodes, burners, and lances).

Utilizing "dynamic" scrap recipes might make the scrap optimization more complicated. Then the parameters in the optimizer, (like cost, density, chemical composition, specific energy consumption and yield) are not constant, but they are updated on a regular basis. The settings of the majority of scrap optimizing systems are updated when specifications from scrap suppliers reveal altered scrap characteristics. When a new delivery of scrap comes, melting tests may be performed to estimate the scrap's qualities.

The price and quality of steel scrap fluctuates and therefore requires regular determination of the relative usage rate for each batch of steel produced in a given period. The determination of the utilization rate leads to an optimization problem aimed at minimizing the scrap purchase cost. The solution to the programming problem should specify from which scrap supplier, which type of scrap, and in which batches to use with scrap in order to fulfill customer orders and maintain desired inventory levels for the steel producer [34].

Important feed stocks for the manufacturing of steel include iron ore and scrap steel. Various suppliers sell scrap steel on the open market as a commodity. To guarantee that overall production and quality goals are reached, the scrap commodity is periodically acquired. In order to create steel that is cast into solid forms and subsequently rolled, processed, and supplied to consumers, scrap steel is typically processed in EAF, where it is combined and melted batch by batch [87, 129, 56]. The most vital source of feed stock for an EAF is purchased scrap steel. Costs associated with manufacturing, purchases, and electrical energy are all heavily impacted. To produce the appropriate physical and chemical qualities of the completed product to meet customer requirements, different types of scrap steel can be mixed in various ratios.

The availability from each scrap provider, the market price, and the quantity of trace metals like copper, tin, sulphur, and phosphorus all have an influence on how economically recovered steel scrap may be used. To guarantee that the steel material qualities are consistent and to satisfy customer criteria like weldability and hardness, it is crucial to control the amount of these elements. The quantity of scrap steel utilized to make each batch of steel must occasionally be adjusted due to the volatility in the price and quality of scrap steel. With the use of a

magnet, the metal is charged by being lifted from particular heaps and placed in a bucket before being sent to the EAF. To consolidate similar qualities and residual metal content, the scrap piles are segregated [96]. To guarantee that the scrap fed into the EAF is of the expected grade, manufacturing processes periodically test the scrap quality. These tests are based on measurements of the product steel quality and on analyses of the scrap feed stocks. The quality constraints are satisfied thanks to this procedure.

2.4 Machine learning

Machine Learning (ML) focuses on the use of data and algorithms to emulate the way that humans learn, gradually improving accuracy [72]. ML is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within application to expand and grow [165]. Within data science, ML can be divided into three primary categories. Supervised learning is defined by its use of labeled datasets to train algorithms that classify data or predict outcomes accurately [18]. Unsupervised learning uses ML algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data grouping without the need for human intervention [8]. Semi-supervised learning offers an appropriate medium between supervised and unsupervised learning; during training, it uses a smaller labeled dataset to guide classification and feature extraction from a larger, unlabeled dataset [166]. Semi-supervised learning can solve the problem of having not enough labeled data to train a supervised learning algorithm.

This work focuses on the supervised regression problem, although many of the contents also apply for the general field of ML. It is important to understand the basics of ML and its typical challenges in order to be able to design practically useful and robust systems. We begin with a description of supervised learning, which includes an explanation of workflow involved in supervised learning. Next, some ML algorithms implemented in this work will be explained. Finally, an overview of AutoML will be presented.

2.4.1 Supervised regression learning

Supervised learning occurs when a ML model learns from sample data and associated target response that can consist of numeric values or string labels, such as classes or tags, to later predict the correct response when posed with new examples [100]. Supervised learning is frequently applied in applications where historical data predict likely future event. Generally, supervised learning problems are categorized into classification and regression. In regression problem, the model tries to predict results within a continuous output based on the continuous functions. For example, it can anticipate the price of a house according to the size and the living area. In contrast to regression, in classification outputs is predicted in discrete value such as yes or no, true or false, positive or negative, etc. In this work, we will focus on regression problem. For a regression problem, a feature vector X contains m components, i.e.

$$X = [x_1, x_2, \ldots, x_m] \in \mathbb{R}^m \tag{2.4.1}$$

If feature vectors appear in a list or dataset, they are usually denoted with an index $x^{(i)}$. In this case the j-th scalar component of the i-th feature vector is written as $x_j^{(i)}$. By y we denote the output vector, and the goal is to find a function

$$f : X \mapsto y \in \mathbb{R} \tag{2.4.2}$$

which is considered as the *regression model*. This model is adapted to the learning task by training data in form of a ground truth dataset $\mathbb{D} \subset \mathbb{R}^{(n+1)\times n}$ of n labeled feature vectors:

$$\mathbb{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\} \tag{2.4.3}$$

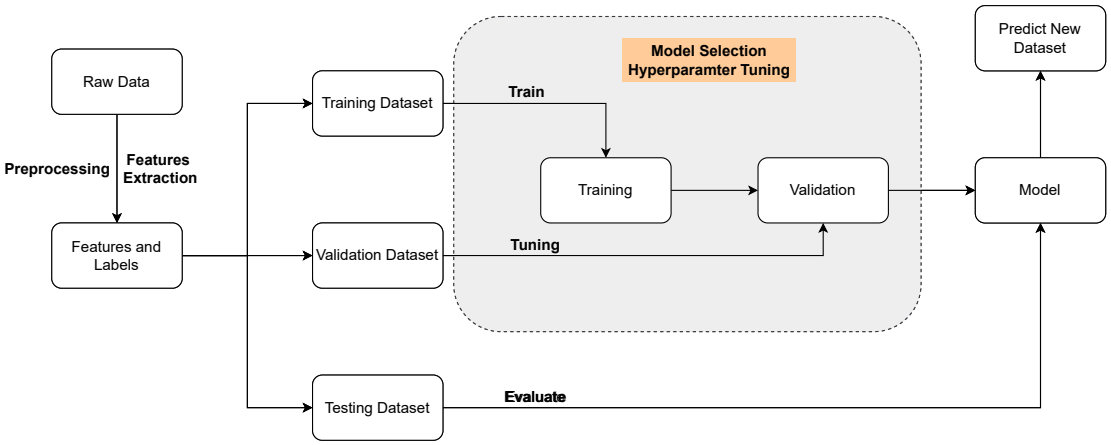

Figure 2.2: Machine Learning workflow

ML workflows define which phases are implemented during a ML project. The typical phases include data collection, building dataset, data preprocessing, model training and selection, evaluation, and deployment to production, as shown in Figure 2.2.

2.4.1.1 Gathering data

Gathering data is one of the most important stages of ML workflows. During data collection, the potential usefulness and accuracy of the project are determined by the quality of the data gathered. To accumulate data, it is essential to identify the sources and aggregate data from these sources into a unified dataset. This might involve streaming data from Internet of Things sensors, acquiring open-source datasets, or constructing a data lake from diverse files, logs, or media.

2.4.1.2 Data preprocessing

Data preprocessing in ML is a crucial step that helps enhancing the quality of data to promote the extraction of meaningful insights from data. Data preprocessing in ML refers to the technique of preparing the raw data to make it suitable for a building and training ML models, it transforms raw data into an understandable and readable format. Typically, real-world data is incomplete, inconsistent, inaccurate, and often lacks specific attribute values [163]. This is where data preprocessing comes into play; it aids in the cleaning, formatting, and organization of the raw data, making it ready for use by ML models.

- **Data Cleaning** is particularly done as part of data preprocessing to clean the data by filling missing values, smoothing the noisy data, resolving the inconsistency, and removing outliers. These missing values need to first be handled to optimally leverage available data. A fundamental strategy to use incomplete datasets is to discard entire rows or columns containing missing values. However, this comes at the price of losing informative

data which may be valuable for model training. By contrast, imputation tries to infer the missing values from the known part of the data. Some well-known methods include replacing missing values with zero, mean, median or mode. Noisy data involves removing a random error or variance in a measured variable. *Binning* is the technique that works on sorted data values to smoothen any noise present in it. The data is divided into equal-size bins, and each bin is dealt with independently. All data is a segment can be replaced by its mean, median or boundary values. Another method is *clustering*, it is a creation of groups from data having similar values. The values that don't lie in cluster can be treated as noisy data and can be removed.

- **Data transformation** consolidates the quality data into alternate forms by changing the value, structure, or format of data using the below-mentioned strategies. The very first one is feature scaling. Different columns can be present in varying ranges. For example, there can be a column with a unit of distance, and another with the unit of temperature. Those columns will have strongly different ranges, making it difficult for many ML models to reach an optimal computational state. Some useful methods are often addressed to solve these issues. *Normalization* is the process of scaling individual samples to have unit norm. Each sample with at least one non zero component is rescaled independently of other samples so that its norm (L_1 or L_2) equals one. *Standardization* scales features by removing the mean and scaling to unit variance. The standard score z of a sample x is calculated by

$$z = \frac{(x - \mu)}{\sigma}\,, \tag{2.4.4}$$

where μ is the mean and σ is the standard deviation of training samples. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zero and one, or so that the maximum absolute value to each feature is scaled to unit size.

Dimension reduction is another common way to transform data, it obtains a reduced representation of the dataset that is much smaller in volume but produces the same quality of analytical results. *Principal component analysis* (PCA) using singular value decomposition (SVD) of the data to project it to a lower dimensional space [152]. There are many variants for PCA, like *kernel PCA* [141], which achieves non-linear dimensionality reduction through the use of kernels. It has many applications including denoising, compression and structured prediction.

In many cases, data is in a format that cannot be processed by algorithms. For instance, a column with string values or text will mean nothing to a model that only accepts numerical values as input. Therefore, it is necessary to convert categorical features to integer codes to help the model interpret it. This method is called *categorical encoding*. Some popular methods include ordinary encoding, which embeds values from 1 to k in an ordinal manner, where k is the number of samples in the column. For example, if a column has 3 sizes of shoes, ordinal encoding will assign values 1, 2 and 3 to the different sizes. One-hot encoding can be used when data has no inherent order. One-hot encoding generates one column for every category and assigns a positive value 1 in whichever row that category is present, and 0 when it is a absent, as demonstrated in Figure 2.3.

Feature engineering is the process of using domain knowledge of the data to create features that make ML algorithm work. If feature engineering is done correctly, it increases the predictive power of ML algorithm by creating features from raw data that help facilitate

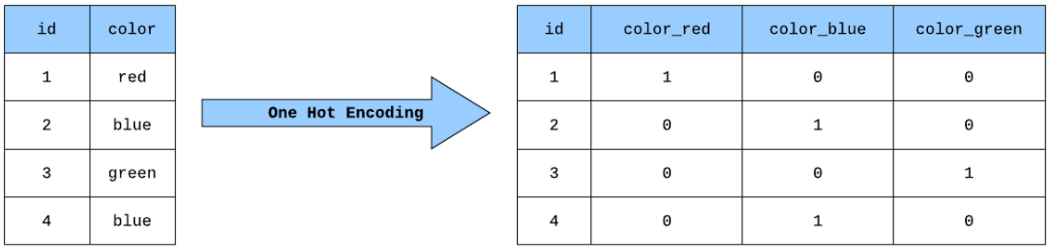

Figure 2.3: One hot encoding converts each categorical value into a new categorical column and assigns a binary value of 1 or 0 to those columns. Each integer value is represented as a binary vector. All the values are zero, and the index is marked with a 1.

the ML process [164]. For instance, the day, month and year can be extracted from a date time column. This gives a new perspective to the model, which can now detect a brand new relation between time and the target variable. In a similar way, discretization is the process to partition continuous features into discrete values. Certain datasets with continuous features may benefit from discretization, because it can transform the dataset of continuous attributes to one with only nominal attributes. Adding complexity to the model by considering nonlinear features of the input data could also be useful. An intuitive and common way is to use polynomial features, which can get features' high-order and interaction terms.

2.4.1.3 Model selection and training

Training a ML model is actually a process of learning the relevant parameters of a prediction function [154]. Training model and testing it on the same data is a methodological mistake: a model that would just "remember" the labels of the samples that it has just seen would have a perfect score but would totally fail to predict unseen data. This is so called overfitting. Besides the model selection, the settings of hyperparameters of the model have to be determined. Since these hyperparameters control the behavior and measure the performance of the model, however, the values of hyperparameters are not adopted by the learning algorithm itself [52]. To state more precisely, the primary intention of model selection is to choose the setting of the hyperparameter in such a way, that the best predictive performance on new data is achieved [32].

A widely used strategy is to divide the dataset into three parts, see Figure 2.4. The first part is the training set, where the algorithm runs on, and the model is trained within it. The parameters will be fitted, and it has to be the massive set. The second part is called the validation set, it is used in the model building process for hyperparameter tuning, feature selections and make other decisions regarding the learning algorithm. It can, therefore, be regarded as a part of the training set. Specifically, the training data can be split into two disjoint subsets. One of these subsets is used to learn the parameters. The other subset is the validation set, used to estimate the generalization error during or after training, allowing for the hyperparameters to be updated accordingly [52]. Once a model is entirely trained by using the training and validation sets, the testing set is generally to evaluate the competing models, this is a third part of the dataset. It is crucial that the test set only used to access the performance, it must not be used in the model building process.

However, by partitioning the available data into three sets, we drastically reduce the number

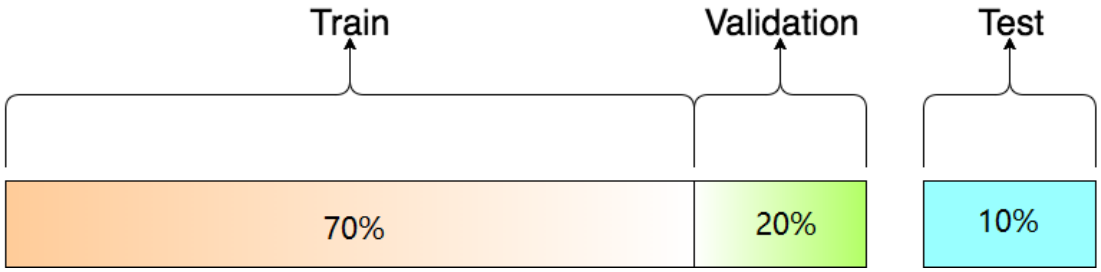

Figure 2.4: Depending on the scale of the dataset, the dataset can be split into two different ways. First, if the size of the dataset is 100 to 1000000, according to the recommendation, the ratio of splitting the dataset should be 70/20/10. For the large-scale dataset, the trend on the ratio of splitting tends to be 98/1/1 or 99.5/0.25/0.25 [105].

of samples which can be used for learning in the model, and the results can depend on a particular random choice for the pair of training or validation sets. One approach to overcome this problem is resample procedure used to evaluate models on limited data sample. Cross-validation (CV) repeats the training and testing computation on different randomly chosen subsets of the original dataset. A typical procedure has a single parameter called k that refers to the number of subsets that a given original dataset is to be divided into. Therefore, the approach is often called k-fold cross-validation. A model is trained using $k-1$ of the folds as training data, the resulting model is validated on the remaining part of the data. For example, the Figure 2.5 shows the workflow of cross-validation when $k = 5$.

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This method can be computationally expensive, but does not waste too much data, which is a major advantage in problem such as inverse inference where the number of samples is very small. Cross-validation provides a more accurate measure of model quality, it merely trade-offs between computational time and the fraction of training-testing split. Every individual sample point gets to be in a testing set exactly once and gets to be in training set $k-1$ times.

2.4.1.4 Model evaluation

Once the model is trained, there are some measurements to evaluate the performance for regression task.

- The *Mean Absolute Error* (MAE) is a loss metric corresponding to the expected value of the absolute error loss. If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, the MAE estimated over n samples is defined as

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1} |y_i - \hat{y}_i| \,. \tag{2.4.5}$$

- The *Mean Squared Error* (MSE) is a loss metric corresponding to the expected value of the squared error. The MSE estimated over n samples is defined as:

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \,. \tag{2.4.6}$$

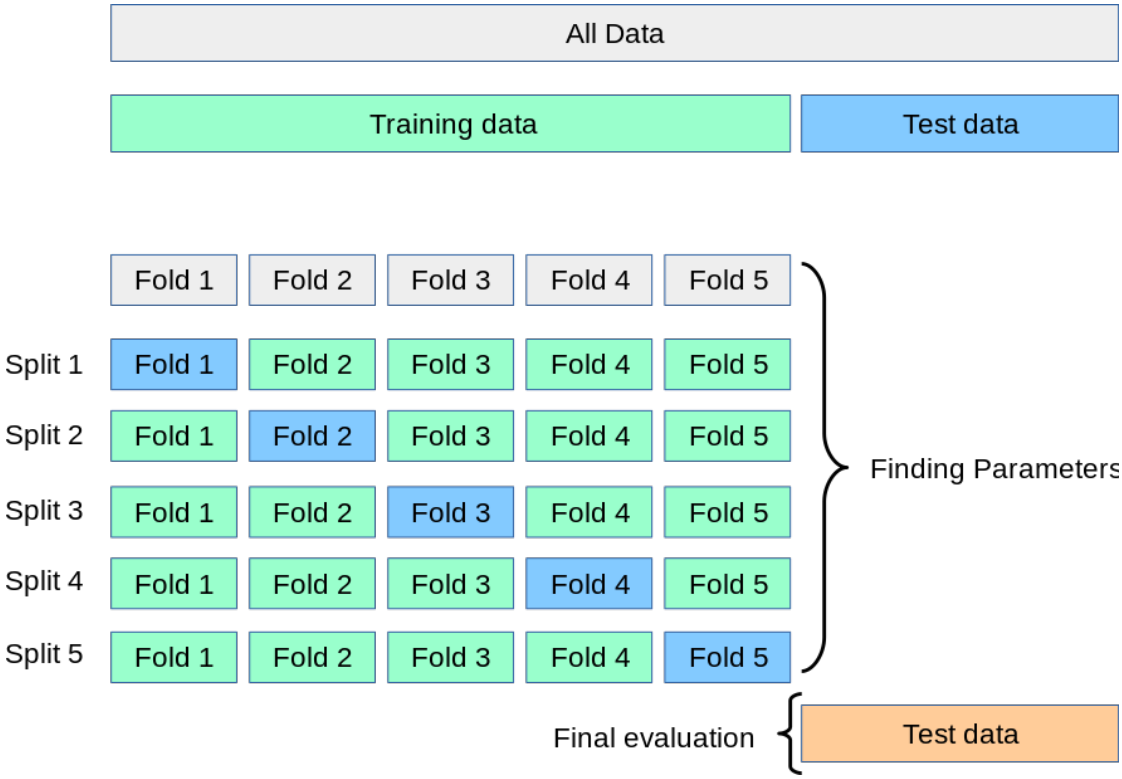

Figure 2.5: 5-folds cross-validation, each time, one of the subset is used as the testing set, and the other four subsets are put together to form a training set [114].

- The *Root Mean Square Error* (RMSE) is the square root of value obtained from MSE:

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{MSE}\,. \tag{2.4.7}$$

- The *Median Absolute Error* (MedAE) is particularly interesting because it is robust to outliers. The loss is calculated by taking the median of all absolute difference between the target and the prediction:

$$\mathrm{MedAE}(y, \hat{y}) = \mathrm{median}(\mid y_1 - \hat{y}_1 \mid, \ldots, \mid y_n - \hat{y}_n \mid)\,. \tag{2.4.8}$$

- The R^2*-score* represents the proportion of variance of y that has been explained by the independent variables in the model. It provides an indication of fitting goodness and therefore a measure of how well unseen samples are likely to be predicted by the model:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}\,. \tag{2.4.9}$$

2.4.2 Regression algorithms

Regression is a technique for investigating the relationship between independent variables or features and a dependent numerical variable or outcome. Solving regression problems is one of the

most common applications for ML models, especially in supervised ML. Algorithms are trained to understand the relationship between independent variables and an outcome. The model can then be leveraged to predict the outcome of new and unseen data, or to fill a gap in missing data. There are a variety of approaches used in ML to perform regression. Various popular algorithms are employed to achieve ML regression. These techniques may include varying numbers of independent variables or process diverse types of data. Unique types of ML regression models may also assume a varied relationship between the independent and dependent variables. For instance, linear regression is one of most simplest algorithm, which assume that the relationship is linear, so would not be effective with datasets with nonlinear relationships. Some of the most common regression algorithms used in this thesis will be introduced below.

2.4.2.1 XGBoost

XGBoost [23], which stands for *Extreme Gradient Boosting*, represents an advancement in supervised learning algorithms, conceived as a modification of the Gradient Tree Boosting algorithm [47]. This method utilizes an iterative process wherein a learning algorithm is trained with the objective of predicting the output values from a model. This prediction process is optimized by minimizing the MSE across a sequence of weak learners, with each successive stage refining the prediction of its predecessor. A significant characteristic of the Gradient Tree Boosting algorithm is its incorporation of a learning rate. This metric quantifies the magnitude of the correction factor that each tree applies relative to its underlying counterpart, thereby influencing the algorithm's capacity to adjust and improve its predictive ability.

For a single tree model, the predicted output $\hat{y}$ is obtained by the sum of K additive functions [23]:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)\,, \quad f_k \in \mathcal{F}\,, \tag{2.4.10}$$

where

$$\mathcal{F} = \{f\colon f(x) = w_{q(x)} \;\text{ with }\; q\colon \mathbb{R}^m \to \{1,\dots,T\}\,,\; w \in \mathbb{R}^T\}\,.$$

In this context, $\mathcal{F}$ represents the space of regression trees, q delineates the architecture of each tree, providing a mapping from an input sample to its respective leaf index, and T signifies the total count of leaves withing a given tree. Each f_k is associated with a distinct tree configuration q and corresponding leaf weightings w. The objective function for the above model is given by

$$obj(\theta) = \sum_{i=0}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)\,, \tag{2.4.11}$$

where the first term is the loss function and the second term is the regularization parameter. Instead of learning the tree all at once, which would make the optimization harder, the additive strategy is applied, which aims to minimize the loss of previously acquired knowledge and incorporating a new tree:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0\,,\\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)\\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)\\
&\;\;\vdots\\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{2.4.12}$$

The objective function of the above model can be defined as

$$\begin{aligned} obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{K} \Omega(f_k) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant. \end{aligned} \tag{2.4.13}$$

XGBoost iteratively trains an ensemble of shallow decision trees, with each iteration using the error residuals of the previous model to fit the next model. The final prediction is a weighted sum of all of the tree predictions. With XGBoost, trees are built in parallel, following a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in training set. Due to the ensembling of decisions, trees can sometimes lead to very complex models. XGBoost uses both Lasso and Ridge regularization to penalize such highly complex models. Furthermore, XGBoost incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data. Most existing tree-based algorithms can find the split points when the data points are of equal weights. However, they are not equipped to handle weighted data. XGBoost employs a distributed weighted quantile sketch algorithm to effectively handle weighted data. Furthermore, for faster computing, XGBoost can make use of multiple cores on the CPU or GPU. This is possible because of a block structure in its system design. Data is sorted and stored in in-memory units called blocks. Unlike other algorithms, this enables the data layout to be reused by subsequent iterations, instead of computing it again. This feature also serves useful steps like split finding and column sub-sampling. In XGBoost, non-continuous memory access is required to get the gradient statistics by row index. Hence, XGBoost has been designed to make optimal use of hardware. This is done by allocating internal buffers in each thread, where the gradient statistics can be stored.

There are two commonly used variants of XGBoost. The first one is *LightGBM* (Light Gradient Boosting Machine) [73]. LightGBM uses histogram-based algorithm [127, 71, 82], which bucket continuous feature values into discrete bins. This speeds up training and reduces memory usage. LightGBM applies Gradient-Based One-Side (GOSS) sample to exclude the significant portion of data instance with small gradients and only uses the remaining data to estimate the information gain. Since the data instance with large gradients play a more important role in the computation of information gain, GOSS can obtain quite accurate information gain with a relatively much smaller dataset. In addition, LightGBM benefits from Exclusive Feature Bundling (EFB), which bundles mutually exclusive features with nothing, but it rarely takes non zero values simultaneously to reduce the number of features; this results in effective feature elimination without hurting the accuracy of the split point.

CatBoost [123] (Category Boosting) is another popular variant of XGBoost. It builds symmetric, balanced trees, unlike XGBoost and LightGBM. In every step, leaves from the previous tree are split using the same condition. The feature-split pair that accounts for the lowest loss is selected and used for all the leave's nodes. This balanced tree architecture aids in efficient CPU implementation, decrease prediction time, makes swift model appliers, and control overfitting as the structure serves as regularization. Classic boosting algorithms are prone to overfitting on small or noisy datasets due to a problem known as prediction shift. When calculating the gradient estimate of a data instance, these algorithms use the same data instance that the model was built with, thus having no chances of experiencing unseen data. CatBoost, on the other hand, uses the concept or ordered boosting, a permutation-driven approach to train model on a subset of data while calculating residuals on another subset, thus preventing target leakage and overfitting.

2.4.2.2 Neural networks

The concept of the *neural network* (NN), also known as *artificial neural network* (ANNs), draws inspiration from biological neural systems to model and understand complex patterns within data [2]. These models constitute a central component in the field of ML and have been instrumental in driving advancements in areas such as image recognition, natural language processing, and autonomous vehicles [33].

A NN is composed of interconnected units or nodes, termed "neurons" or "perceptrons". These units are organized into layers, including an input layer, one or more hidden layers, and an output layer, as shown in Figure 2.6. Each connection between the units carries an associated weight, which is adjusted during the training process to minimize the error in the network's output.

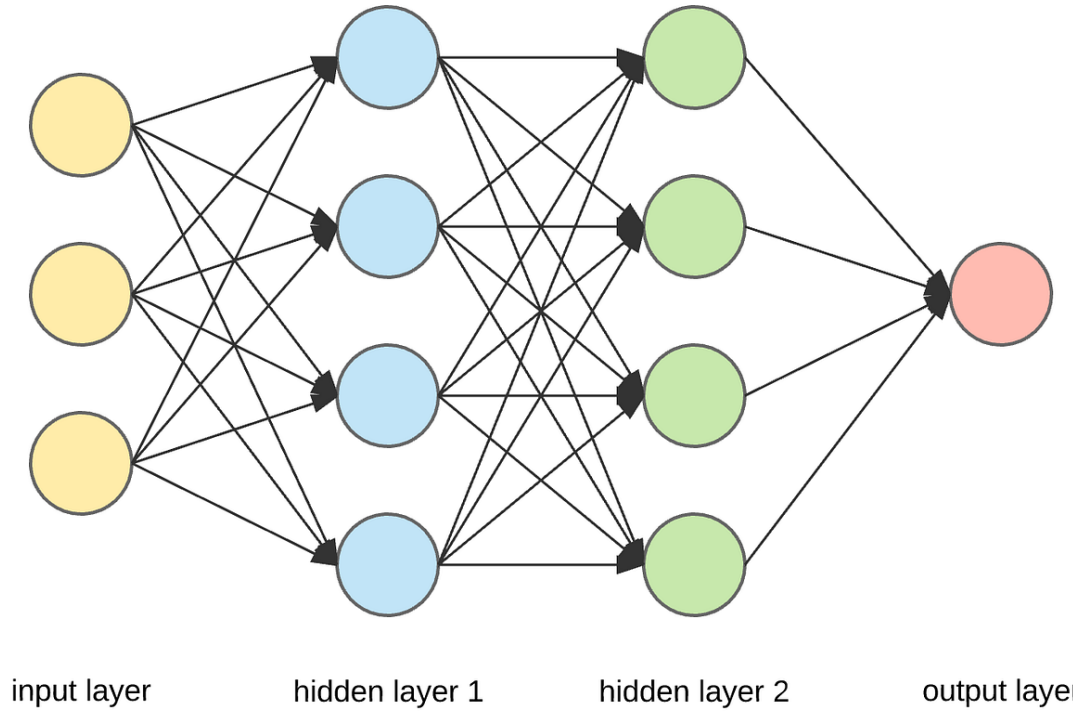

Figure 2.6: Structure of a NN, the inputs are marked with yellow circles, whereas the blue circles and green circles illustrate the hidden neurons, the final output is marked with red circles. Each layer is fully connected with the neuron of the subsequent layer [107].

The structure shown in Figure 2.6 is a multi-layer fully connected net, since every neuron in one layer is connected to every other neuron in the next layer. Inputs to a neuron can either be features from a training set or outputs from a previous layer's neurons [107]. A neuron takes the weighted sum of its inputs and passes it through a non-linear activation function. The output of a neuron, which then becomes the input of another neuron in the next layer, is thus given by

$$z = f(b + x \cdot w) = f\big(b + \sum_{i=1}^{m} x_i w_i\big)\,, \tag{2.4.14}$$

where f is a non-linear activation function, w is a weight vector and b is the bias. The final

output is calculated by performing this procedure recursively for all neurons.

Activation functions are used to convert an input signal of a neuron in a NN to an output signal. Specifically, in NN, by giving a linear combination of inputs and weights from the previous layer, the activation function controls the way how the information passes on to the next layer. An ideal activation function for NN is both non-linear and differentiable. However, the *Rectified Linear Unit* (ReLU) activation function

$$f(x) = \max(0, x) \tag{2.4.15}$$

has become very common in the context of NN in recent years [1]. The ReLU function and its derivative are shown in Figure 2.7

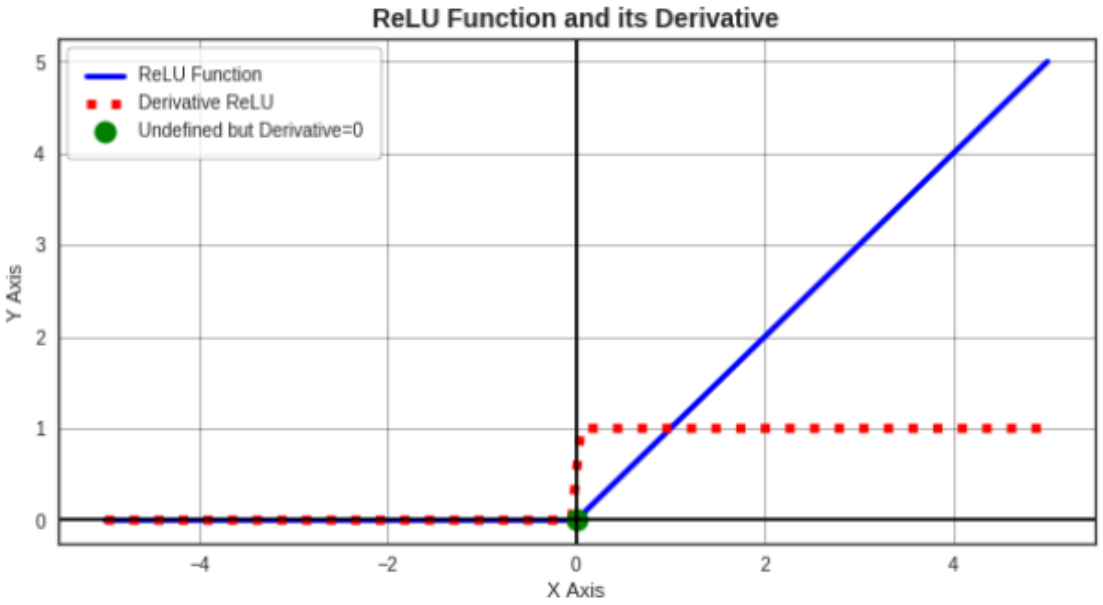

Figure 2.7: ReLU function and its derivative.

The discrepancy between the linear identity function and a ReLU is that a ReLU outputs zero when $x < 0$. This makes the derivatives through a ReLU unit remain large whenever the unit is active [52]. Its derivative at $x \neq 0$ given by

$$f'(x) = \begin{cases} 1 & : x > 0\,, \\ 0 & : x < 0\,. \end{cases} \tag{2.4.16}$$

The derivative of ReLU is 1 everywhere that the unit is active (i.e. attains a non-zero value). Notice that the function is not differentiable at $x = 0$; for convenience, however, it is convention to define $f'(0) = 0$. Furthermore, the input of ReLU is usually the result of a number of summed products, so the probability for it be exactly 0 is low.

The power of NNs comes from their ability to learn complex patterns and representations from data. This is achieved through a process called *backpropagation* and an optimization algorithm, typically a variant of gradient descent. Backpropagation calculates the gradient of the cost function with respect to the weights of the network, and gradient descent then uses this gradient to update the weights and minimize the cost function. Backpropagation is a very common NN

learning algorithm because it is conceptually simple, computationally efficient, and because it often works [81].

The goal of backpropagation is to find the partial derivatives of the loss function regarding all of the weights and biases in a NN. Ultimately, this means computing

$$\frac{\partial J}{\partial w_{jk}^l}, \quad \frac{\partial J}{\partial b_j^l},$$

where w_{jk}^l is the weight from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer and b_j^l is the bias of the j^{th} neuron in the l^{th} layer. Figure 2.8 gives an intuitive illustration of backpropagation.

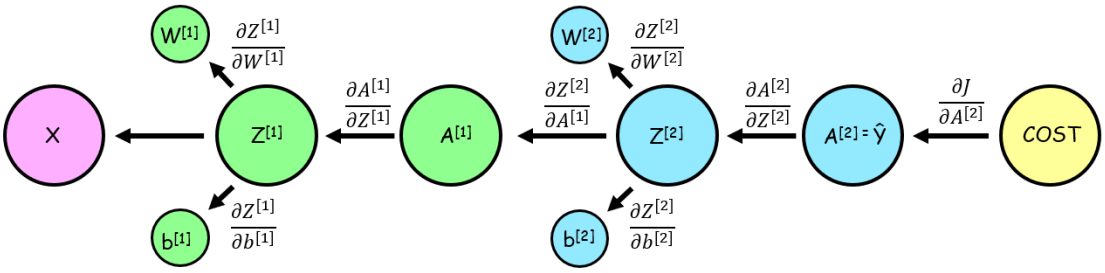

Figure 2.8: Backpropagation [107].

However, computing gradients through recursive application of chain rule is oftentimes more effective. By applying forward propagation, the cost function of the model can be determined. The cost function is $J(a_1^L, a_2^L, \ldots, a_m^L, y)$ where L represents the last layer which has m output neurons. To make partial derivatives easier, the intermediate quantity

$$\delta_j^l \equiv \frac{\partial J}{\partial z_j^l}$$

is introduced, which is the rate of change of the cost function with respect to the j^{th} neuron in the l^{th} layer. By using chain rule, the above equation can be extended as

$$\delta_j^l \equiv \frac{\partial J}{\partial z_j^l} = \sum_k \frac{\partial J}{\partial a_k^l} \frac{\partial a_k^l}{\partial z_j^l}.$$

Since each a_i^l is a function of z_i^l, namely $a_i^l = \delta(z_i^l)$, the expression of δ_j^l can be simplified to

$$\delta_j^l = \frac{\partial J}{\partial a_j^l} \sigma'(z_j^l)$$

The first term $\frac{\partial J}{\partial a_j^l}$ denotes how fast the cost is changing as a function of the j^{th} output activation. The second term $\sigma'(z_j^l)$ measures rate at which the activation function σ is changing at z_j^l. For the backpropagation algorithm, by using chain rule a representation for δ^l in terms of δ^{l+1} can be derived, i.e. from the very end layer of a NN, computing the way backward, all the values δs can be found. So analogously,

$$\delta_j^l = \sum_k \frac{\partial J}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l}.$$

In particular,

$$\delta_j^l = \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1} .$$

Now, z_k^{l+1} can be explicitly written as

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \delta(z_j^l) + b_k^{l+1} .$$

The derivative of z_k^{l+1} with respect to z_k^l is given by

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \delta'(z_j^l) ,$$

so the equation of the weight derivative with respect to cost function is

$$\frac{\partial J}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l .$$

Thus backpropagation provides a way to compute the derivative of the cost function; the derivative with respect to weights and biases can be immediately used with gradient-based optimization methods.

2.4.3 Hyperparameter tuning

A hyperparameter for a ML algorithm is defined as a variable to be set prior to the actual application of the algorithm to the data, one that is not directly learned from the training process [70]. Hyperparameter selection is crucial for the success of ML architecture since they profoundly impact the behavior of the learned model. For any hyperparameter influences the valid capacity of a model, it is significantly to select its value based on a validation set. Hyperparameters represent the numerous decisions that the user must make while building a ML model, such as data processing steps, NN architecture, and the optimizer utilized during training. Each hyperparameter has a complex impact on the model's ability to predict, and more advanced models (like deep NN) have an increasing number of hyperparameters to adjust. Modifications to the hyperparameters may have a significantly influence on the model's quality. In the following, three main strategies to optimize the ML hyperparameter will be introduced.

2.4.3.1 Grid search

Grid search is a naive and straightforward algorithm that can be used for hyperparameter optimization. With this method, a set of parameter values should be defined to train the model for all possible parameter combinations, evaluating each model and then selecting the architecture which produces the best results. Grid search is an exhaustive algorithm that spans all the combinations, so it can actually find the best point in the domain. The great drawback is that it is very slow. Checking every combination of the searching space requires a lot of time that, sometimes, is not available [10].

2.4.3.2 Random search

Grid search thoroughly explores through the entire hyperparameter space and is not feasible in high dimension scenarios. By contrast, a *random search* is a valid approach, which samples the

search space randomly, and is broadly used in practice. One of the primary theoretical backings to motivate the use of random search is the fact for most cases that hyperparameters are not equally important [10]. Random search suggests configurations randomly from the subset of the hyperparameter space, where each configuration is sampled from a distribution over possible parameter values and then evaluate them. Therefore, it requires less computational time. Besides, a budget can be chosen independently of the number of hyperparameters and possible values. The drawback of random search, however, is that it does not use information from prior experiment to choose the next possible setting. This is particularly impermissible when the cost of running is enormous, and it doesn't guarantee that an optimal set of hyperparameters can be found.

2.4.3.3 Bayesian optimization

Bayesian optimization employs the Bayesian theorem of setting a prior over the target function and combining it with evidence to get a posterior function [12]. It is a robust strategy for searching the extrema of objective functions that are extraordinarily expensive to evaluate. Bayesian inference derives that the posterior probability of a model H given current evidence of data D is proportional to the likelihood of D given H multiplied by the prior probability of H:

$$P(H \mid D) = \frac{P(D \mid H) \cdot P(H)}{P(D)} \,. \tag{2.4.17}$$

In Bayesian optimization, the prior indicates the belief about the hyperparameter space of possible objective function. The posterior distribution captures the updated beliefs and will be computed over the objective function based on the data. There are a few different algorithms for this type of optimization. In the next, these algorithms will be discussed.

By definition, a Bayesian optimization depends on a prior distribution over objective function. That is, the beliefs about $f(x)$ for each x should be determined, where f is a stochastic function. A *Gaussian Process* (GP) is well-suited to the task which contains continuous hyperparameters and the prior is homogeneous [98]. GP is a generalization of *Multivariate Gaussian Distributions* to an infinite-dimension stochastic process, which means that for each input x, GP has defined a mean function m and covariance function k: [12]

$$f(x) \sim GP(m(x), k(x, x')) \,. \tag{2.4.18}$$

The basic assumption behind GP is that if vectors x and x' are similar, then $f(x)$ and $f(x')$ should be similar, too. The mean function $m(x)$ encodes a prior expectation of the unknown function. In most cases, the prior mean is set to zero $m(x) = 0$. The covariance function $k(x, x')$ returns a measure of the similarity of x and x', i.e. encodes how similar $f(x)$ and $f(x')$ should be. The selection of covariance function for the GP is important, as it defines the smoothness of samples drawn from it [12]. In general, the covariance function k is a positive definite kernel function. Typically, the default covariance function is the *Squared Exponential Kernel* (RBF or Gaussian kernel)

$$k(x, x') = \exp\left(-\frac{1}{2\theta^2} \big|\big|x - x'\big|\big|^2 \right), \tag{2.4.19}$$

where θ is the hyperparameter length-scale that controls the width of the kernel. Another critical kernel of GP is Maltérn kernel. It is a generalization of the RBF kernel, which has an additional parameter ν that controls the smoothness of the resulting function. It is parameterized by a length-scale parameter $\theta > 0$, which can either be a scalar or a vector with the same dimension

as the input x:

$$k(x, x') = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}||x - x'||}{\theta} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}||x - x'||}{\theta} \right), \tag{2.4.20}$$

where Γ and K are the *Gamma function* and the *Bessel function* of order ν, respectively. Hence, given the data $\{x_{1:t}, f_{1:t}\}$ from previous iterations, by using GP to decide what data point x_{t+1} should be next, and $f_{t+1} = f(x_{t+1})$:

$$\begin{bmatrix} f_{1:t} \\ f_{t+1} \end{bmatrix} \sim N \left(0, \begin{bmatrix} K & \mathrm{k} \\ \mathrm{k}^T & k(x_{t+1}, x_{t+1}) \end{bmatrix} \right), \tag{2.4.21}$$

where the kernel matrix K is given by

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \cdots & k(x_t, x_t) \end{bmatrix}. \tag{2.4.22}$$

and

$$\mathrm{k} = [k(x_{t+1}, x_1), k(x_{t+1}, x_2), \ldots, k(x_{t+1}, x_t)]. \tag{2.4.23}$$

More precisely, we use the *Sherman-Morrison-Woodbury* formula [79] for covariance matrix inverse

$$P(f_{t+1} \mid \mathcal{D}_{1:t}, x_{t+1}) = N(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1})), \tag{2.4.24}$$

where $\mathcal{D}_{1:t}$ is the first t observations. And μ_t, σ_t^2 are expressed by:

$$\mu_t(x_{t+1}) = \mathrm{k}^T K^{-1} f_{1:t} \tag{2.4.25}$$

$$\sigma_t^2(x_{t+1}) = k(x_{t+1}, x_{t+1}) - \mathrm{k}^T K^{-1} \mathrm{k} \tag{2.4.26}$$

Since the predicted value is distributed as a Gaussian, and this Gaussian has two central statistical mean and variance, to search for the optimal predicted value while trading off high mean (exploitation) and large variance (exploration), a new component called the *Acquisition Function* for Bayesian optimization is introduced. This quantity provides a single measure of how useful it would be to try any given point. The point that maximizes the acquisition function will chosen.

The first option is *Probability of Improvement* (PI). This acquisition function chooses the next query point as the one which has the highest probability of improvement over the current max $f(x^+)$. Mathematically, the selection of next point can be defined by

$$x_{t+1} = \mathrm{argmax}(\alpha \mathrm{PI}(x)) = \mathrm{argmax}(P(f(x) \geq (f(x^+) + \epsilon))), \tag{2.4.27}$$

where $P(\cdot)$ indicates probability, and ϵ is a small positive number, $x^+ = \mathrm{argmax}_{x_i \in x_{1:t}} f(x_i)$ where x_i is the location queried at i^{th} time step. PI uses ϵ to strike a balance between exploration and exploitation. Increasing ϵ results in querying locations with a large σ as their probability density is spread.

However, PI only examines the likelihood of improvement, but does not consider the extent of potential enhancement. The next criterion, called *Expected Improvement* (EI), does exactly that. It is defined as

$$EI(x) = \mathbb{E}\left[\max\left\{0, f(x) - f(x^+)\right\}\right], \tag{2.4.28}$$

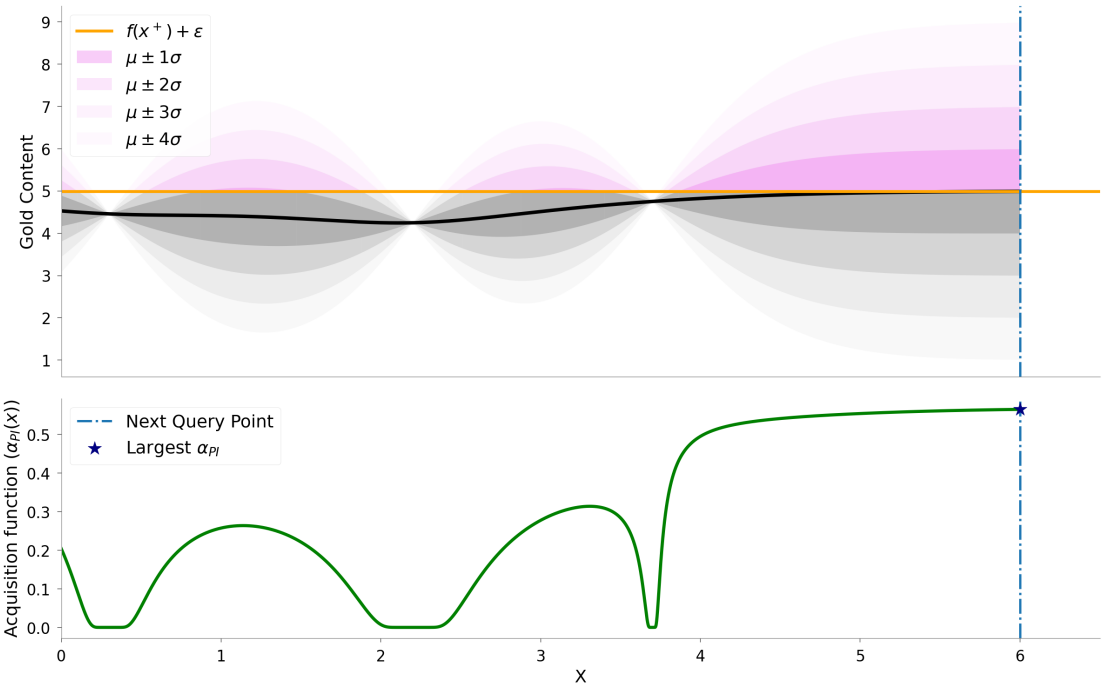

Figure 2.9: The visualization above shows the calculation of $\alpha PI(x)$. The orange line represents the current max (plus an ϵ) or $f(x^+)+\epsilon$. The violet region shows the probability density at each point. The gray regions shows the probability density below the current max. The "area" of the violet region at each point represents the probability of improvement over current maximum [3].

where x^+ is the current optimal set of hyperparameters. Maximising this quantity will yield the point that, in expectation, improves upon f the most. Furthermore, the EI can be evaluated analytically by using a GP [46]:

$$EI(x) = \begin{cases} (\mu(x) - f(\hat{x}))\Phi(Z) + \sigma(x)\phi(Z) & : \sigma(x) > 0\,, \\ 0 & : \sigma(x) = 0\,, \end{cases}$$
$$Z = \frac{\mu(x) - f(\hat{x})}{\sigma(x)}\,, \tag{2.4.29}$$

where $\Phi(z)$, and $\phi(z)$, are the *cumulative distribution function* (CDF) and *probability density function* (PDF) of the standard normal distribution. Intuitively, if the EI is maximal, then the points which are expected a higher value of f will be sampled, or points in a region of f that haven't explored yet ($\sigma(x)$ is high). In other words, EI provides a trade-off between exploitation and exploration.

2.4.4 AutoML

Automated machine learning, also referred to as *AutoML*, is the process of automating the time-consuming, iterative tasks of ML model development [168]. Traditional ML model development is resource-intensive, requiring significant domain knowledge and time to produce and compare

dozens of models. With AutoML it is possible to accelerate ML model development with great ease and efficiency. Naturally, the AutoML's high level of automation enables non-experts to use ML models and approaches without extensive prior familiarity with ML.

AutoML aims to automatically compose and parameterise ML algorithms into ML pipelines with the objective of optimizing a certain given metric. Typically, the algorithms are connected to preprocessing or to the main functionality, like classification or regression. A supervised AutoML system can be described as

$$f(x) = \upsilon_{\theta_\upsilon}\left(\Phi_{\theta_\Phi(X)}\right), \tag{2.4.30}$$

where f is the task's best generalization. The function f is also identical to a full model or pipeline [36]. The pipeline is described by υ, which denotes the supervised learning algorithm (e.g., XGBoost, Random Forest, etc.) and θ_υ, which denotes the hyperparameters of the supervised learning algorithm. Moreover, Φ is the processing technique (e.g., feature imputation/feature selection, etc.), and, if applicable, $\theta_\Phi(X)$ is the hyperparameters associated with the preprocessing technique chosen. Finally, an AutoML system will attempt to identify the optimal combination of preprocessing technique and learning algorithm, as well as their respective hyperparameters.

In a nutshell, an AutoML search optimization system aims to perform the optimization of estimators and predictors (i.e., algorithm selection) [132]; the optimization of learning algorithms and their hyperparameters [36, 55, 150, 151], and the optimization of meta-learning algorithms [40, 143]. Meta-learning in ML refers to learning algorithms that learn from other learning algorithms. Most commonly, this means the use of ML algorithms that learn how to best combine the predictions from other ML algorithms in the field of ensemble learning. Preprocessing techniques are subject to the same optimization process but based on a subset of techniques designed specifically for preprocessing.

To reach the mail goal of AutoML, there are some subproblems must be addressed firstly. Combined Algorithm and Hyperparameter Selection (CASH) problem [151] involves autonomously selecting a learning algorithm and its parameters, while the Hyperparameter Optimisation (HPO) problem involves offering the best model instance from a vector of selected techniques. Combining HPO with CASH encodes all viable learning algorithms and associated hyperparameter combinations. The CASH and HPO problems involve testing a huge number of hypotheses and selecting the most interesting one as the best predictive model for the training set.

Sequential Model-based Algorithm Configuration (SMAC) [40, 39, 60] is a versatile HPO tool that helps algorithm creators optimize hyperparameters. This promising strategy builds a promising configuration using tree search, compares all possible configurations using a Random Online Adaptive Racing (ROAR) method [65], and then reveals the most accurate hyperparameter combination discovered for the algorithm and the given dataset.

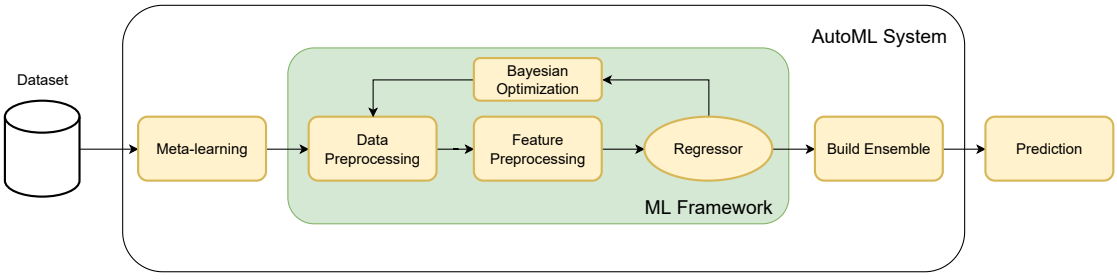

Figure 2.10: The architecture of an AutoML system [40].

The AutoML pipeline accepts the training dataset as input. The meta-learning phase is then

executed, which is one of the greatest advancements of the framework in the AutoML field as it, broadly speaking, uses the similarity of dataset to some already known from the literature or web, and if there is a match, a list of techniques that performed well on such a dataset is passed as the priority to investigate them through the pipeline. Then, regardless of whether the meta-learning step outputs, a data preprocessing method, a feature preprocessing method, a regression algorithm is randomly selected. The Bayesian optimizer is then used to optimize the hyperparameters until the sub-pipeline threshold is attained. This cycle is repeated for every available regressor until the overall threshold is reached. Note that there are two unique thresholds: one for the search of a particular sub-pipeline, such as after the process has gone through a, b, and c to optimize the hyperparameters; and a second, more global threshold that applies to all searches simultaneously. As soon as the overall threshold is hit, the pipeline stops and builds an ensemble of all sub-pipeline combinations, ranking them from most accurate to least accurate based on a user-defined metric, and provides the user with the best model. An AutoML system could not only find the best performance model, but also provide some post-analysis. Post-analysis include interpretation, explanation, and visualization of the analysis process and the output model, model production, model monitoring, and model updating.

AutoML is important because it represents a milestone in the fields of ML and AI. AI and ML have been subject to the black-box criticism, meaning that ML algorithms can be difficult to reverse engineer [161]. Although they improve efficiency and processing power to produce results, it can be difficult to track how the algorithm delivered that output. Consequently, this also makes it difficult to choose the correct model for a given problem, because it can be difficult to predict a result if a model is a black box. AutoML helps to make ML less of a black-box by making it more accessible. This process automates parts of the ML process that apply the algorithm to real-world scenarios. A human performing this without need to understanding of the algorithm's internal logic and how it relates to the real-world scenarios. It learns about learning and makes choices that would be too time-consuming or resource-intensive for humans to do with efficiency at scale.

2.5 Numerical optimization

This study concentrates on the evolution from strictly predictive to prescriptive analysis, an approach that incorporates data-driven models for process optimization within the context of scrap optimization in the foundry industry. It requires the identification of optimal control variable values that enhance the output quality in alignment with the process model. For data-driven prediction algorithms, this represents a restricted optimization problem that calls for the identification of the best control variable values that satisfies the process constraints while minimizing the objective function.

In this context, numerical optimization becomes instrumental in locating the optimal solution. The following introduction will provide a detailed explanation of essential numerical optimization methodologies. These will include both derivative-based and derivative-free methods, and will also introduce surrogate-based optimization.

In general, optimization refers to the investigation of extremal points and values intrinsic to mathematical functions. The primary objective of optimization is to minimize or maximize a real-valued function, designated as objective function, within a defined set of points $\Omega \subset \mathbb{R}^n$, commonly referred to as the *feasible set*. It is widely acknowledged in the academic community that pivotal information requisite for optimization is encapsulated within the (potentially generalized) derivatives of the concerned functions. Nevertheless, in practical applications, the computation of such derivatives may be fraught with unreliability or might involve expensive

computational costs, if they are available at all. This practical challenge prompts the study of *Derivative-Free Optimization* (DFO) as referenced in [27, 5], where the resolution of problems is executed exclusively through the use of function values.

In this work, the constrained optimization problem

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \quad & f(x) && && \text{(2.5.1a)}\\ \text{s.t.} \quad & c_i(x) \leq 0 && \forall\, i \in \mathcal{I}\,, && \text{(2.5.1b)}\\ & c_i(x) = 0 && \forall\, i \in \mathcal{E}\,, && \text{(2.5.1c)}\\ & l_k \leq x_k \leq u_k && \forall\, k \in \{1,\dots,n\} && \text{(2.5.1d)} \end{aligned}$$

is considered, where the $f\colon \mathbb{R}^n \to \mathbb{R}$ is the objective function, $\mathcal{I}$ and $\mathcal{E}$ are disjoint finite index sets, $c_i\colon \mathbb{R}^n \to \mathbb{R}$ are the constraint functions and l_k, u_k denote the lower and upper bounds on x_k, respectively. By defining the feasible set

$$\Omega \;=\; \{x \in \mathbb{R}^n \,|\, c_i(x) \leq 0\,,\; c_j(x) = 0\,,\; l_k \leq x_k \leq u_k \;\; \forall\, i \in \mathcal{I},\, j \in \mathcal{E},\, k \in \{1,\dots,n\}\}\,,$$

problem (2.5.1) can be stated as

$$\underset{x\in\Omega}{\operatorname{argmin}} \quad f(x)\,.$$

In particular, the Lagrangian function of this problem is defined by

$$\mathcal{L}(x,\lambda) \;=\; f(x) + \sum_{i\in\mathcal{I}\cup\mathcal{E}} \lambda_i c_i(x), \quad \text{for } x \in \mathbb{R}^n \text{ and } \lambda_i \in \mathbb{R}, \text{with } i \in \mathcal{I}\cup\mathcal{E}, \tag{2.5.2}$$

where $\lambda = [\lambda_i]^T_{i\in\mathcal{I}\cup\mathcal{E}}$ is the dual variable of the considered problem.

2.5.1 Sequential quadratic programming

The *Sequential Quadratic Programming* (SQP) algorithm works by generating steps by solving quadratic subproblems, and is recognized for its effectiveness in resolving nonlinearly constrained optimization. The versatility of the SQP methodology allows its application within both *line-search* and *trust-region* frameworks, rendering it suitable for problems of varying scales, from small to large.

2.5.1.1 Overview of the SQP method

The SQP method is widely recognized as a highly effective approach for addressing the problem as defined by Equation (2.5.1), particularly when derivatives of f and c_i for $i \in \mathcal{I}\cup\mathcal{E}$, are accessible. A depiction of the traditional SQP method can be found in Algorithm 1.

The new iterate is given by $(x^k + d^k, \lambda^{k+1})$, where d^k and λ^{k+1} are the solution and the corresponding Lagrange multiplier of Equation (2.5.3). A local SQP method for Equation (2.5.1) is thus given by Algorithm 1 with the modification that the step is computed from Equation (2.5.3). The initial mention of a similar methodology can be traced back to [159], wherein H^k was defined as $\nabla^2_{x,x}\mathcal{L}(x^k,\lambda^k)$. Han introduced a methodology that aligns with the context, as documented in his works [62, 63]. However, his approach was confined to approximating only the $H^k = \nabla^2_{x,x}\mathcal{L}(x^k,\lambda^k)$. In contrast, Garcìa-Palomares and Mangasarian [113] extended this by implementing quasi-Newton approximations to the entirety of the second derivative matrix of the Lagrangian. Furthermore, Han proposed a line-search methodology to ensure both global convergence and local Q-superlinear convergence rate. This strategy necessitates the

Algorithm 1: Traditional SQP method

Data: Objective function f, constraint functions $\{c_i\}_{i\in\mathcal{I}\cup\mathcal{E}}$, initial guess $x^0 \in \mathbb{R}^n$, and estimated Lagrange multiplier $\lambda^0 = [\lambda_i^0]_{i\in\mathcal{I}\cup\mathcal{E}}^T$.

for $k = 0, 1, \ldots$ **do**

Define $H^k \approx \nabla^2_{x,x}\mathcal{L}(x^k, \lambda^k)$

Generate a step $d^k \in \mathbb{R}^n$ by solving approximately

$$\min_{x\in\mathbb{R}^n} \quad \nabla f(x^k)^T d + \frac{1}{2} d^T H^k d \tag{2.5.3a}$$

$$\text{s.t.} \quad c_i(x^k) + \nabla c_i(x^k)^T d \le 0, \quad i \in \mathcal{I}\,, \tag{2.5.3b}$$

$$c_i(x^k) + \nabla c_i(x^k)^T d = 0, \quad i \in \mathcal{E}\,, \tag{2.5.3c}$$

Update the iterate $x^{k+1} \leftarrow x^k + d^k$

Estimate the Lagrange multiplier $\lambda^{k+1} = [\lambda_i^{k+1}]_{i\in\mathcal{I}\cup\mathcal{E}}^T$

end

condition that the second-order directional derivative, denoted as $H^k = \nabla^2_{x,x}\mathcal{L}(x^*, \lambda^*)$, is positive definite at the solution point (x^*, λ^*) [62]. Powell conducted extensive research on the method in the same direction, specifically proposing the application of the *damped BFGS* (Broyden–Fletcher–Goldfarb–Shanno) quasi-Newton formula to update H^k [118, 119, 121]. This application of the BFGS quasi-Newton formula represents a significant advancement in the field, demonstrating a novel approach to address this computational challenge.

The interpretation of the SQP subproblem is that at each iteration, the method approximates the original non-linear problem by a quadratic programming problem (the subproblem), and then solves this problem to get a direction for a step. The quadratic approximation relates to the Lagrangian associated with the given problem, and the constraints have been linearized for the sake of simplification and analytical tractability. Subsequent to this point, attention is focused on a single iteration of Algorithm 1, with a constant value for k. The rationale behind updating x^k via a solution to Equation (2.5.3) will be described, reinforcing the validity of this approach within the given context.

Bilinear approximation of the *KKT* (Karush-Kuhn-Tucker) conditions is the most classical interpolation of the SQP subproblem. In alignment with [109], given that $x^* \in \mathbb{R}^n$ is a local solution to the problem delineated in Equation (2.5.3), and under the consideration of specific moderate assumptions, a Lagrange multiplier $\lambda^* = [\lambda_i^*]_{i\in\mathcal{I}\cup\mathcal{E}}^T$ with $\lambda_i^* \in \mathbb{R}^n$ for all $i \in \mathcal{I}\cup\mathcal{E}$ can be identified:

$$\begin{cases} \nabla_x \mathcal{L}(x^*, \lambda^*) = 0\,, & (2.5.4a)\\ c_i(x^*) \le 0 \qquad \forall\, i \in \mathcal{I}\,, & (2.5.4b)\\ c_i(x^*) = 0 \qquad \forall\, i \in \mathcal{E}\,, & (2.5.4c)\\ \lambda_i^* c_i(x^*) = 0 \qquad \forall\, i \in \mathcal{I}\,, & (2.5.4d)\\ \lambda_i^* \ge 0 \qquad \forall\, i \in \mathcal{I}\,. & (2.5.4e) \end{cases}$$

Considering Equation (2.5.4) as a nonlinear system constituted by a combination of inequalities and equalities, and perceiving (x^k, λ^k) as an approximation to the optimal solution (x^*, λ^*), the objective is to solve this system using the *Newton-Raphson method* [124, 134]. To initiate the method from the starting point (x^k, λ^k), the objective is to identify a step (d, μ) which conforms to the system that provides a linear approximation of Equation (2.5.4) at the point

(x^k, λ^k). This approach is inherently associated with the Newton-Raphson method's iterative process for refining approximate solutions to the system of nonlinear equations and inequalities. Nonetheless, as highlighted by Robinson [133], there exists a critique against such a method due to its inability to solve a linear program in a single iteration. To address this limitation, the pair (d, μ) solves the bilinear approximation

$$\begin{cases} \nabla_x \mathcal{L}(x^k, \lambda^k + \mu) + \nabla^2_{x,x} \mathcal{L}(x^k, \lambda^k) d = 0\,, & \text{(2.5.5a)} \\ c_i(x^k) + \nabla c_i(x^k)^T d \leq 0 \qquad \forall\, i \in \mathcal{I}\,, & \text{(2.5.5b)} \\ c_i(x^k) + \nabla c_i(x^k)^T d = 0 \qquad \forall\, i \in \mathcal{E}\,, & \text{(2.5.5c)} \\ \lambda_i^k [c_i(x^k) + \nabla c_i(x^k)^T d] + \mu_i c_i(x^k) = 0 \qquad \forall\, i \in \mathcal{I}\,, & \text{(2.5.5d)} \\ \lambda_i^k + \mu_i \geq 0 \qquad \forall\, i \in \mathcal{I}\,. & \text{(2.5.5e)} \end{cases}$$

of Equation (2.5.4). We also consider the system

$$\begin{cases} \nabla_x \mathcal{L}(x^k, \lambda^k + \mu) + \nabla^2_{x,x} \mathcal{L}(x^k, \lambda^k) d = 0\,, & \text{(2.5.6a)} \\ c_i(x^k) + \nabla c_i(x^k)^T d \leq 0 \quad i \in \mathcal{I}\,, & \text{(2.5.6b)} \\ c_i(x^k) + \nabla c_i(x^k)^T d = 0 \quad i \in \mathcal{E}\,, & \text{(2.5.6c)} \\ (\lambda_i^k + \mu_i)[c_i(x^k) + \nabla c_i(x^k)^T d] = 0 \quad i \in \mathcal{I}\,, & \text{(2.5.6d)} \\ \lambda_i^k + \mu_i \geq 0 \quad i \in \mathcal{I}\,; & \text{(2.5.6e)} \end{cases}$$

note that the sole difference between (2.5.5) and (2.5.6) lies in the condition stipulated by Equation (2.5.6d), which encapsulates the bilinear term $\mu_i \nabla c_i(x^k)^T d$. If the problem delineated by Equation (2.5.1) constitutes a linear program, then Equation (2.5.6) accurately represents its KKT system, whereas Equation (2.5.5) merely approximates it. It is noteworthy that the bilinear system expressed by Equation (2.5.6) essentially encompasses the KKT conditions of the SQP subproblem, as defined by Equation (2.5.3), with $\lambda^k + \mu$ functioning as the Lagrange multiplier.

Consequently, a KKT pair for the SQP subproblem Equation (2.5.2) bears a resemblance to a Newton-Raphson step for the KKT system of the problem Equation (2.5.1). Furthermore, it exceeds the latter in efficacy in that the resultant method is capable of resolving a linear program within a single iteration. Observe that the incongruity between system (2.5.5) and system (cf. Equation 2.5.6) dissolves when $\mathcal{I} = 0$ within problem (cf. Equation 2.5.1). Finally, under these circumstances, a KKT pair for the SQP subproblem (cf. Equation 2.5.2) precisely corresponds to a Newton-Raphson step for the KKT system of problem (cf. Equation 2.5.1).

The approximation of a modified Lagrangian is another common used approach [133]. Let $\tilde{\mathcal{L}}$ denote the function

$$\tilde{\mathcal{L}}(x, \lambda) \;=\; f(x) + \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i \delta_i(x), \quad \text{for } x \in \mathbb{R}^n \text{ and } \lambda_i \in \mathbb{R}, \text{with } i \in \mathcal{I} \cup \mathcal{E},$$

where δ_i, for $i \in \mathcal{I} \cup \mathcal{E}$, is defined by

$$\delta_i(x) \;=\; c_i(x) - c_i(x^k) - \nabla c_i(x^k)^T (x - x^k), \quad \text{for } x \in \mathbb{R}^n.$$

The term δ_i is often designated as the deviation from linearity for c_i at the point x_k as presented in [49, 50]. The SQP subproblem expressed by Equation (2.5.3) with $H^k = \nabla^2_{x,x} \mathcal{L}(x^k, \lambda^k)$ can thus be interpreted as the process of minimizing the second-order Taylor approximation of the Lagrangian function $\tilde{\mathcal{L}}$. This minimization is subject to the linear approximations of the constraints specified in Equations (2.5.1b) and (2.5.1c) evaluated at the point (x^k, λ^k), implying

that

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \nabla_x \tilde{\mathcal{L}}(x^k,\lambda^k)^T d + \frac{1}{2} d^T \nabla^2_{x,x} \tilde{\mathcal{L}}(x^k,\lambda^k) d \\ \text{s.t.} \quad & c_i(x^k) + \nabla c_i(x^k)^T d \leq 0, \quad i \in \mathcal{I}\,, \\ & c_i(x^k) + \nabla c_i(x^k)^T d = 0, \quad i \in \mathcal{E}\,. \end{aligned}$$

The SQP method, upon clarifying its subproblem in this particular format, can be viewed as a specific instantiation of Robinson's method as delineated in [134]. It is well-documented that Robinson's method possesses a local R-quadratic convergence rate, thereby implying the SQP method inherits this characteristic under these conditions.

In the following, we consider the SQP subproblem's Lagrangian and the *Augmented Lagrangian*. First, we will examine an alternative to problem (cf. Equation 2.5.1). This will involve the contemplation of the problem

$$\min_{x\in\mathbb{R}^n} \quad f(x) \tag{2.5.7a}$$

$$\text{s.t.} \quad h(x) = 0\,, \tag{2.5.7b}$$

$$x \geq 0\,, \tag{2.5.7c}$$

with $h\colon \mathbb{R}^n \to \mathbb{R}^m$. Recall that the Lagrangian of (2.5.7) is given by

$$\mathcal{L}(x,\lambda) \;=\; f(x) + \lambda^T h(x), \quad \text{for } x \geq 0 \text{ and } \lambda \in \mathbb{R}^m\,.$$

The corresponding SQP subproblem of the problem (cf. Equation 2.5.7) with assumptions that $x^k \geq 0$ and $\lambda^k \in \mathbb{R}^m$ are given, can be expressed as

$$\min_{d\in\mathbb{R}^n} \quad f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T H^k d \tag{2.5.8a}$$

$$\text{s.t.} \quad h(x^k) + \nabla h(x^k) d = 0\,, \tag{2.5.8b}$$

$$x^k + d \geq 0\,, \tag{2.5.8c}$$

with $H^k \approx \nabla^2_{x,x}\mathcal{L}(x^k,\lambda^k)$. It is essential to observe that the constant term $f(x^k)$ in Equation (2.5.8a) could potentially be omitted, as demonstrated in Equation (2.5.3a).

The Augmented Lagrangian [64, 117] of the problem (cf. Equation 2.5.7) is

$$\mathcal{L}_A(x,\lambda) \;=\; \mathcal{L}(x,\lambda) + \frac{\gamma}{2}||h(x)||^2, \quad \text{for } x \geq 0 \text{ and } \lambda \in \mathbb{R}^m\,, \tag{2.5.9}$$

where $\gamma \geq 0$ is a penalty parameter. The Lagrangian of the SQP subproblem (cf. Equation 2.5.8) can be expressed as

$$\begin{aligned} \tilde{\mathcal{L}}(d,\lambda) \;&=\; f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T H^k d \\ +\lambda^T[h(x^k) + \nabla h(x^k) d], \quad &\text{for } d \geq -x^k \text{ and } \lambda \in \mathbb{R}^m\,, \end{aligned}$$

and the Augmented Lagrangian $\tilde{\mathcal{L}}$ of the SQP subproblem (cf. Equation 2.5.8) is given by

$$\tilde{\mathcal{L}_A}(d,\lambda) \;=\; \tilde{\mathcal{L}}(d,\lambda) + \frac{\gamma}{2}||h(x^k) + \nabla h(x^k) d||^2, \quad \text{for } d \geq -x^k \text{ and } \lambda \in \mathbb{R}^m\,.$$

Assume now that f and h are twice differentiable. Then by direct calculation,

$$\nabla_x \mathcal{L}_A(x^k,\lambda^k) = \nabla_x \mathcal{L}(x^k,\lambda^k) + \gamma \nabla h(x^k)^T h(x^k)$$

and

$$\nabla^2_{x,x}\mathcal{L}_A(x^k,\lambda^k) = \nabla^2_{x,x}\mathcal{L}(x^k,\lambda^k) + \gamma\Big[\nabla h(x^k)^T\nabla h(x^k) + \sum_{i=1}^{m} h_i(x^k)\nabla^2 h_i(x^k)\Big]\,.$$

Therefore, the second-order Taylor expansion of $\mathcal{L}_A(x^k+d,\lambda^k)$ with respect to d at 0 is

$$\begin{aligned}&\mathcal{L}_A(x^k,\lambda^k) + \nabla_x\mathcal{L}_A(x^k,\lambda^k)^T d + \frac{1}{2}d^T\nabla^2_{x,x}\mathcal{L}_A(x^k,\lambda^k)d\\ =\ &\mathcal{L}(x^k,\lambda^k) + \nabla_x\mathcal{L}(x^k,\lambda^k)^T d + \frac{1}{2}d^T H^k d + \frac{1}{2}||h(x^k)+\nabla h(x^k)||^2\,,\end{aligned}$$

where $H^k = \nabla^2 f(x^k) + \sum_{i=1}^m[\lambda_i^k + \gamma h_i(x^k)]\nabla^2 h_i(x^k)$. Note that the Augmented Lagrangian method, which is utilized for addressing the problem outlined in Equation (2.5.7), conventionally updates the dual variable, denoted as λ^k, by:

$$\lambda^{k+1} = \lambda^k + \gamma h(x^k)\,.$$

Consequently, one may interpret the second-order Taylor expansion of the Lagrange multiplier $\mathcal{L}_A$ as the Augmented Lagrangian of the SQP subproblem expressed by Equation (2.5.8), where the Hessian matrix is $H^k = \nabla^2_{x,x}\mathcal{L}(x^k,\lambda^{k+1})$.

Various potential configurations for H^k have been suggested in the literature [108, 158]. For instance, H^k could be assigned the value of $\nabla^2_{x,x}\mathcal{L}(x^k,\lambda^k)$ or a suitable approximation thereof.

2.5.1.2 Merit functions for the SQP method

The *merit function* is a key component of SQP methods and other optimization algorithms [84]. It is used to guide the search direction and to determine the step size in each iteration. The merit function combines the objective function and the constraint violation into a single scalar quantity that can be minimized. The merit function takes into account not only the value of the objective function, but also the extent to which the constraints are violated. This helps to ensure that the optimization algorithm makes progress towards a solution that satisfies the constraints. The forthcoming discussion will describe several merit functions commonly used in the field.

The *Courant function* [28] is the most classical merit function, defined by

$$\varphi_\gamma(x) \;=\; f(x) + \gamma\Big(\sum_{i\in\mathcal{I}}[c_i(x)]_+^2 + \sum_{i\in\mathcal{E}} c_i(x)^2\Big) \qquad \text{for } x\in\mathbb{R}^n \text{ and } \gamma\geq 0\,.$$

Here, the operator $[\cdot]_+ = \max\{\cdot,0\}$ signifies the positive component of a given numerical value. The merit of such a function lies in its differentiability, provided that f and c_i, for $i\in\mathcal{I}\cup\mathcal{E}$, are differentiable. However, under typical circumstances, a global minimizer of φ_γ does not constitute a solution to Equation (2.5.1) when γ is finite. This observable behaviour can be demonstrated through the elementary case of minimizing x, given the constraint that $x\geq 0$.

The category of non-smooth merit functions serves as an illustrative case of the broader class of ℓ_p merit functions, given by

$$\varphi_\gamma(x) \;=\; f(x) + \gamma\Big(\sum_{i\in\mathcal{I}}[c_i(x)]_+^p + \sum_{i\in\mathcal{E}} |c_i(x)|^p\Big)^{1/p} \qquad \text{for } x\in\mathbb{R}^n \text{ and } \gamma\geq 0\,.$$

These merit functions possess the attribute of exactness under certain conditions. In a generalized sense, this implies that a resolution to the constrained problem (cf. Equation 2.5.1) can be retrieved by minimizing φ_γ, provided that γ is sufficiently large [61, 90].

A notable limitation of ℓ_p-merit functions is their potential lack of differentiability. Specifically, even in instances where both f and c_i are differentiable for $i \in \mathcal{I} \cup \mathcal{E}$, the function φ_γ may not possess differentiability at points $x \in \mathbb{R}^n$ where $c_i(x) = 0$ for $i \in \mathcal{I} \cup \mathcal{E}$. Nevertheless, the existence of smooth merit functions exhibiting the property of exactness has been demonstrated, a point that will be elaborated upon in the following section.

The Augmented Lagrangian merit function can be simplified when $\mathcal{I} = \emptyset$. The Augmented Lagrangian [64, 117, 135] of the problem (cf. Equation 2.5.1) can be defined as

$$\mathcal{L}_A(x, \lambda) \;=\; \mathcal{L}(x, \lambda) + \frac{\gamma}{2} \sum_{i \in \mathcal{E}} c_i(x)^2, \quad \text{for } x \in \mathbb{R}^n \text{ and } \lambda = [\lambda_i]_{i \in \mathcal{E}}^T \,,$$

where $\mathcal{L}$ is the Lagrangian function of the problem (cf. Equation 2.5.1). The Augmented Lagrangian merit function is then denoted as

$$\varphi_\gamma(x) \;=\; \mathcal{L}_A(x, \lambda^{\mathrm{LS}(x)}) \,,$$

where λ^{LS} is the least-norm solution to

$$\min_{\lambda} \left\| \nabla f(x) + \sum_{i \in \mathcal{E}} \lambda_i \nabla c_i(x) \right\|, \tag{2.5.10}$$

where $\lambda = [\lambda_i]_{i \in \mathcal{E}}^T$. Given that x^* is a solution to the problem defined in Equation (2.5.1), it follows that the pair $(x^*, \lambda(x^*))$ forms a KKT pair. In the scenario where $\mathcal{I} \neq \emptyset$, to ensure the equivalent property, it becomes essential to incorporate the complementary slackness conditions within the constraints outlined in Equation (2.5.10).

An evident disadvantage of using such a merit function lies in its high computational cost. A single evaluation requires the resolution of a linear least-squares problem, which necessitates the computation of gradients for both the objective function f and the constraint function c_i, where $i \in \mathcal{E}$. This intensifies the computational demand, thereby increasing the expense of the evaluation process.

Another merit function is called *second-order correction*. In practical applications, the SQP method necessitates globalization to ensure global convergence. This is typically achieved through the employment of a merit function, which serves to adjudicate the acceptance or rejection of a step. Nevertheless, the utilization of the merit function could potentially compromise the swift local convergence characteristic of the SQP method. This particular phenomenon is referred to within the academic community as the *Maratos effect* [88]. The observed phenomenon, wherein the SQP method generates a step leading to the augmentation of both the objective function and constraint violation, can occur for certain problem sets irrespective of the proximity of the current iteration to the solution. This step would be excluded by any merit function merging the objective function and the constraint violation, given its tendency to elevate with respect to both parameters [88, 118].

A potential strategy to address the Maratos effect involves the implementation of second-order correction steps. The underlying principle of the second-order correction is to adjust the current step, denoted as $d^k \in \mathbb{R}^n$, by incorporating a term $r^k \in \mathbb{R}^n$. Consequently,

$$\begin{cases} [c_i(x^k + d^k + r^k)]_+ & = \mathrm{o}(||d^k||^2) \,, \quad i \in \mathcal{I} \,, \\ |c_i(x^k + d^k + r^k)| & = \mathrm{o}(||d^k||^2) \,, \quad i \in \mathcal{E} \,. \end{cases}$$

Nonetheless, it is crucial to maintain the magnitude of step d^k within a reasonable range, thereby necessitating the condition $||r^k|| = \mathrm{o}(||d^k||^2)$ to be concurrently satisfied. Perhaps the simplest

of these steps, is the least-square solution [91].

$$\min_{r\in\mathbb{R}^n} \sum_{i\in\mathcal{I}} [c_i(x^k + d^k) + \nabla c_i(x^k + d^k)^T r]_+^2 + \sum_{i\in\mathcal{E}} [c_i(x^k + d^k) + \nabla c_i(x^k + d^k)^T r]^2 \,.$$

A multitude of additional second-order correction procedures may also be established within this framework [24, 25, 41, 48].

2.5.1.3 The trust-region SQP method

Trust-region SQP methodologies exhibit numerous desirable characteristics. Notably, these include their capability to operate independently of the requirement for the Hessian matrix $\nabla^2_{x,x}\mathcal{L}_k$ in Equation (2.5.3) to be positive definite. Additionally, they possess the ability to regulate the quality of steps even when Hessian and Jacobian singularities are present. Furthermore, they incorporate a mechanism that ensures the enforcement of global convergence.

For the remainder of this discussion, the merit function under consideration will be the ℓ_2 norm merit function, as defined by

$$\varphi_\gamma(x) \;=\; f(x) + \gamma \sqrt{\sum_{i\in\mathcal{I}} [c_i(x)]_+^2 + \sum_{i\in\mathcal{E}} |c_i(x)|^2}\,, \quad \text{for } x \in \mathbb{R}^n \text{ and } \gamma \geq 0\,,$$

where γ is the penalty parameter. The subsequent algorithm outlined in this study maintains a penalty parameter, denoted as $\gamma^k \geq 0$, at the k-th iteration. Further, the function φ_{γ^k} is represented as φ^k for convenience. The ℓ_2-merit function $\hat{\varphi}^k$ computed on the SQP subproblem (cf. Equation 2.5.3)

$$\hat{\varphi}^k(d) \;=\; \nabla f(x^k)^T d + \frac{1}{2} d^T \nabla^2_{x,x}\mathcal{L}(x^k, \lambda^k) d + \gamma^k \Phi(d)\,, \quad \text{for } d \in \mathbb{R}^n\,,$$

where Φ is defined by

$$\Phi(x) \;=\; \sqrt{\sum_{i\in\mathcal{I}} [c_i(x^k) + \nabla c_i(x^k)^T d]_+^2 + \sum_{i\in\mathcal{E}} [c_i(x^k) + \nabla c_i(x^k)^T d]^2}\,, \quad \text{for } d \in \mathbb{R}^n\,.$$

It is important to clarify that while φ^k is fundamentally a function of x, for convenience, $\hat{\varphi}^k$ is characterized as a function of d. As indicated in [42], this function serves as a quadratic approximation of φ^k at the point x^k. This approximation is particularly appropriate for application within the context of a trust-region SQP method. The basic algorithm is given in Algorithm 2.

In order to fulfill the stipulation outlined in Line 4 of Algorithm 2, the trial step d^k is selected in such a manner that there is $\bar{\gamma} \geq 0$, where $\gamma \geq \bar{\gamma}$, the condition $\hat{\varphi}^k(d^k) \leq \hat{\varphi}^k(0)$ is invariably met. Based on the preceding definition of $\hat{\varphi}^k$, this outcome can be realized if the step d^k adheres to the condition

$$\sum_{i\in\mathcal{I}} [c_i(x^k) + \nabla c_i(x^k)^T d^k]_+^2 + \sum_{i\in\mathcal{E}} [c_i(x^k) + \nabla c_i(x^k)^T d^k]^2 < \sum_{i\in\mathcal{I}} [c_i(x^k)]_+^2 + \sum_{i\in\mathcal{E}} c_i(x^k)^2\,.$$

To put it in more formal terms, the step d^k serves to either enhance the feasibility of x^k for the SQP subproblem, or it serves to decrease the objective function of the SQP subproblem without compromising the previously mentioned feasibility. In real-world applications, it may occur that neither the constraint violation nor the objective function of the SQP subproblem can be minimized. It is imperative that a practical algorithm is designed to accommodate such

Algorithm 2: Basic trust-region SQP method

Data: Objective function f, constraint functions $\{c_i\}_{i\in\mathcal{I}\cup\mathcal{E}}$, initial guess $x^0 \in \mathbb{R}^n$, estimated Lagrange multiplier $\lambda^0 = [\lambda_i^0]_{i\in\mathcal{I}\cup\mathcal{E}}^T$, initial trust-region radius $\Delta^0 > 0$, and parameters $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \theta_1 < 1 < \theta_2$

1 Set the penalty parameter $\gamma^{-1} \to 0$

2 **for** $k = 0, 1, \ldots$ *until convergence* **do**

3 Set the trial step d^k to an approximate solution to

$$\min_{x\in\mathbb{R}^n} \quad \nabla f(x^k)^T d + \frac{1}{2} d^T \nabla^2_{x,x} \mathcal{L}(x^k, \lambda^k) \tag{2.5.11a}$$

$$\text{s.t.} \quad c_i(x^k) + \nabla c_i(x^k)^T d \leq 0, \quad i \in \mathcal{I}\,, \tag{2.5.11b}$$

$$c_i(x^k) + \nabla c_i(x^k)^T d = 0, \quad i \in \mathcal{E}\,, \tag{2.5.11c}$$

$$||d|| \leq \Delta^k\,, \tag{2.5.11d}$$

4 Pick a penalty parameter $\gamma^k \geq \max\{\gamma^{k-1}, ||\lambda^k||\}$ providing $\hat{\varphi}^k(d^k) < \hat{\varphi}^k(0)$

5 Evaluate the trust-region ratio

$$\rho^k \leftarrow \frac{\varphi^k(x^k) - \varphi^k(x^k + d^k)}{\hat{\varphi}^k(0) - \hat{\varphi}^k(d^k)}$$

6 **if** $\rho^k \geq 0$ **then**

7 Update the trial point $x^{k+1} \leftarrow x^k + d^k$

8 **else**

9 Retain the trial point $x^{k+1} \leftarrow x^k$

10 **end**

11 Estimate the Lagrangian multiplier $\lambda^{k+1} = [\lambda_i^{k+1}]_{i\in\mathcal{I}\cup\mathcal{E}}^T$

12 Update the trust-region radius

$$\Delta^{k+1} \leftarrow \begin{cases} \theta_1 \Delta^k\,, & \text{if } \rho^k \leq \eta_1\,, \\ \Delta^k\,, & \text{if } \rho_1 < \rho^k \leq \rho_2\,, \\ \theta_2 \Delta^k\,, & \text{otherwise} \end{cases}$$

13 **end**

scenarios. However, from a theoretical perspective, such situations can be circumvented by imposing some modest assumptions [122].

A pivotal aspect is subtly embedded within Line 3 of Algorithm 2: in the event of infeasibility of the trust-region subproblem (cf. Equation 2.5.11), what should d^k aim to approximate? Given that constraints (cf. Equation 2.5.11b) and (cf. Equation 2.5.11d) are linear approximations of constraints (cf. Equation 2.5.1b) and (cf. Equation 2.5.1c), the feasibility of subproblem (cf. Equation 2.5.11) cannot be assured, even when the original problem is feasible. This underscores a potential limitation in the approximation approach and warrants further consideration for robust algorithm development. A potential resolution to this dilemma involves the formulation of d^k as a composite-step. This implies that d^k constitutes the sum of two distinct steps. The first of these, a normal step denoted as n^k, is designed with the objective of mitigating the violation of constraints. The second, a tangential step or t^k, is targeted at reducing the objective function as expressed in Equation (2.5.11a), whilst simultaneously avoiding an escalation in the

violation of linearized constraints. More precisely, the normal step n^k represents either an exact or an approximate resolution of the normal subproblem

$$\min_{d\in\mathbb{R}^n} \quad \sum_{i\in\mathcal{E}}[c_i(x^k)+\nabla c_i(x^k)^T d]^2 \tag{2.5.12a}$$

$$\text{s.t.} \quad ||d|| \leq \zeta\Delta^k \tag{2.5.12b}$$

for some $\zeta \in [0,1]$. Additionally, the tangential step t^k solves the tangential subproblem either exactly or approximately. Let $\mathcal{V}$ represent a particular set with the property that for $d \in \mathcal{V}$, the term $n^k + d$ does not exceed the linearized constraint violation of n^k. The precise formulation of $\mathcal{V}$ is contingent on the method employed for quantifying the violation of constraints:

$$\min_{d\in\mathbb{R}^n} \quad [\nabla f(x^k)+\nabla^2_{x,x}\mathcal{L}(x^k,\lambda^k)n^k]^T d+\frac{1}{2}d^T\nabla^2_{x,x}\mathcal{L}(x^k,\lambda^k)d \tag{2.5.13a}$$

$$\text{s.t.} \quad ||n^k+d|| \leq \Delta^k\,, \tag{2.5.13b}$$

$$d \in \mathcal{V}\,. \tag{2.5.13c}$$

Some methods that are covered by this composite-step framework include the Bryrd-Omojokun approach [15, 112], the Vardi approach [156] and the Celis-Dennis-Tapia approach [19].

2.5.2 Derivative-free optimization

Derivative-free optimization algorithms constitute a vital component in the landscape of optimization methodologies. Derivative-free algorithms, offer significant utility in situations where gradient information is unavailable, such as when handling black-box functions. While it is feasible to approximate gradients utilizing finite differences, these approximations might have significant inaccuracies. Contrastingly, derivative-based algorithms necessitate a higher degree of user expertise due to their inherent complexity [106]. The setup and execution of these algorithms require a more detailed and effort-intensive approach. In general, derivative-free algorithms possess a relative ease of implementation, facilitating expedited initiation of their operation. However, they perform at lesser levels of efficiency, a trait that is more obvious as the problem's dimensionality increases [74].

A salient benefit of derivative-free algorithms is their ability to operate independently of the assumption of function continuity. On the other hand, derivative-based algorithms necessitate the presence of function smoothness, a prerequisite integral to the derivation of optimality conditions. This requirement holds true for both unconstrained and constrained functions. More specifically, the KKT conditions, necessitate that the function demonstrate continuity in its objective value, Jacobian, and Hessian [54]. This continuity must be maintained at least within a minimal range of the optimal point, ensuring the feasibility and validity of the optimization process.

In instances where the gradient at the optimal point is discontinuous and consequently undefined, the KKT conditions fail to hold validity. The requirement of gradient continuity is less strict when considered away from optimal points [167]. While derivative-based algorithms operate under similar assumptions of continuity, they generally exhibit tolerance towards scattered discontinuities, provided these are located away from an optimal point. However, for functions characterized by excessive numerical noise and discontinuities, derivative-free algorithms may represent the sole viable option [80].

The decision to employ a derivative-based or derivative-free algorithm demands an array of factors, many of which are frequently misunderstood. A prevalent misconception pertains to the characteristic of multimodality, often invoked as justification for selecting derivative-free methods [85]. The multimodality of the design space may originate from an objective function that exhibits

multiple local minima. Conversely, in a constrained problem scenario, multimodality can stem from the imposition of constraints that delineate disconnected or nonconvex feasible regions.

Certain derivative-free optimization techniques incorporate a global search component, augmenting the probability of attaining the global minimum. This attribute designates these derivative-free methods as prevalent choices for tackling multimodal problems [78]. However, it is critical to note that not all derivative-free methods employ global search strategies; some are limited to local search operations.

Moreover, while derivative-based methods inherently operate on local search principles, they are frequently integrated with global search strategies. It is not universally accurate to assert that a global search, derivative-free method is more likely to discover a global optimum than a multi-start derivative-based technique. As with any analysis, problem-specific empirical evaluation is indispensable.

The utilization of derivative-free methods frequently arises in scenarios involving discrete design variables. Given that the concept of a derivative concerning a discrete variable does not hold validity, the direct application of gradient-based algorithms becomes infeasible [89]. Consequently, the exploration of alternatives, such as derivative-free methods, becomes a necessary step in such cases.

In the following, two derivative-free optimization methods utilized in this study are introduced, namely *COBYLA* (Constrained Optimization BY Linear Approximation) [120] and *COBYQA* (Constrained Optimization BY Quadratic Approximation) [126].

2.5.2.1 COBYLA

The COBYLA algorithm is applied to address the optimization problem denoted by Equation (2.5.1), under the condition that the constraint functions, represented as c_i for $i \in \mathcal{I}$, are nonlinear functions with unknown derivatives. This implies that solely the values of these constraint functions are available for computation, precluding the use of derivative information in the optimization process.

During the k-th iteration, the COBYLA algorithm constructs linear interpolations of both the objective and constraint functions. These interpolations are based on the interpolation set $\mathcal{Y}^k \in \mathbb{R}^n$. The set $\mathcal{Y}$ comprises of $n+1$ points, which are iteratively updated as the algorithm progresses.

Following the establishment of linear models, denoted as $\hat{c_i}^k$, for the constraint functions c_i, where $i \in \mathcal{I}$, the subsequent phase requires the resolution of the associated trust-region subproblem

$$\min_{x \in \mathbb{R}^n} \quad \hat{f}^k(x) \tag{2.5.14a}$$

$$\text{s.t.} \quad \hat{c_i}^k(x) \leq 0 \quad i \in \mathcal{I}\,, \tag{2.5.14b}$$

$$||x - x^k|| \leq \Delta^k\,. \tag{2.5.14c}$$

The resolution of this problem can be achieved by replacing Δ^k with a constant that is progressively increasing from zero to Δ^k. This approach will generate a piecewise linear path starting from x^k and ending at the solution of this problem. To accurately identify this solution, the trust-region subproblem solver of COBYLA begins this path by iteratively updating the active sets of the linear constraints as expressed in Equation 2.5.14b.

Nevertheless, potential contradictions may arise between the linear constraints (cf. Equation 2.5.14b) and the trust-region constraint (cf. Equation 2.5.14c). When such a contradiction arises,

the trial point is selected to provide an approximate solution:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \max_{i\in\mathcal{I}} [\hat{c_i}^k(x)]_+ \\ \text{s.t.} \quad & ||x - x^k|| \leq \Delta^k \,. \end{aligned}$$

The operator $[\cdot]_+$ is designed to extract the positive component of a given numerical input. The utilization of this operator within the methodology aims to reduce the ℓ_∞-norm constraint violation that arises from the linearized constraints within the defined trust-region. This operation is integral to ensuring the constraints are adequately satisfied within the trust-region during the optimization process.

Preserving the appropriate geometry of $\mathcal{Y}^k$ is critical for the accuracy of model generation. If the geometry of $\mathcal{Y}^k$ proves insufficient in facilitating accurate models, the COBYLA algorithm initiates a process of point substitution. Specifically, it removes a point from $\mathcal{Y}^k$ and incorporates a new one, selected along the direction orthogonal to the face of $\mathcal{Y}^k$. This face is construed as a simplex, and it is counterpoised to the location of the point that was previously removed. This substitution process is likely to increase the volume of the simplex generated by the interpolation set, consequently enhancing the conditioning of the interpolation system.

2.5.2.2 COBYQA

COBYQA, a derivative-free, trust-region SQP method, is strategically engineered to manage nonlinearly constrained optimization problems, which include both equality and inequality constraints [126]. A salient characteristic of COBYQA is its rigorous observance of bound constraints during its iterative point exploration procedure, given the presence of such constraints. This aspect exhibits significant advantages, especially considering that objective functions pertinent to applications with bounded constraints frequently become indeterminate when these constraints are violated.

In reference to the foundational trust-region SQP approach, it typically constructs models of the objective and constraint functions using their gradients and Hessian matrices. However, in the absence of access to such information in this study, COBYQA employs interpolation-based quadratic models for these functions. Specifically, COBYQA utilizes quadratic models derived from underdetermined interpolation predicated on the derivative-free symmetric Broyden update methodology.

At the k-th iteration, the objective function f is approximated by the quadratic model $\hat{f}_k$ that solves

$$\min_{Q\in\mathcal{P}_{2,n}} ||\nabla^2 Q - \nabla^2 \hat{f}^{k-1}||_F \quad \text{s.t.} \quad Q(y) = f(y),\, y \in \mathcal{Y}^k \,.$$

Here, $\mathcal{P}_{2,n}$ represents the space of quadratic polynomials on $\mathbb{R}^n$. The *Frobenius norm*, denoted as $||\cdot||_F$, serves as a matrix norm and $y \in \mathcal{Y}^k$ refers to a finite interpolation set that undergoes updates through iterative procedures. The requisite condition for this context is the fulfillment of specific interpolation constraints, which are elaborated below. In general,

$$n + 2 \leq \text{card}(\mathcal{Y}^k) \leq \frac{1}{2}(n+1)(n+2) \,.$$

Ensuring the well-definition of the quadratic model $\hat{f}^k$ is crucial. Analogously, the quadratic models $\hat{c_g}^k$ and $\hat{c_h}^k$, belonging to c_g and c_h, are formulated. Here, the subscript g signifies the inequality constraint, while h designates the equality constraint.

The Lagrangian form $\hat{\mathcal{L}}^k$ of COBYQA can be expressed as

$$\hat{\mathcal{L}}^k(x,\lambda,\mu) = \hat{f}^k(x) + \lambda^T \hat{c_g}^k(x) + \mu^T \hat{c_h}^k(x)\,.$$

Furthermore, given a penalty parameter $\gamma^k \geq 0$, the ℓ_2-merit function φ^k is given by

$$\varphi^k(x) = f(x) + \gamma^k \sqrt{||[c_g(x)]_+||_2^2 + ||c_h(x)||_2^2}\,.$$

Consequently, the quadratic form $\hat{\varphi}^k$ is

$$\hat{\varphi}^k(d) = \hat{f}^k(x^k) + \nabla \hat{f}^k(x^k)^T d + \frac{1}{2} d^T \nabla^2 \hat{\mathcal{L}}_k(x^k,\lambda^k,\mu^k) d +$$
$$\gamma^k \sqrt{||[\hat{c_g}^k(x^k) + \nabla \hat{c_g}^k(x^k) d]_+||_2^2 + ||\hat{c_h}^k(x^k) + \nabla \hat{c_h}^k(x^k) d||_2^2}\,,$$

where $x^k \in \mathcal{Y}^k$, represents the optimal interpolation point as determined by the merit function φ^{k-1}. Furthermore, the quantities λ^k and μ^k serve as approximate representations of the Lagrange multipliers that correspond to the inequality and equality constraints, respectively. Algorithm 3 demonstrates the framework of COBYQA.

2.5.3 Derivative-based optimization

Derivative-based optimization, as the name suggests, is a category of optimization methods that make use of derivative information. These methods are powerful and efficient, especially for smooth functions where derivative information is readily available. However, real-world optimization problems often come with constraints, which lead to the field of constrained derivative-based optimization. One of the most famous methods in constrained derivative-based optimization is the SQP method (cf. Equation 2.5.1). Another popular method is the Interior Point Method [92], which transforms the constrained problem into a series of unconstrained problems using a barrier function. The barrier function is designed to tend to infinity as the constraints are violated, thus "forcing" the solution to remain within the feasible region [147].

The Sequential Least Squares Programming (SLSQP) algorithm is a versatile and widely used method for constrained optimization [11]. It falls within the broader class of SQP methods, which are iterative techniques for solving nonlinear optimization problems, both with and without constraints. The SLSQP algorithm is especially known for its ability to handle both equality and inequality constraints. It does this by solving a sequence of optimization subproblems, each of which optimizes a quadratic model of the objective subject to a linearization of the constraints. The SQP method requires the solution of a generic quadratic programming problem. This presents a considerable challenge, especially when integrating second derivative information into the algorithm. To address this complexity, Schittkowski proposed a two-phase computation approach for the step [140]. Specifically, Schittkowski suggested substituting the quadratic programming subproblem with a linear least squares subproblem, facilitated by a robust LDL^T-factorization [77] of the inverse of the Hessian matrix $\mathbb{B}$:

$$\min_{d \in \mathbb{R}^n} ||(D^k)^{1/2}(L^k)^T d + (D^k)^{-1/2}(L^k)^{-1} \nabla f(x^k)||\,,$$

subject to Equations (2.5.11b) and (2.5.11d).

In more detail, SLSQP approximates the objective function and the constraint functions using first-order Taylor series expansions. It then solves the resulting quadratic program to find a direction of search. A line search is then conducted in this direction to find a new point that improves the objective. The process is repeated until a solution is found that satisfies the KKT conditions for optimality, or until a maximum number of iterations is reached.

Algorithm 3: COBYQA algorithm

Data: Initial trust-region radius $\Delta^0 > 0$

Set the penalty parameter $\gamma^{-1} \to 0$

Build the initial interpolation set $\mathcal{Y}^0 \in \mathbb{R}^n$

Define x^0 to a solution to $\min_{y \in \mathcal{Y}^0} \varphi^0(y)$

Estimate the Lagrange multipliers λ^0 and μ^0

for $k = 0, 1, \ldots$ *until convergence* **do**

  Compute the models $\hat{f}^k$, $\hat{c_g}^k$, $\hat{c_h}^k$

  Set the trial step d^k to an approximate solution to

$$\min_{d \in \mathbb{R}^n} \quad \hat{f}^k(x^k) + \nabla \hat{f}^k(x^k)^T d + \frac{1}{2} d^T \nabla^2 \hat{\mathcal{L}}^k(x^k, \lambda^k, \mu^k) d \tag{2.5.15a}$$

$$\text{s.t.} \quad \hat{c_g}^k(x^k) + \nabla \hat{c_g}^k(x^k) d \leq 0\,, \tag{2.5.15b}$$

$$\hat{c_h}^k(x^k) + \nabla \hat{c_h}^k(x^k) d = 0\,, \tag{2.5.15c}$$

$$l \leq x^k + d \leq u\,, \tag{2.5.15d}$$

$$||d|| \leq \Delta^k \tag{2.5.15e}$$

  Pick a penalty parameter $\gamma^k \geq \max\{\gamma^{k-1}, \sqrt{||\lambda^k||^2 + ||\mu^k||^2}\}$ providing $\hat{\varphi}^k(d^k) < \hat{\varphi}^k(0)$

  Evaluate the trust-region ratio

$$\rho^k \leftarrow \frac{\varphi^k(x^k) - \varphi^k(x^k + d^k)}{\hat{\varphi}^k(0) - \hat{\varphi}^k(d^k)}$$

  if $\rho^k > 0$ **then**

    Choose a point $\bar{y} \in \mathcal{Y}^k$ to remove from $\mathcal{Y}^k$

  else

    Choose a point $\bar{y} \in \mathcal{Y}^k \setminus \{x^k\}$ to remove from $\mathcal{Y}^k$

  end

  Update the interpolation set $\mathcal{Y}^{k+1} \leftarrow (\mathcal{Y}^k \setminus \{\bar{y}\}) \cup \{x^k + d^k\}$

  Update the current iterate x^{k+1} to a solution $\min_{y \in \mathcal{Y}^{k+1}} \varphi^k(y)$

  Estimate the Lagrange multipliers λ^{k+1} and μ^{k+1}

  Update the trust-region radius Δ^{k+1}

  Improve the geometry of $\mathcal{Y}^{k+1}$ if necessary

end

The algorithm begins by determining a feasible point, which satisfies all the constraints. From this point, it computes search directions by solving the quadratic subproblem. The algorithm then updates the current point by moving along the search direction. This iterative process continues until the algorithm converges to an optimal solution or until a stopping criterion is met. One of the key features of SLSQP is its use of a trust-region strategy (cf. section 2.5.1.3), which limits the step size in each iteration to ensure that the approximations remain valid. This makes the algorithm robust and efficient for a wide range of optimization problems.

2.5.4 Surrogate-based optimization

This work introduces a methodology that utilizes distinct, independently trained differentiable ML models as surrogate substitutes during the optimization process. The forthcoming section will delve into an in-depth discourse on surrogate-based optimization (SBO) [125, 142].

A surrogate model, alternatively termed a response surface model or a metamodel, serves as an approximated representation of a functional output, mirroring a curve fit to an underlying dataset [20]. The primary objective in constructing a surrogate model is to develop a computationally efficient substitute for the original function, while ensuring the preservation of adequate accuracy even in regions removed from known data points. This surrogate model thus provides a balance between computational efficiency and model fidelity, facilitating expedited computations without compromising the integrity of the predictive capability.

Within the context of optimization, the surrogate could encompass the entirety of the optimization model (that is, the inputs constitute design variables, and the outputs represent objective and constraint functions), or alternatively, the surrogate may merely form a component of the comprehensive model [99]. SBO exhibits a more targeted approach compared to the expansive domain of surrogate modeling. Rather than striving for a globally precise surrogate, the objective of SBO is to develop a surrogate model that possesses sufficient accuracy to guide the optimization algorithm towards the true optimal solution.

Surrogate models serve a significant role in numerous circumstances. A prominent scenario arises when the primary model faces substantial computational costs. Although the invocation of surrogate models is associated with minimal computational expense, their construction necessitates multiple evaluations of the primary model. It is plausible that the number of evaluations required to construct a surrogate model with satisfactory accuracy is fewer than those required for the direct optimization of the primary model [44]. Under such circumstances, SBO might present a viable alternative. The construction of a surrogate model becomes even more compelling when there is potential for its application across multiple optimization tasks [59].

Utilizing surrogate modeling has demonstrated effectiveness in addressing models characterized by intrinsic noise, as it synthesizes a refined representation of such noisy datasets. This attribute holds particular utility in the context of gradient-based optimization. A situation useful to both costly evaluation and noise-ridden output is the instance of experimental data. In circumstances where the model data is experimental and the optimizer lacks the capacity for automated querying of the experiment, the construction of a surrogate model predicated on the experimental data becomes feasible. Subsequently, the optimizer may query this surrogate model during the optimization process. Surrogate models also prove beneficial when an understanding of the design space is desired, which involves discerning how the objective and constraints (outputs) fluctuate relative to the design variables (inputs). The construction of a continuous model over discrete data yields functional relationships that can be visualized with enhanced efficacy.

The SBO process starts with the application of sampling techniques to designate initial points for function evaluation or experimental conduction. These points are commonly known as *training data*. Following this initial step, a surrogate model is established utilizing the sampled points.

The subsequent stage involves executing optimization by interrogating the surrogate model. The optimization outcome informs the inclusion of supplementary points in the sample and consequent surrogate reconstruction - a process often termed as *infill*. This iterative sequence persists until the attainment of a predefined convergence criterion or the execution of a maximum number of iterations.

It is noteworthy that in some methodologies, the infill step is discarded; the surrogate model is constructed in its entirety at the outset and is not updated thereafter.

Note that the methodology introduced in chapter 5 employs surrogates which are not directly

fitted to the objective function as described above. Instead, we will consider the case that a differentiable data-driven model can be obtained from the same observational training data as the non-differentiable objective ML model, which is a common occurrence in cases where the best available model – and thus the one used as the objective – is not sufficiently regular, whereas differentiable models such as NNs are still available, even if they are less accurate.

2.6 Cloud computing

In the following, some tools for the development and deployment of software in cloud computing will be discussed, including the general architecture of containers (a common option for deploying and managing software in the cloud) and a description of *Kubernetes*, a container orchestration tool that can be used to monitor and manage container lifecycles in more complex environments.
.

2.6.1 Docker containers

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another [67]. Containers are made possible by process isolation and virtualization capabilities built into the Linux kernel. These capabilities - such as control groups for allocating resources among process, and namespaces for restricting a processes access or visibility into other resources or areas of the system - enable multiple application components to share the resources of a single instance of the host operating system in much the same way that a hypervisor enables multiple virtual machines (VM) to share the CPU, memory and other resources of a single hardware server.

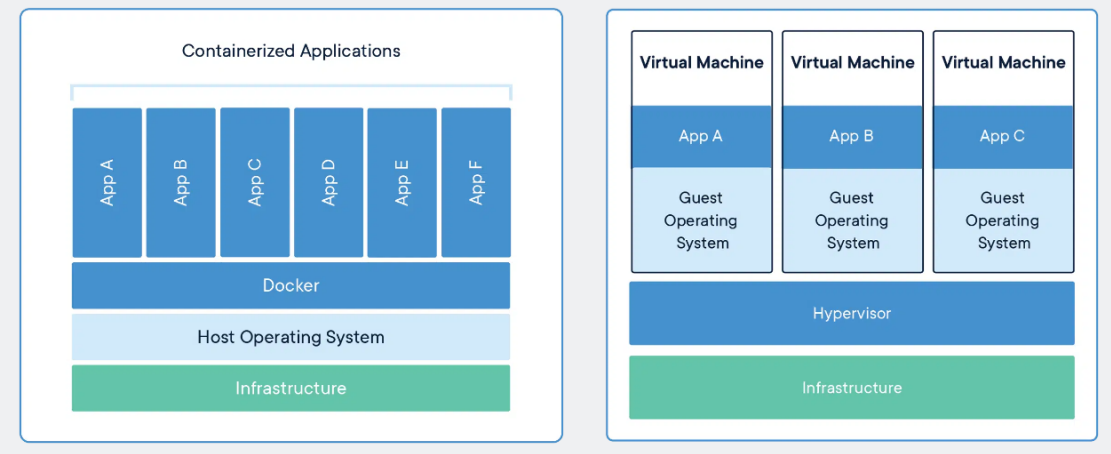

Figure 2.11: The comparison of containers and virtual machines. Containers are an abstraction at the top layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Virtual machines are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, neccessary libraries [67].

As a result, container offers all the functionality and benefits of VMs. Containers can run anywhere, as long as the container engine supports the underlying operating system - it is possible

to run containers on Linux, Windows, MacOs, and many other operating systems. Containers can run in virtual machines, on bare metal servers, locally on a developer's laptop. They can easily be moved between op-premise machines and public cloud, and across all these environments, continue to work consistently.

Containers do not require a separate operating system and therefore use few resources. VMs are typically a few GB in size, but containers commonly weigh only tens of megabytes, making it possible for a server to run many more containers than VMs [146]. Containers require less hardware, making it possible to increase server density and reduce data center or cloud costs.

Multiple containers can be run on the same server, while ensuring they are completely isolated from each other. When containers crash, or application within them fail, other container running the same application can continue to run as usual. Container isolation also has security benefits, as long as containers are securely configured to prevent attackers from gaining access to the host operating system.

Containers are a lightweight package that everything needed to run, including its own operating system, code, dependencies and libraries. The lightweight design of containers ensures that new applications can be released and upgraded easily. This often leads to a quicker development process and speeds up the time to deployment.

Containers make it easy to horizontally scale distributed applications. Multiple, identical containers can be added to create additional instances of the same application. Container orchestrators can perform intelligent scaling, running only the necessary number of containers to serve application loads, considering the resources available to the container cluster. Containers allow developers to create predictable runtime environments, including all software dependencies required by an application component, isolated from other applications on the same machine. From a development point of view, this guarantees that the component they are working on can be deployed consistently, no matter where it is deployed. In a containerized architecture, developers and operations teams spend less time debugging and diagnosing environmental differences, and can spend their time building and delivering new product features. In addition, developers can test and optimize containers, reducing errors and adapting them to production environments.

Docker is one of the most widely used container tools. A docker uses a client-server model and comprise of the following components: Docker daemon is responsible for all container related actions and receives commands. A docker client is how user interact with docker. The docker client can reside on the same host as the daemon or a remote host. Docker objects are used to assemble an application. Apart from networks, volumes, services, and other objects the two main requisite objects are: Images, the read-only template used to build containers. Images are used to store and ship applications. Containers, are encapsulated environments in which applications are run. A container is defined by the image and configuration options. At a lower level, we have *containerd*, which is a core container runtime that initiates, and supervise container performance. Finally, docker registries are locations from where we store and download images. Figure 2.12 shows the architecture of docker.

More specifically, every docker container starts with a simple text file containing instructions for how to build the docker container. *Dockerfile* automates the process of docker image creation. It is essentially a list of command-line interface instructions that docker engine will run in order to assemble the image. Docker images contain executable application source code as well as the tools, libraries, and dependencies that the application code needs to run as a container. Multiple docker images can be created from a single base image, and they will share the commonalities of their stack. Docker images are made up of layers, and each layer corresponds to a version of the image. Whenever a developer makes changes to the image, a new top layer is created, and this top layer replaces the previous top layer as the current version of the image. Previous layers are

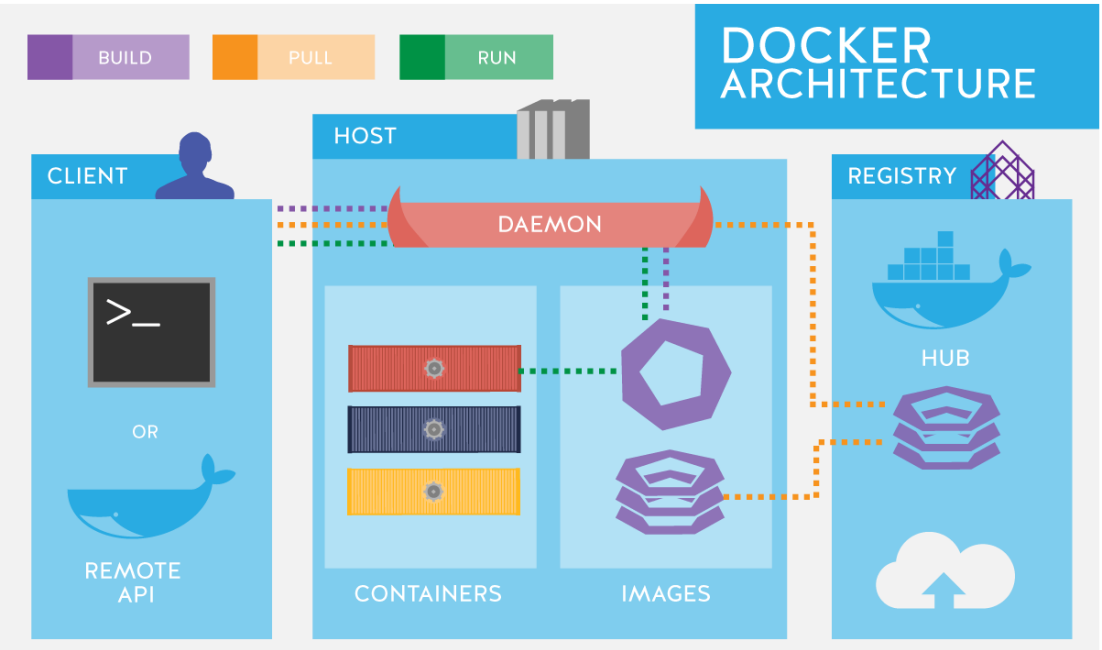

Figure 2.12: Docker architecture [110].

saved for rollbacks or to be re-used in other projects. Each time a container is created from a docker image, yet another new layer called the container layer is created. Changes made to the container - such as the addition or deletion of files - are saved to the container layer only and exist only while the container is running. This iterative image creation process enables increased overall efficiency since multiple live container instances can run from just a single base image, and when they do so, they leverage a common task.

2.6.2 Kubernetes

As a powerful open-source orchestration technology, *Kubernetes* assists in managing containerized applications and microservices over a distributed cluster of compute machines [17]. Through the use of numerous crucial features, including REST APIs and declarative templates that can handle the whole lifecycle, Kubernetes aims to hide the complexity of maintaining containers. Client-server architecture is the cornerstone of the Kubernetes architecture. The server side of Kubernetes is the known as the control plane. By default, there is a single control plane server that acts as a controlling node and point of contact. This server consists of components including the api server, storage, controller manager, scheduler and Kubernetes DNS server. The client side of Kubernetes comprises the cluster nodes—these are machines on which Kubernetes can run containers. Node components include the kubelet and kube-proxy on top of docker. A Kubernetes cluster is a collection of nodes on which we can run workloads. A node can be a physical machine, a VM, or managed by a serverless computing system. Clusters are managed by the Kubernetes control plane, which coordinates container activity on nodes and moves the cluster towards the user's desired state.

A node is a Kubernetes worker machine managed by the control plane, which can run one or more pods. Master note serves as the starting point for all administrative activities and is

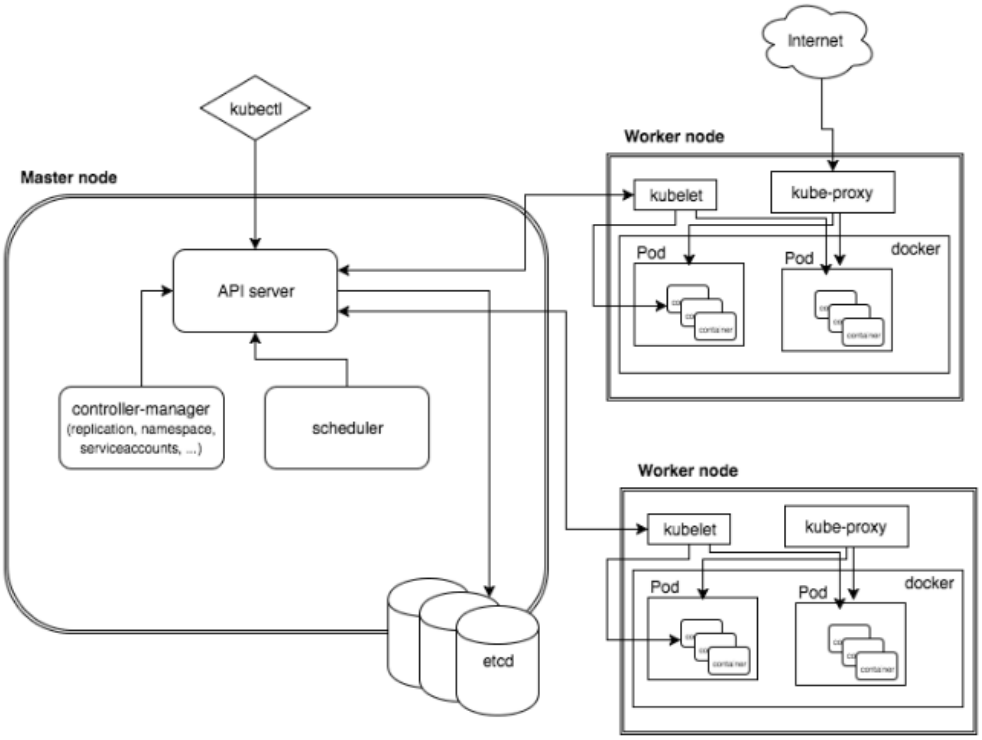

Figure 2.13: High level Kubernetes architecture diagram showing a cluster with a master and two worker nodes [162].

the charge of controlling the Kubernetes cluster architecture. A master node inside Kubernetes has three main components. The first one is the Kubernetes API server, which is where all the administrative tasks will be performed. Kubernetes delivers REST commands that will process and validate the requests. Upon requests, the cluster's resulting state will be stored based on the distributed key value. The second component is the scheduler, which schedules the tasks to specified worker nodes. Besides, each worker node will store resource usage information. All the work will be scheduled in the form of services and pods. Before the tasks is scheduled, the scheduler will consider the service requirements quality, affinity, anti-affinity, data locality, etc. The *control manager* is a controller, which is the third component of a master node; it is a daemon that adjusts the Kubernetes cluster. The Kubernetes cluster is used to manage multiple non-terminating controls loops. Controller also handles node garbage collection, event garbage collection and cascading-deletion garbage collection. It also provides lifecycle functions such as namespace creation. A controller, in essence, examines the ideal state of a controlled object, but it also employs an API server to monitor and manage its present state. If an object's intended state is not satisfied, the control loop will guarantee that the current and desired states are leveled by taking precise measures to achieve this aim.

On the other hand, worker node runs applications via pods, and master node controls the pods. A worker node consists of four components. The first one is container runtime, worker node requires a container runtime like Docker to manage and run the container's lifecycle. Kubelet

interacts with the master node and runs on worker nodes. It retrieves pod specifications from an API server. It also runs the corresponding containers indicated in healthy and active pods. Advisor is the third component of a worker node. It serves to analyze all the metrics for network usage, file, CPU, and memory for every container that runs on a specified node. Kube-proxy operates on each node and interacts with each host sub-netting individually to ensure that all services are accessible to external parties. It also works as a load balancer and network proxy for any services running on a worker node. Furthermore, kube-proxy will handle network routing for UDP and TCP traffic. For kube-proxy to reach service endpoints, it will create different routes.

A pod is the smallest unit of management in a Kubernetes cluster. It represents one or more containers that constitute a functional component of an application. Pods encapsulate containers, storage resources, unique network, and other configurations defining how container should run. In Kubernetes, a service is an abstraction that represents a set of pods which represent an application or component and includes access policies for those pods. Kubernetes guarantees the availability of a given pod and its replicas, but the actual pod instances running within a service are temporary and may be replaced by others. This means that other pods that need to communicate with this application or component rely on the IP address of underlying pod.

Similar to a container volume in docker, a Kubernetes volume applies to a whole pod and is mounted on all containers in the pod. Kubernetes guarantees data is preserved across container restarts. The volume will be removed only when the pod gets destroyed. Also, a pod can have multiple volumes associated. Namespace is another important concept. It is a virtual cluster intended for environments with many users spread across multiple teams or projects, for isolation of concerns. Resources inside a namespace must be unique and cannot access resources in a different namespace. Also, a namespace can be allocated a resource quota to avoid consuming more than its share of the physical cluster's overall resources. Finally, deployment in Kubernetes describes the desired state of a pod or a replica set. The deployment controller then gradually updates the environment until the current state matches the desired state specified in the deployment file.

In practice, Kubernetes is the standard for container orchestration, commonly used to orchestrate docker containers. Kubernetes was designed to support the features required by highly available distributed systems, such as auto-scaling, high availability, security and portability.

Kubernetes allows pods to be scaled horizontally based on CPU use. The CPU utilization threshold can be set, and if it is achieved, Kubernetes will immediately start fresh pods. For example, if the CPU utilization threshold is 70% but the application grows to 220%, three more pods will be deployed gradually, bringing the average CPU utilization down to 70%. When a single application has several pods, Kubernetes provides load balancing capacity across all of them. Stateful pods can be horizontally scaled in Kubernetes. A stateful set is similar to a deployment, but it ensures that storage is persistent and reliable even when a pod is removed.

Kubernetes aims to provide high availability for both applications and infrastructure. Replica sets ensure that the desired number of stateless pod replicas are running for a given application. Stateful sets and stateful pods serve the same purpose. Kubernetes provides a variety of distributed storage options at the infrastructure level. Adding a dependable, available storage layer to Kubernetes ensures that stateful applications are always available. To achieve improved availability, each of the master components can be set for multi-node replication.

Kubernetes protects clusters, applications, and networks on various layers. Transport layer security is used to protect API endpoints (TLS). On the cluster, only authenticated users can perform operations. Kubernetes secrets can store sensitive information per cluster at the application level. It is worth noting that secrets can be accessed by any pod in the same cluster. In a deployment, network policies for pod access can be defined. A network policy governs how pods

communicate with one another and with other network endpoints.

Finally, Kubernetes' portability presents itself in terms of operating systems, processor architectures, cloud providers, and the addition of other container runtimes other than docker. It can also support workloads across hybrid or multi-cloud systems thanks to the federation concept. Within a single cloud provider, this also offers availability zone fault tolerance.

2.7 Summary

Given the criticality of scrap procurement to the foundry industry's operational efficiency and sustainability, the complexities and challenges embedded in optimizing this process need to be discussed. The task of scrap procurement optimization is multifaceted, encompassing aspects such as cost minimization, quality assurance, and environmental sustainability. These intricacies call for an advanced, data-driven approach, thereby introducing the concept of data-driven optimization.

The suggested way to solve these problems is by using ML to get a strong grasp of this approach. In the quest to optimize scrap procurement, the role of numerical optimization needs to be brought into focus, i.e. the mathematical discipline that seeks to find the best solution – in terms of a specified objective function – among a set of available alternatives, subject to certain constraints. Whereas classical gradient-based methods can be considered very effective and, in many cases, highly efficient, the challenges associated with gradient-free optimization motivate the use of differentiable surrogates, if they are available.

Furthermore, current state-of-the-art techniques in cloud computing can simplify employing and updating various kinds of software. In particular, cloud-based methods can be used for providing software tools to end users in the foundry industry more easily.

Chapter 3

Platform for scrap procurement

One of the many challenges in improving the efficiency of scrap procurement in the foundry industry is the lack of unified supply data: in order to fully utilize ML and optimization techniques to make an informed purchase decision, it is necessary to collect and compare the availability and pricing of different scrap types across suppliers. As part of this work, an online platform was developed which connects foundries and scrap suppliers, provides a unified interface for data exchange and integrates ML and optimization tools that can automatically select the optimal composition of scrap for a batch with respect to the total cost, including the transport and energy cost, for a given chemical target composition of the steel that is to be produced. The developed platform therefore represents a comprehensive solution serving the foundry industry's needs concerning the optimization of scrap procurement.

In the following, the key components of the platform as well as some of the implementation details are presented. The described software was developed as part of the BMBF funded research project OPTIRODIG (project ID 033R247C).

3.1 Introduction to the platform

The first primary component of the platform is the *Scrap Dealer*. An important role of this module, as depicted in Figure 3.1, pertains to the provision of an interface enabling users to submit and modify data concerning their respective scrap dealers. This includes elements such as the communication form and individualized records of cumulative scrap quantities pertaining to each dealer. The component designated as the *Scrap* module functions as the underlying data repository for the simulation process. This module encapsulates various types of scrap utilized within the foundry sector, including but not limited to domestic scraps, external scraps, and alloy-based scraps. In the case of external scraps, the *Scrap* module integrates additional details such as associated costs, quantities, and affiliated merchants. Pertinent data regarding domestic scraps specific to individual foundry industries are also incorporated within this component for comparative purposes, as illustrated in Figure 3.2.

Of course, the most important task of foundry industry is to produce steel. In this context, the metallic elements are subjected to high temperatures until they achieve a liquefied state. Subsequent to this, the molten metal is transferred into a purposefully designed mold. The process continues as the heated metal undergoes a transition from a liquid to a solid state within the confines of the mold. Once solidified, the metal is extracted from the mold, resulting in a finished product or component. These components can be incorporated into an array of products, some of which are integral to contemporary life. Depending on the end product,

certain mechanical or chemical properties of the produced steel are desirable or even necessary. In order to ensure these properties, the *chemical composition* of the steel needs to be controlled. The target composition will act as the equality constraints for Equation (2.5.1c) in the cost optimization described below.

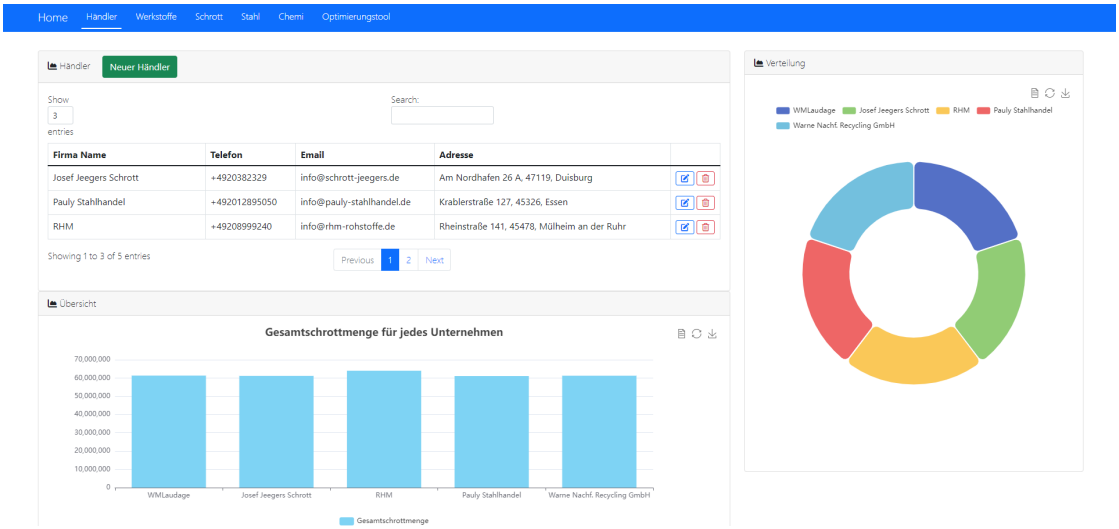

Figure 3.1: The scrap dealer component of the platform.

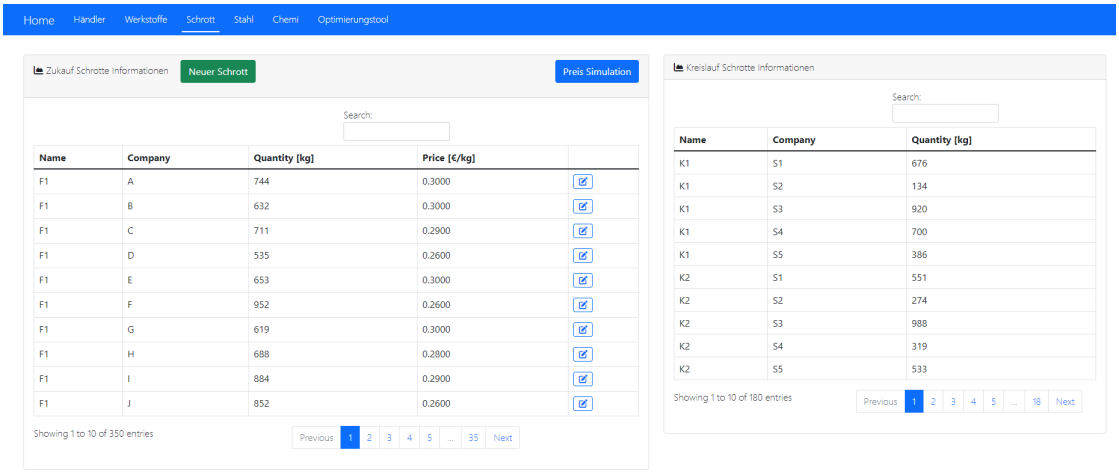

Figure 3.2: The scrap component of the platform.

Some common types of steel and their chemical composition are listed in Table 3.1. Steel of type **1.2379** represents a high-alloy steel variant that facilitates thorough hardening while exhibiting moderate machinability characteristics. It is exceptionally resistant to wear and demonstrates minimal distortion or warping. This steel also offers beneficial attributes such as commendable dimensional stability, resilience, and the ability to harden throughout its mass. The utilization of this steel extends to mold plates and inserts, in addition to wear plates and

cutting dies, with an enhancement in their resistance to wear and tear. The material known as steel **1.2344** is frequently utilized in various applications due to its distinctive characteristics. Its composition, classified as a high-alloy hot-work steel, imparts notable heat resistance and wear resistance, contributing to its longevity in demanding environments. Additionally, its hardness, thermal conductivity, and resistance to hot cracks further enhance its desirability. In terms of application, steel **1.2344** serves as the standard material for hot-work tools, lending itself particularly well to use in the manufacture of extrusion molds and dies. Moreover, it is prevalently used in the fabrication of tools required for plastic processing, further broadening its scope of application. The material designated as steel **1.3343** is frequently used in various applications due to its distinctive characteristics. It's a type of high-speed steel that exhibits exceptional resistance to abrasive wear, complemented by high toughness and compressive strength. Additionally, this material demonstrates high heat resistance, making it ideal for thorough hardening processes. Steel **1.3343** is regularly utilized in the creation of blocks for erosion procedures. It also finds use in the production of cold forming tools, including but not limited to cutting instruments, fine blanking mechanisms, and punches and dies used in impact extrusion. Furthermore, this material is incorporated into inserts that require significant wear resistance. The material identified as steel **1.2343** is frequently utilized due to its distinct properties. It is characterized by its high-alloy composition which contributes to its hot-workability, exhibiting robust toughness and superior resistance to heat. Moreover, it demonstrates resistance against hot cracks, supplemented with commendable thermal conductivity. With regard to practical usage, the **1.2343** steel is principally utilized in the construction of mold plates and inserts, critical elements within the machinery used for plastic injection molding. The variant of this steel produced through Electroslag Remelting process is notably advantageous in the field of die casting.

Table 3.1: Chemical composition of 4 steels in %.

Steel	**C**	**Si**	**Mn**	**Cr**	**Mo**	**V**
1.2379	1.53	0.30	0.35	12.00	0.80	0.80
1.2344	0.40	1.00	0.00	5.30	1.40	1.00
1.3343	0.90	0.30	0.350	4.00	5.00	1.90
1.2343	0.38	1.00	0.40	5.30	1.20	0.40

Numerous additional chemical formulations of steels can be found in the component dedicated to Steel on the platform.

3.2 Software architecture

In the process of developing the architecture of the platform, the primary hurdle lies in its successful deployment within a cloud-based infrastructure. Given that the platform functions as a Python-oriented web application, it requires the ability to manage an array of distinct scenarios. These scenarios include, but are not limited to, handling the vital request/response cycle that requires minimized latency, such as during user authentication procedures, as well as managing longer duration background tasks, where an increased latency is permissible due to the indirect effect on the user experience, such as during data processing and simulation tasks. The web framework selected for this project is Django, a high-level Python framework known for facilitating swift development and promoting a clean, pragmatic design [31].

Django benefits from a Model-View-Template (MVT) architecture, which is a software design pattern. As the name suggests, it is made up of three key components: Model, View, and

Template. These three layers are responsible for multiple tasks and can be utilized independently. Figure 3.3 illustrates the interactions of the MVT architecture in Django.

The *Model* helps to handle databases. It is a data access layer, which contains the required fields and behaviors of the data should be stored. In this work, PostgreSQL [58] was used as database. A model is a Python class that has no knowledge of the other Django layers. The only way to interact across layers is through an application programming interface (API). Models assist developers in creating, reading, updating and deleting items in the original database (CRUD operations). They also store business logic, custom methods, properties, and other data manipulation-related items.

The *View* is used to execute business logic, interact with a model, and render a template. The view retrieves information from a model. Then, it either grants each template access to particular data to be shown, or it processes data beforehand. It receives HTTP requests, implements business logic given by Python classes and functions, and responds to client requests through HTTP.

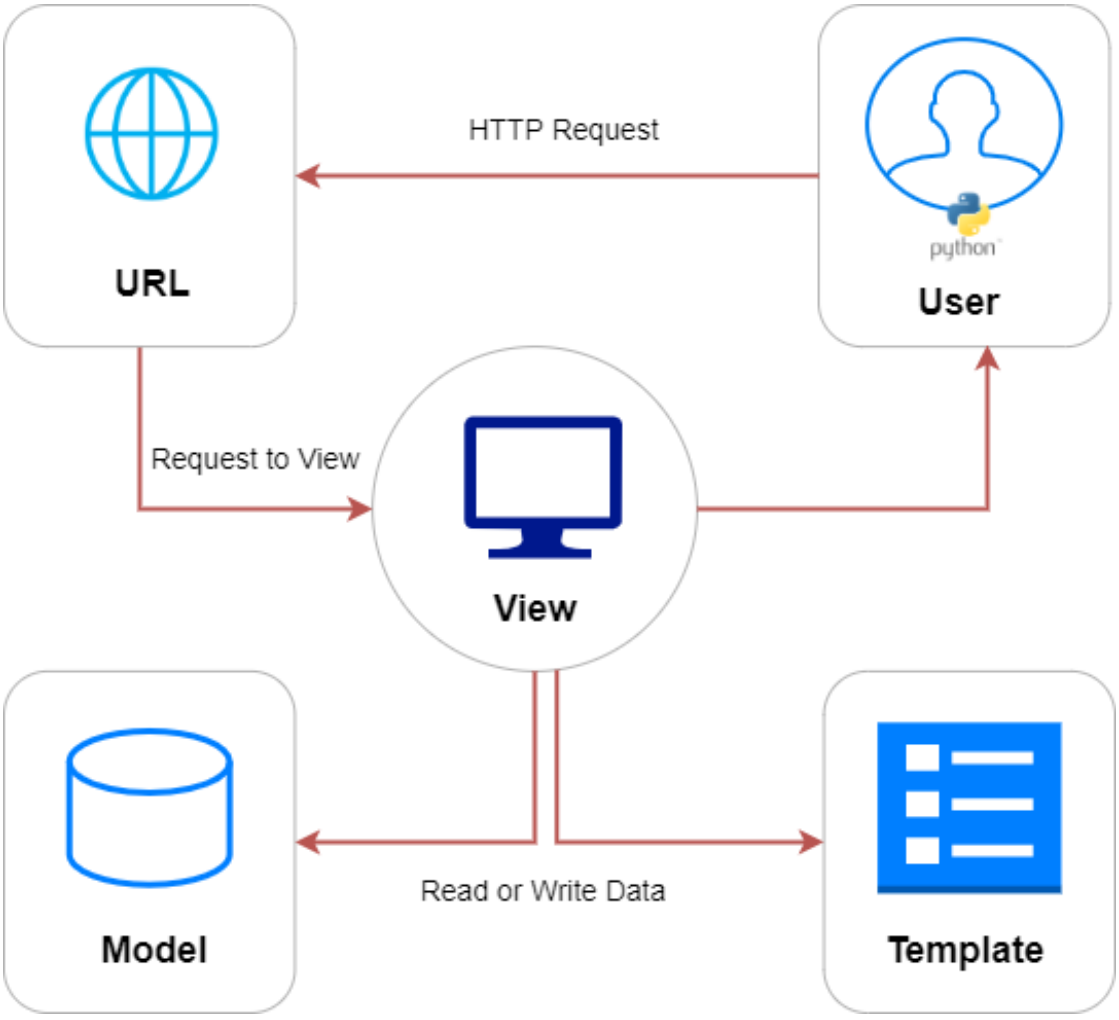

Figure 3.3: Django MVT architecture.

Finally, the *Template* is a presentation layer that manages the whole user interface. These are files that include HTML code is used to render data. These file's contents might be static or dynamic. A template is used to present data because it has no business logic. The MVT architecture take cares of the separation of Django application development. The model will work with data, the view will work with logic, and the template will work with layouts. In addition, MVT architecture makes the whole process of transmitting over the internet simpler and faster.

Django ORM (Object Relational Mapper) provides an intuitive and powerful object-oriented mechanism to communicate with the database. ORM is a library automatically converts data from databases into objects that can be utilized in application code. To create tables and insert data, there is not need to write specialized structured query language (SQL) queries. When a class is defined, ORM automatically generates a table for the class. This table includes fields for all of the variables in the class. In addition, the table adds records automatically when objects are created. ORM simplifies the construction of tables using classes and adds data using class objects. The Django ORM's ability to extract information speeds up the web application development process. It also enables developers to quickly create functioning prototypes. The ORM in Django allows developers to switch between relational databases with little code modifications. Django allows you to migrate from one database to another and conduct typical operations without having to write a lot of extra code.

The platform needs to manage some computation-intensive tasks, such as running ML training and optimization works. As a result, background tasks are typically executed as asynchronous processes outside the request/response thread. Celery [69] is a popular Python task queue library with a focus on real time processing. Task queues are used to distribute work across threads or machines. The input to a task queue is a unit of work known as a task. Task queues are regularly monitored by dedicated worker processes for new work to accomplish. Celery interacts by messages, with a broker commonly acting as a mediator for clients and workers. To initiate a task, the client adds a message to the queue, which is subsequently sent to a worker via the broker. Celery systems can include numerous workers and brokers, allowing for high availability and horizontal scaling.

Celery utilizes the producer consumer design pattern where a producer creates the task. The task is then placed in a messaging queue. Finally, consumers subscribe to the messaging queue can receive the messages and process the tasks in a different queue. Celery is both a producer and a consumer in the manner that it serves as a producer when an asynchronous task is called in the request/response thread, adding a message to the queue, as well as listening to the message queue and processing the message on a separate thread.

For Celery to work effectively, a broker is required for message transport. In this work, Redis [131] is used as storage of tasks until consumed and to persist results of the task. Figure 3.4 shows a brief illustration of how celery and redis work in the application.

Redis is an in-memory data structure store. It is a disk-persistent key-value database with support for multiple data structure or data types. This implies that, in addition to mapping key-value based strings for storing and retrieving data (similar to the data model offered by traditional types of databases), Redis supports additional complex data structure as lists, sets, and so on. Redis works by mapping keys to values with a sort of predefined data model. All Redis data is stored in memory, allowing for low latency and high throughput data access. In-memory data storage, unlike traditional databases, do not require a trip to disk, decreasing engine latency to microseconds. As a result, in-memory data storage can handle others of magnitude more operations and respond faster. As a consequence, read and write operations take less than a millisecond on average, and millions of operations per second are supported [131].

In the architectural design of Redis, a primary-replica configuration is employed, utilizing

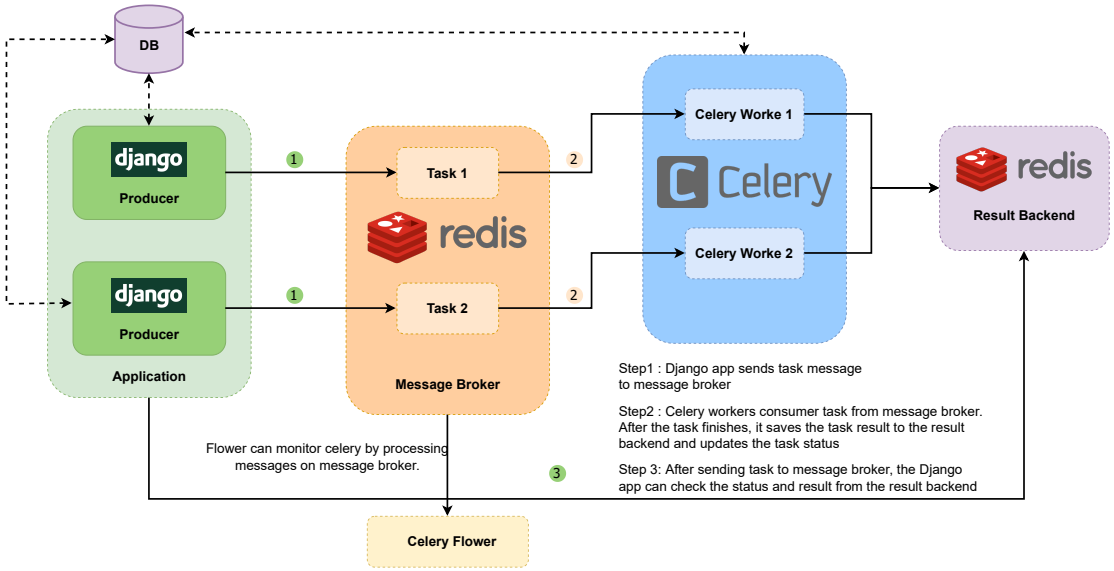

Figure 3.4: The application acts as producers and workers as consumers waiting for tasks to be put in a queue so they can be consumed. Redis acts as message broker and storage caching. Flower is a web based tool for monitoring Celery workers and task progress.

asynchronous replication to facilitate the distribution of data across multiple replica servers. This architectural choice enhances the system's read performance by allowing the dispersion of queries among the various servers, and it augments the recovery speed following a primary server failure. The persistence of data is ensured by the feature of point-in-time backups in Redis, which involves copying the Redis data set to disk. In the context of a high availability setup, the fault tolerance provided by Redis contributes to robust system performance, and it offers rapid data accessibility, which is particularly advantageous for systems experiencing high traffic loads. A specific application of Redis is its role in storing messages produced by the application code, which delineate the tasks awaiting execution in the Celery task queue. Moreover, Redis serves as a storage medium for the results of the Celery queue, which are subsequently accessed by the queue's users.

To monitor and manage the Celery workers and task process, Flower [43] is used as the tool to administrate the celery clusters. The Flower dashboard lists all Celery workers connected to the message broker. Flower is able to monitor Celery events in real-time, it shows the task process and history. Celery can control workers remotely - restart, revoke, or terminate worker instances. Besides, it has ability to show task details (arguments, start time, runtime, and more) with graphs and statistics.

With this setting, the software can run different computation-intensive tasks asynchronously. Celery makes scheduling periodic tasks easy. If the application's workflow includes a lengthy procedure, Celery can be used to run that process in the background as resources become available, allowing the application to keep up with client requests. This keeps the task independent of the context of the application. Figure 3.5 illustrates a screenshot of the Flower dashboard.

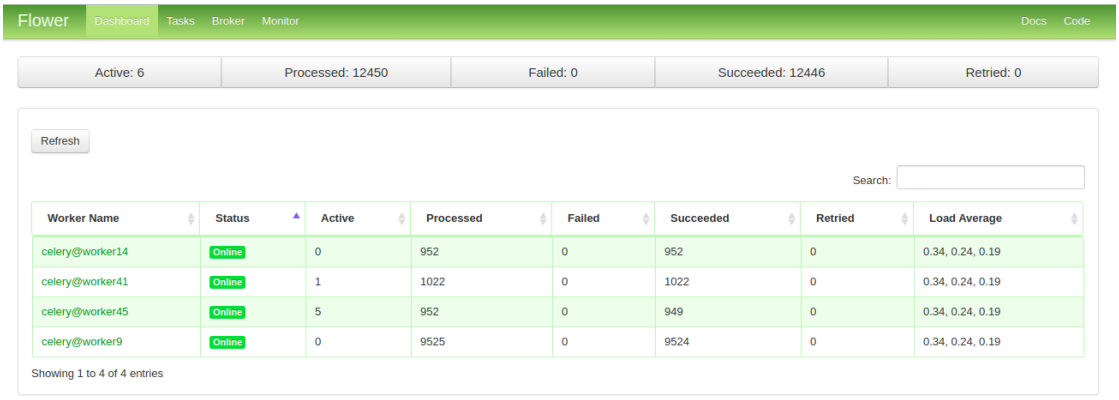

Figure 3.5: Flower can monitor all tasks and their status.

3.3 Implementation details

The architecture of the software provides a robust and flexible foundation of implementation possibilities. The extension and new features can be added easily with this architecture. The platform simulates the process of an end-to-end foundry system based on the combination of ML and numerical optimization tools. The simulation systems contains three main components, the first one is a dashboard that presents the available scrap information. The scrap dealers can upload their scrap information, like quantity, price, chemical analysis results for specific scrap. The second component is an AutoML (cf. section 2.4.4) system that allows user to train, evaluate, and deploy ML models based on predefined metrics and target variables. The last component is an optimization tool that integrates the data-driven methods and numerical optimization, which is described in chapter 6.

3.3.1 AutoML system

The system was built to accepted ML models from many sources, which means that the system can take trained ML models either from the software EidoData Desktop (cf. section 3.1) and EidoData Web (cf. section 3.2) or user can train a ML model online. The AutoML system of the software is implemented based on the open source Python library AutoGluon [35].

AutoGluon automates the time-consuming tasks for ML, such as data preparation, feature engineering, validation splitting, missing value handling, model selection (cf. section 2.4.1.3) and hyperparameter tuning (cf. section 2.4.3). AutoGluon automatically recognizes the data type in each column for robust data preprocessing, including particular handling of text fields. It can fit a variety of models, from pre-built boosted trees (cf. section 2.4.2.1) to unique neural network models (cf. section 2.4.2.2). These models are ensembled in an approach that ensures raw data can be converted into high-quality predictions within a certain amount of time, the models are stacked in multiple layers and trained in a layer-wise fashion [35]. Specifically, AutoGluon simplifies reuses all of its base layer model types as stackers, which with the identical hyperparameter values. Unlike other approaches, the stacker of AutoGluon use the original data attributes together with the predictions from the models at the preceding layer. Thus, during training, the higher-layer stackers can check the initial data values. Through the use of all available data for training and validation, as well as k-fold (cf. section 2.4.1.3) ensemble bagging of all models at all layers of the stack, AutoGluon can further enhances its stacking performance. Each

model is asked to make out-of-fold predictions on the portion it did not view during training after AutoGluon bags all the models. As every training example is out-of-fold for one of the bagged model copies, this allows us to obtain out-of-fold predictions from every model for every training example. Single individual models may have poorer accuracy but are computationally advantageous, whereas training ensemble models typically results in computationally demanding models but better results. AutoGluon also allows model distillation, which offers one way to retain the computational benefits of a single model, while keeping some of the accuracy-boost that comes with ensembling [38].

The AutoML provides many functions in terms of a production-ready ML system. An Entity Relationship Diagram (ERD) of the AutoML system is illustrated in Figure 3.6. Every ML training project is an *Experiment* in the AutoML system. Experiment defines the target variable of a ML task, it can be a classification problem or a regression problem (cf. section 2.4.1). An experiment also provides evaluation metrics (cf. section 2.4.1.4) as objective function for AutoGluon to optimize. Furthermore, experiment allows user to select best fit algorithms for different scenarios. For instance, if algorithms is specified as best quality, which allows AutoGluon to automatically construct powerful model ensembles based on stacking or bagging, and will greatly improve the resulting predictions if granted sufficient training time. On the other hand, medium quality, which produces less accurate models but facilitates faster prototyping. If the training and inference speed are more important than predictive performance, user can also choose the algorithm optimize for deployment. In the experiment, a *File* object controls operations related to files, such as uploading and downloading. *FileMetaData* object stores information of the data type in each column that recognized by AutoGluon. Those information can be further used for data visualization, feature engineering, model prediction and model evaluation. AutoGluon saves the best predictive ML algorithm as an experiment Model. The model object contains the validation score and testing score of the selected algorithm, and the feature importance of all columns are stored in the model object as well.

Every ML related job can have multiple status during execution. The *Task* object is an instance of Celery (cf. section 3.2) state. A task performs dual functions by defining what occurs when a task is called (sends a message) and what happens when a worker receives that message. Every task has a unique name, which is used in messages to help the worker identify the appropriate function to execute. Until a worker has acknowledged a task message, it will not be removed from the queue. A worker can reserve many messages in advance and even if the worker is killed – by power failure or some other reason – the message will be redelivered to another worker. *Task* functions should ideally be idempotent, which means they will not have unintended consequences even if called repeatedly with the identical arguments. The worker's default action is to acknowledge the message in advance, just before it is executed, ensuring that a task invocation that has already begun is never completed again. This is because the worker cannot tell if the tasks are idempotent.

Celery can keep track of the tasks current state. The state also contains the result of a successful task, or the exception and the traceback information of a failed task. During its lifetime a task will transition through several possible states, and each state may have arbitrary meta-data attached to it. When a task moves into a new state the previous state is forgotten about, but some transitions can be deduced. There are six states that indicate the Celery task:

- Pending: waiting for execution or unknown task id;
- Started: a task has been started;
- Success: a task was executed successfully;
- Failure: a task execution resulted in exception;

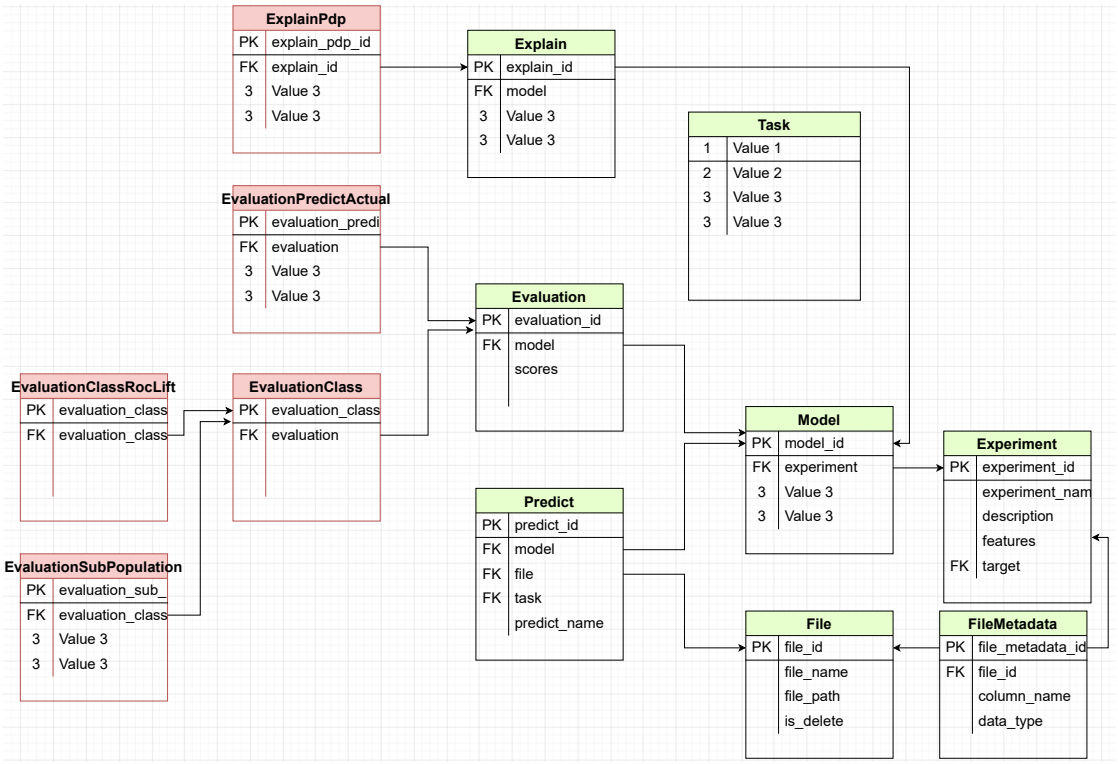

Figure 3.6: ERD for the AutoML system.

- Retry: a task is being retried;
- Revoked: a task has been revoked.

The states information of every task will then be reported in Flower (cf. section 3.2). All ML tasks such as file uploading, predictions, evaluations, are run in the background asynchronously, these processes are passed off to a task queue and let a separate worker process deal with them. The AutoML system works in conjunction with Celery to handle long-running processes outside the normal request/response cycle. If a task is began, the information will be sent to server-side via POST request. Within the route handler, a task is added to the queue and the task id is sent back to the client-side. Using Django, the client continues to poll the server to check the status of the task while the bask itself is running in the background. Figure 3.7 shows the whole workflow as described above.

3.3.2 Software deployment

The schematic design of the software, as outlined in section (3.2), promotes effortless dockerization (cf. section 2.6.1) of the Django application in conjunction with Redis, Celery, PostgreSQL, and Flower (cf. section 3.3), thereby facilitating the management of asynchronous tasks. Primarily, the conceptual model underlines the integration of Django with docker containers, with particular emphasis on Redis and Celery. Ultimately, each docker container is earmarked for deployment to Kubernetes, as discussed in (cf. section 2.6.2).

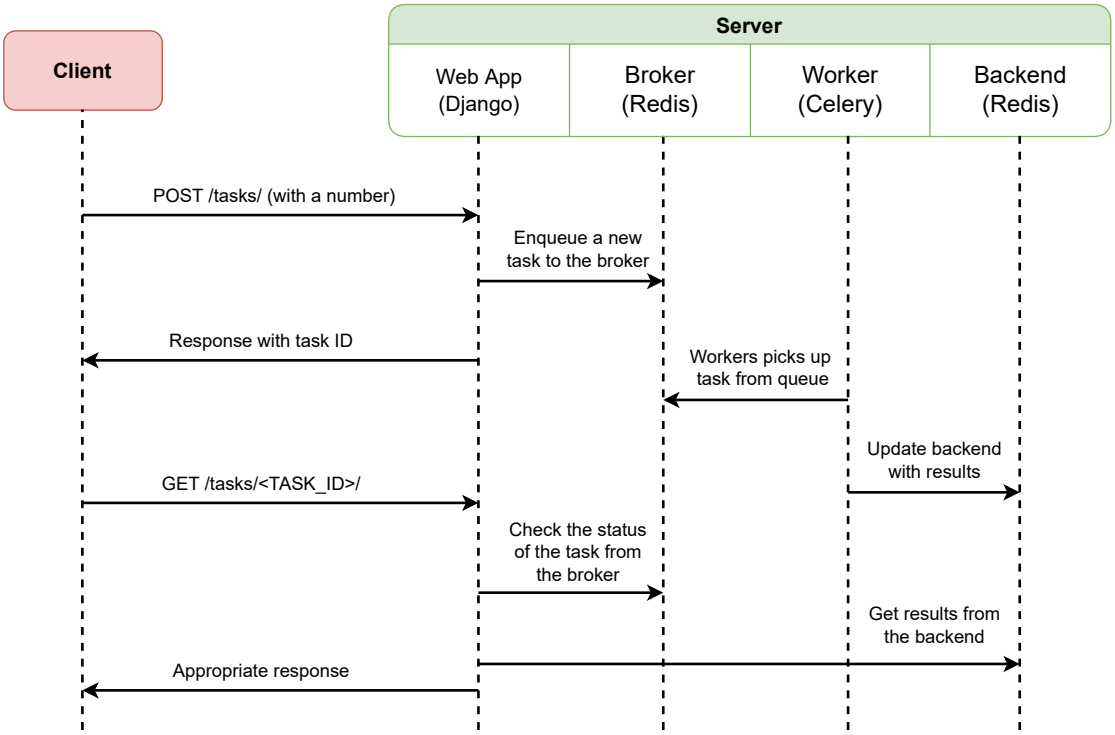

Figure 3.7: The workflow of the asynchronous tasks in AutoML with Celery.

The Kubernetes secrete resource API can be utilized to handle credentials used by the PostgresSQL and Django pods. The PostgresSQL deployment is a database engine that would allow storing the information of our software such as the ML tasks. The actual data coming from the PostgresSQL application should be stored separately in a volume. This volume would need be persistent so that if one or all of the pods running the PostgresSQL containers stops for any reason the data is not lost. The PostgresSQL engine would run within a container in the 'postgres' pod. To expose the 'postgres' pod to other objects within the node, such as Django pods in this case, an internal service called 'ClusterIP' need to be created. The Django deployment would create any defined number of Django pods, which together would constitute the backend application that provides the ML and optimization functions to interact with our inputs from the frontend. Just like the PostgresSQL application, the Django application has a ClusterIP service. The ClusterIP services enables the Django backend API to interact with the frontend via the browser. In a similar way, the Redis, Celery, and Flower are also dockerized as Kubernetes pods.

Each worker is booted with the docker container of the algorithmic repository. Once the container is up and running, it connects to Celery and Redis. It then takes a task from the queue and calculates it. The output is written by the worker to PostgresSQL. If the task was completed successfully then the return value of the Celery task, is a JSON containing a URL to the output saved to Flower and its metadata. That return value is automatically saved to Redis by Celery and also save to PostgresSQL. If the task failed to finish, an exception is saved to redis. The workflow of deployment is demonstrated in Figure 3.7. In the end, the software can be deployed to any cloud provider, such as Google Cloud Platform, Amazon Web Services or Microsoft Azure.

3.4 Summary

In order to provide a unified framework for the exchange between foundries and scrap suppliers, a web platform was developed which integrates the supply data with ML-based models of the melting process for individual foundries in order to determine the actual costs (including energy costs) of batches for any selected scrap composition. For producing any desired steel type, the platform then allows for an automatic selection of the scrap composition which is optimal with respect to the cost, using methods described in section 5.1. The architecture of this platform is based on state-of-the-art design patterns and software libraries, with a focus on flexibility and resilience. The feasibility of the developed solution will be demonstrated via time discrete simulations in chapter 6.

Chapter 4

Scrap procurement optimization

The optimization of scrap materials for the foundry industry is an intricate process, necessitating the consideration of numerous factors that influence the melting of scrap. This complexity arises from the multifaceted nature of the industry and the diverse variables that come into play.

Primarily, an effective optimization program must incorporate the fluctuating market prices of scrap materials. These prices are subject to constant change due to various economic factors, including supply and demand dynamics, global economic conditions, and industry-specific trends. Therefore, a robust optimization program should have the capability to adapt to these price fluctuations and make purchasing decisions that are economically advantageous.

Secondly, the program should take into account the metallurgical and operational constraints inherent in steelmaking. This includes considerations such as the chemical composition of the scrap, its physical properties, and the operational parameters of the steelmaking process. These factors can significantly influence the quality of the final product and the efficiency of the steelmaking process.

Finally, the exact scrap composition further influences the foundry process beyond the pure chemical composition of the resulting steel. Achieving an optimal mix is essential not only for ensuring the quality of the steel but also for reducing the energy consumption and thereby managing production costs: The correct scrap mix can lead to a more efficient melting process, thus lowering energy requirements and contributing to overall cost savings.

The ultimate objective of such a sophisticated and proprietary purchasing and melt-chemistry optimization program is to determine the optimal scrap mix for every heat melt order. This involves a delicate balance between economic considerations, operational constraints, and quality requirements. The successful implementation of such a program can result in significant cost savings, improved operational efficiency, and enhanced product quality.

In real-world scenarios, the implementation of such an optimization program can be challenging due to the inherent variability in scrap materials and the complex nature of the steelmaking process. ML can be used to develop sophisticated optimization algorithms that can determine the optimal mix of scrap for each melt order. These algorithms can consider a wide range of factors, including the chemical composition of the scrap, operational constraints, and the desired properties of the final product.

In the following section, the traditional methodologies utilized for scrap optimization are systematically outlined. These traditional approaches, while having served the industry for a considerable period, have certain limitations when faced with the intricate realities of scrap optimization in the contemporary foundry industry.

Acknowledging the potential of ML as a pivotal instrument for scrap optimization, the ob-

jective is to proliferate its adoption among foundry industry stakeholders, especially those with limited domain-specific knowledge. Consequently, two intuitive ML software applications have been developed for this purpose. These tools have been designed with an intuitive interface and robust capabilities, enabling users to leverage the power of ML for scrap optimization without the need for in-depth technical knowledge.

These software applications represent the dedication to integrating advanced technology with practical implementation. This facilitates the foundry industry in optimizing scrap utilization, improving operational efficiency, and promoting sustainable growth.

4.1 Traditional methods for scrap purchase optimization

Purchased steel scrap serves as the principal input material for an EAF, playing a substantial role in influencing production expenditures. The utilization of steel scrap, in varying proportions, is instrumental in attaining the requisite physical and chemical attributes of the end product, thereby fulfilling customer specifications. In the absence of considerations for energy pricing and other operational constraints, the Simplex algorithm [75] emerges as a potentially viable method for scrap purchase optimization. The Simplex algorithm, a well-established and widely utilized method in the realm of linear programming, offers a straightforward approach to problem-solving.

In the context of scrap optimization, the primary objective is to minimize the cost of procured scrap materials while adhering to the chemical composition requirements of the designated steel materials and their final weight. This objective is inherently a linear programming problem, where the cost of different types of scrap materials is minimized subject to the constraints of the chemical composition requirements. The Simplex algorithm operates by iteratively moving along the edges of the feasible region (defined by the constraints) towards the optimal solution. In the case of scrap optimization, each point in the feasible region represents a possible mix of scrap materials, and the algorithm seeks to find the mix that minimizes the total cost.

However, in real-world scenarios, the optimization of scrap materials is often more complex and involves additional considerations. For instance, the energy cost associated with melting different types of scrap materials can significantly impact the total cost. Similarly, operational constraints such as the capacity of the furnace, the melting time, and the availability of different types of scrap materials can also influence the optimal solution.

Moreover, the prices of scrap materials are often subject to fluctuations due to market dynamics, which adds another layer of complexity to the problem. Therefore, a more sophisticated approach may be required to effectively optimize scrap material purchases in real-world scenarios. Despite these complexities, the Simplex algorithm provides a valuable foundation for understanding the basic principles of scrap optimization and serves as a useful starting point for developing more advanced optimization strategies.

A more complex model, employing mixed-integer nonlinear optimization techniques, has been developed to assist in the procurement of scrap steel [97]. This advanced model is a demonstration of the intersection of mathematical optimization and industrial operations, specifically in the context of the steel industry. Mixed-integer nonlinear optimization is a sophisticated mathematical method that allows for the inclusion of both discrete and continuous variables, as well as nonlinear relationships, in the optimization problem. This makes it particularly suitable for modeling complex industrial processes where certain decisions are binary (e.g., whether to purchase a particular type of scrap or not), some variables are continuous (e.g., the amount of a particular type of scrap to purchase), and the relationships between variables are not necessarily linear.

The procurement expenses and energy consumption costs can be correlated to the quantities of

utilized scrap. This expression, constitutes the economic objective function for the scrap blending optimization problem, serving as an approximation of the total operational expenditure, denoted as c.

The procurement cost, represented as c_p in Equation ((4.1.1)), encompasses the expenses incurred for the inclusion of c_{pij} [kg] of each scrap type i into an EAF batch j, in order to fulfill production demand. Beyond the quantity of scrap incorporated into each produced batch, the cost c_p is associated with the volume of scrap procured from a specific supplier, the amount extracted from inventory, and the integer decision variables employed to select the source and the pricing option for the scrap.

The computation of energy expenditure within the context of steel production involves a mathematical operation that multiplies the quantity of energy consumed, denoted as y [kWh] , by the prevailing market price for electricity, represented as c_e [$/kWh]. This calculation is executed individually for each of the J batches, which are distinct production cycles within the foundry operation. Each unique batch of steel necessitates its own specific inclusion of each of the I types of scrap. The total number of batches, J, is manually determined in advance of the optimization process. Simultaneously, the number I of accessible commodity types is also established for a predetermined time frame that corresponds to a specific scrap purchase, typically spanning a monthly period. The primary objective of the optimization process is to minimize cost. This is accomplished by determining the optimal $I \times J$ scrap additions for the designated purchase period, as well as the quantities to be procured from suppliers. This process involves a complex interplay of variables and constraints, necessitating the application of advanced mathematical techniques and a comprehensive understanding of the operational parameters and market dynamics of the steel industry.

In a broader context, this mathematical representation of the scrap blending optimization problem provides a framework for understanding and managing the complex trade-offs involved in scrap procurement and usage in the steelmaking process. By quantifying the costs associated with different aspects of the process, it enables decision-makers to make informed choices that balance economic considerations with operational constraints and quality requirements. This, in turn, can lead to improved operational efficiency, cost savings, and enhanced product quality in the steelmaking industry.

$$c = \sum_{i=1}^{I} \sum_{j=1}^{J} (c_{p\,ij}) + \sum_{j=1}^{J} (c_e \cdot y_j)\,. \tag{4.1.1}$$

The electricity consumption y, for each batch ($j = 1, \ldots, J$) is modeled through a regression equation based on Partial Least Squares methodology [138]:

$$y_j = \beta_0 + \beta_1 m_{1j} + \ldots + \beta_I m_{Ij} + \beta_{I+1} q_{1j} + \ldots \beta_{I+Q} q_{Qj}\,. \tag{4.1.2}$$

This approach facilitates the formulation of the relationship between the I scrap commodities and operational variables q. These operational variables encompass the additions of chemical reagents and temperature settings, which are integral to the steelmaking process. The operational variables q are established as constants for each optimization iteration, grounded on the understanding of each specific batch type of steel. This knowledge is derived from the inherent characteristics of the steel batch, including its chemical composition, physical properties, and the specific requirements of the steelmaking process. Models such as the one represented in Equation (4.1.2) can be fitted utilizing historical data sourced from production databases.

However, the linear regression model presented in Equation ((4.1.2)) may not adequately capture the detailed relationships between various types of scrap and specific furnace characteristics

during operations. To address this, a more comprehensive data-driven approach should be considered. This approach would utilize a wide range of collected data, encompassing elements like the chemical composition of input materials, the duration of the melting process, and the energy requirements for different melting operations. Such a method would allow for a more accurate representation of the energy consumption associated with steel casting in a particular furnace, taking into account multiple influencing factors. This approach is grounded in the analysis of extensive data sets, aiming to develop a more precise model that reflects the complexities of the steelmaking process.

The data-driven approach is instrumental in constructing a robust predictive relationship between process data and energy consumption. This relationship is not merely linear or simplistic; it is complex, reflecting the multifaceted nature of the steelmaking process. The predictive model, therefore, needs to account for this complexity and variability, making it capable of accurately predicting energy consumption under a wide range of operating conditions and input parameters. In a real-world scenario, the implementation of such a data-driven model would require a robust data infrastructure capable of handling large volumes of data, sophisticated data processing and analysis tools, and a deep understanding of the steelmaking process and ML workflow. In order to democratize the benefits of ML and make it accessible to non-experts, two user-friendly machine learning software tools have been developed. These tools have been designed with a focus on usability and simplicity, enabling individuals without a deep technical background in ML to leverage its capabilities effectively.

4.2 EidoData desktop

One of the most common worries of a ML beginner is understanding how to code. EidoData Desktop software attempts to bridge the gap between expertise ML skills and non-professional ML users by offering an simple-to-use GUI and complete functionality for almost all aspects of a ML project. EidoData Desktop automates ML tasks to enable people make fully informed decisions based on massive amounts of data. The software displays all the details of a model's configuration so it is easy to understand how it generates predictions. Analysts who may not be familiar with modeling or ML processes can quickly uncover insights to help solve complicated problems. Users can fine tune model's hyperparameters and develop highly sophisticated models using an interface with no coding required. The software's interactive GUI also allows users to save enormous amounts of time developing complex models so they can begin delivering useful predictions quickly.

The challenges in implementing the software EidoData Desktop is the fact that Python, as the dominant ML programming language, has a versatile ecosystem consisting of millions of open source libraries, but Python is not a suitable language for developing state-of-the-art Windows GUI software. Meanwhile, languages such as Java, C++ or *C#* are good options for GUI software development, but do not have direct access to powerful ML open source libraries as Python.

4.2.1 Implementation details

To make use of Python's ML open source services and provide a state-of-the-art interface to user, a separate software architecture was designed. More precisely, Python only takes responsibility for all ML tasks; it plays the role of the backend worker for the software. On the other hand, *C#* was chosen as the frontend programming language for developing the GUI, which only receives the inputs and commands from user and sends them to backend. Finally, the frontend displays the ML task's results to the user.

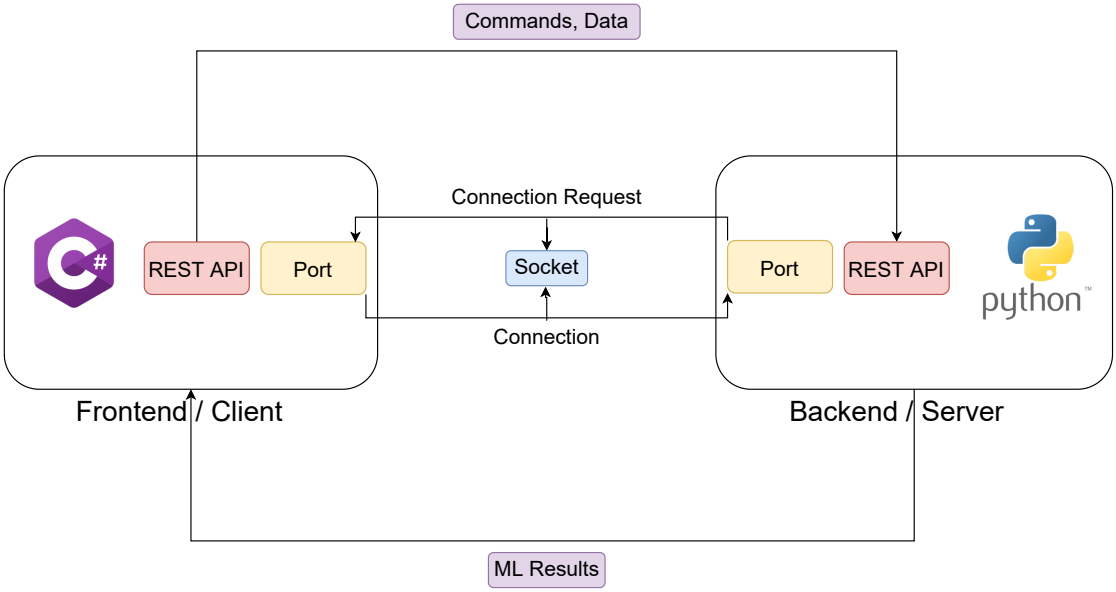

Figure 4.1: The *C#* module acts as client, it sends the commands and data to the server-side. The server-side processes the request and send back the decoded ML results to the client.

Figure 4.1 shows the basic architecture of the software EidoData. Since *C#* and Python are two different programming languages, a communication mechanism via local sockets was used to establish the connection between them. The socket API are used to send message across networks. It provides a form of inter-process communication (IPC). The network can be a logical, local network to the computer, or one that is physically connected to an external network, with its own connections to other networks. In the case of EidoData, the network is the local computer, which also implements the most common type of socket applications, namely the client-server application. The default protocol is used is the Transmission Control Protocol (TCP). TCP is reliable since it ensures that packets dropped in the network are detected and retransmitted by the sender. TCP also enables in-order data delivery; data is read by the application in the order it was written by the sender [7]. With TCP, there is no need to worry about packet loss, data not arriving properly, and other problems that inevitably occur when communicating over a network via other protocols such as UDP.

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request. On the client-side, the client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system. If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound the the same local port and also has its remote endpoint set to the address and port of the client. An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. The new socket is required so that the host can continue to listen to the original socket for connection requests when tending to the needs of the connected client. On the client side, if the connection is accepted, a socket is successfully created and the client

can use the socket to communicate with the server.

Once the connection between *C#* and Python is established, a data transfer mechanism is needed so that they can exchange data. APIs are commonly used to retrieve data from other resources. The API acts as a layer between client and server. The client send a certain command and data and receive data in a predetermined format from server. A REST (Representation State Transfer) API is a software architecture style that defines a pattern for client and server communications over a network. This structure enables a web service to expose operations and data through a series of well-defined endpoints. The requests that a client program sends to a REST service to retrieve, modify or create, or delete data use a predefined set of actions. A REST service responds to these requests in a standardized manner. This approach makes it easier to construct client programs. REST provides a set of constraints for software architecture to promote performance, scalability, simplicity, and reliability in the system. It handles HTTP requests and routes them to the appropriate function in the application. To convert the state of data or commands from the client into a form that can be persisted or transported, all the inputs from client will be serialized in the JSON (JavaScript Object Notation) format and sent to server-side. Once the ML tasks finished successfully, the client-side will convert the serialized stream of data into the original object state. This process is called deserialization, ensures that the original state is not altered and is recreated when needed.

Although the *C#* front-end only plays the role of a graphical interface, it still needs a robust design principle to be scalable, modular and maintainable. Different architectures may vary in their details, but they all have the same objective, the separation of concern. This separation is achieved by dividing the application into layers. The main concept of clean architecture is that application code or logic which is very unlikely to change, has to be written without any direct dependencies [45]. In clean architecture, the Domain and Application layers remain at the center of the design which is known as the Core of the system. The domain layer contains actual enterprise logic and the application layer contains business logic. Enterprise logic can be shared across many systems, but the business logic will typically only be used within the system. The core will be independent of data access and other infrastructure concerns.

Business logic and the application model are at the heart of clean architecture. Instead of business logic being dependent on data access or other infrastructure issues, this dependency is inverted, with the application core being dependent on implementation and infrastructure details. This functionality is achieved by defining abstractions, or interfaces, in the application core, which are then implemented by types defined in the infrastructure layer. Figure 4.2 shows a more traditional horizontal layer diagram that better reflects the dependency between the user interface and other layers.

4.2.2 Features of EidoData desktop

The EidoData Desktop software adheres to a design principle that streamlines the ML process. This section outlines its primary functions and features, which are categorized into four fundamental components: data exploration, data preprocessing, model training, and evaluation. The initial stage in any ML task facilitated by this software is data importation. To this end, the software is equipped with fundamental import functions that enable loading data from various sources. These sources include Excel or CSV files, text files, clipboard content, and data from database systems like SQLite or MySQL. This functionality ensures versatility in handling different data formats and origins, a crucial aspect in the initial stages of the ML workflow.

Once the data is imported, EidoData can immediately analyze the data. For each column in the dataset, the corresponding statistics are displayed:

- type inference: recognize column types in a data frame.

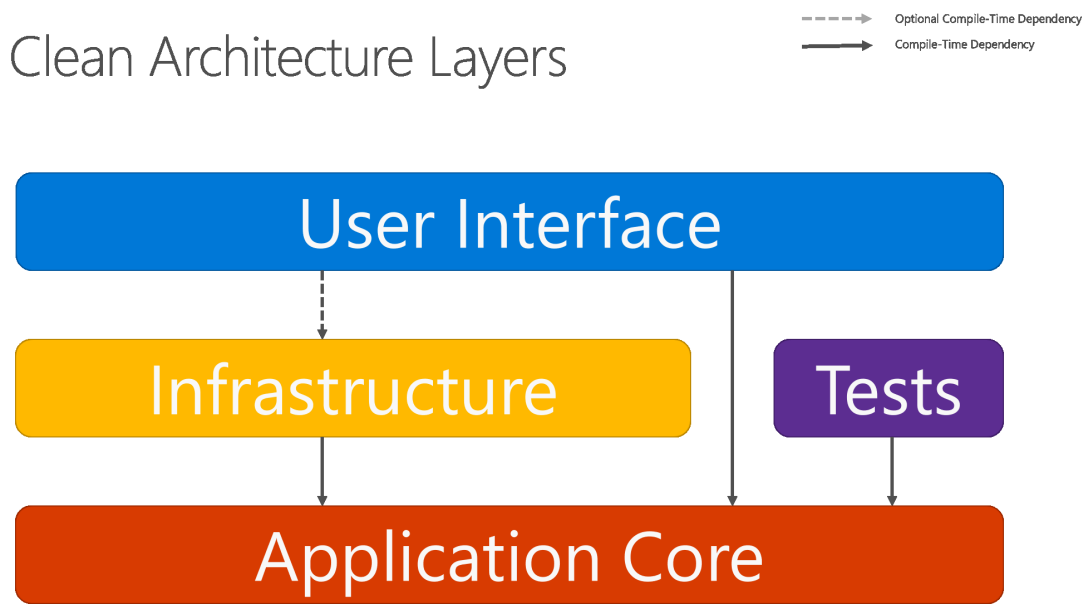

Figure 4.2: Clean architecture: the user interface layer uses interfaces declared at compile time in the application core and, in theory, should not be aware of the implementation types provided in the infrastructure layer. However, these implementation types must be present and connected to the Application Core interfaces via dependency injection at runtime in order for the application to execute. [95]

- key features: type, unique values, missing values.
- quantile statistics: minimum value, Q1, median, Q3, maximum, range and interquartile range.
- descriptive statistics: mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, skewness, process capability index.
- highlighting correlations of highly correlated variables, Spearman, Person and Kendall matrices.

To utilize ML effectively, data preprocessing is a critical step, as detailed in section 2.4.1.2. This preprocessing involves a series of operations, each designed to refine the raw data, thereby enhancing the ML algorithms' ability to construct more accurate predictive models. For structured data, EidoData incorporates a range of processing operations, which may include:

- data cleaning: removing or correcting records with broken or invalid values (outliers) from raw data, as well as removing records that are missing a large number of columns.
- partitioning: select data points from the input dataset to create training, validation and testing dataset (cf. section 2.4.1.2).
- data transformation: transforming categorical features into numerical representation.
- feature extraction: reducing the number of features by creating data representations with fewer dimensions and higher performance. Methods such as PCA, SVD, KPCA are used for this purpose (cf. section 2.4.1.2).

- feature scaling: improve the quality of a feature of ML, including scaling and normalization of numerical values, input of missing values, adjustment of values with asymmetric distribution (cf. section 2.4.1.2).

When the user is satisfied with the preprocessing result, training a ML model in EidoData can become very straightforward. EidoData supports both classification and regression problems for supervised learning. For a regression problem, EidoData provides 26 algorithms with extensive parameter settings whereas 23 algorithms for classification problem are also implemented. There is also an ensemble method called "supervisor" for both regression and classification tasks. The training result is displayed with many different metrics depending on the task. Figure 4.3 shows the result of a random forest and its prediction on the test dataset. Users are then able to adjust the hyperparameters and select the machine learning model that gives the best result.

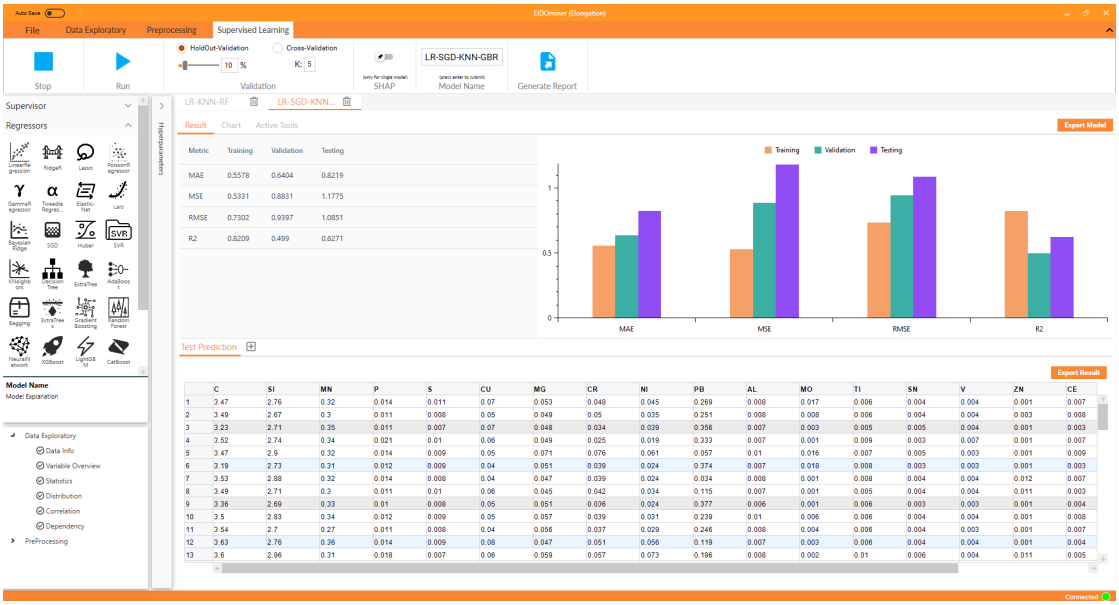

Figure 4.3: Training a regression problem with random forest in EidoData.

To evaluate the training result, EidoData also provides various plots for visualization. For the regression problem, there are "Residual Plot" and "Forecast Error Plot". For classification problems, EidoData supports the "Confusion Matrix Heatmap". With the help of these diagrams, the result becomes more understandable.

The ability to understand how models generate their output is critical to building confidence in the reliability and value of ML investments. With EidoData, it is easy for model designers, analysts, and decision-makers to understand exactly how models work. The software's highly visual approach to model construction maintains high levels of ML performance and prediction accuracy while allowing people to understand, trust, and manage their ML models. EidoData provides two essential tools for model interpretation: feature permutation and SHAP values [86]. EidoData supports a transparent model-building process. Users, including people who are not data scientists, can see how models gather and process data to generate predictions.

For each ML algorithm, EidoData provides an intelligent way for hyperparameter tuning (cf. section 2.4.3). EidoData predefines some important hyperparameters for all ML models. To find the best training result, a combination of hyperparameters can be determined easily. Finally,

EidoData can generate a report about the entire ML process, which contains information about the original dataset, the methods used for preprocessing, the selected ML algorithms for training and the corresponding results.

4.3 EidoData web

The limitation of EidoData Desktop is that the user has to install the software on a local Windows machine. For some users, this could be problematic and inconvenient. To make the ML services even more accessibly and reduce the cost of the learning curve, a web service could be an alternative. The authorized user is then able to use the ML services without installing any software. The rigorous tuning of hyperparameters offered by EidoData Desktop also call for the user to have some ML expertise. EidoData Web provides a user-friendly interface without addressing ML hyperparameters to make ML services more easily accessible for all users.

The benefits of using the web version of EidoData can be concluded as follows. First of all, it enables experimenting and testing multiple models. The advantage of the web lies in its capacity to facilitate the scalability of ML projects. Initiating with a limited dataset, there exists the potential to incrementally incorporate more data points as confidence in predictions increases. The web interface permits experimentation with ML functionalities, allowing for expansion as projects transition to production and as demand intensifies due to fluctuating usage. Furthermore, ML can be employed to assess multiple datasets to determine optimal performance. Conversely, the establishment, operation, and maintenance of robust servers present significant challenges. Leveraging cloud solutions mitigates much of the intricacy associated with these tasks, as the cloud service provider assumes responsibility for a significant portion of these complexities.

4.3.1 Implementation details

To automate the EidoData Web development workflows and deploy better quality code, more often, CI/CD (continuous integration and continuous delivery) is a very important tool. CI/CD automates much or all of the manual human intervention traditionally needed to get new code from a commit into production such as build, test, and deploy, as well as infrastructure provisioning. Utilizing a CI/CD pipeline facilitates the automation of code modifications, ensuring they undergo rigorous testing before proceeding to delivery and deployment stages.

Continuous integration refers to the systematic practice of validating each modification to the source code by automatically initiating tests upon commit or merge. This process facilitates the prompt and consistent amalgamation of code alterations into the primary branch of a shared source code repository. Continuous integration makes it easier to find and fix bugs and security problems, and it does so considerably earlier in the software development lifecycle. Even when numerous developers are working on the same application, code conflicts may be minimized by often merging changes and initiating automated testing and validation procedures. The expedited response time and the capability to address bugs and security issues represent additional advantages. A static code analysis that confirms the code's quality is the first step in most code validation methods. Once the static tests are passed, automated CI processes package and compile the code in preparation for additional automated testing. A version control system that monitors change should be utilized in CI procedures.

Continuous delivery (CD) automates the release of verified code to a repository after the automation of builds and unit and integration testing in CI. Therefore, it is critical that CI is already built into the development pipeline in order to have a successful continuous delivery process. A code base that is constantly prepared for deployment to a production environment is the aim of continuous delivery. In continuous delivery, every stage—from the merger of code

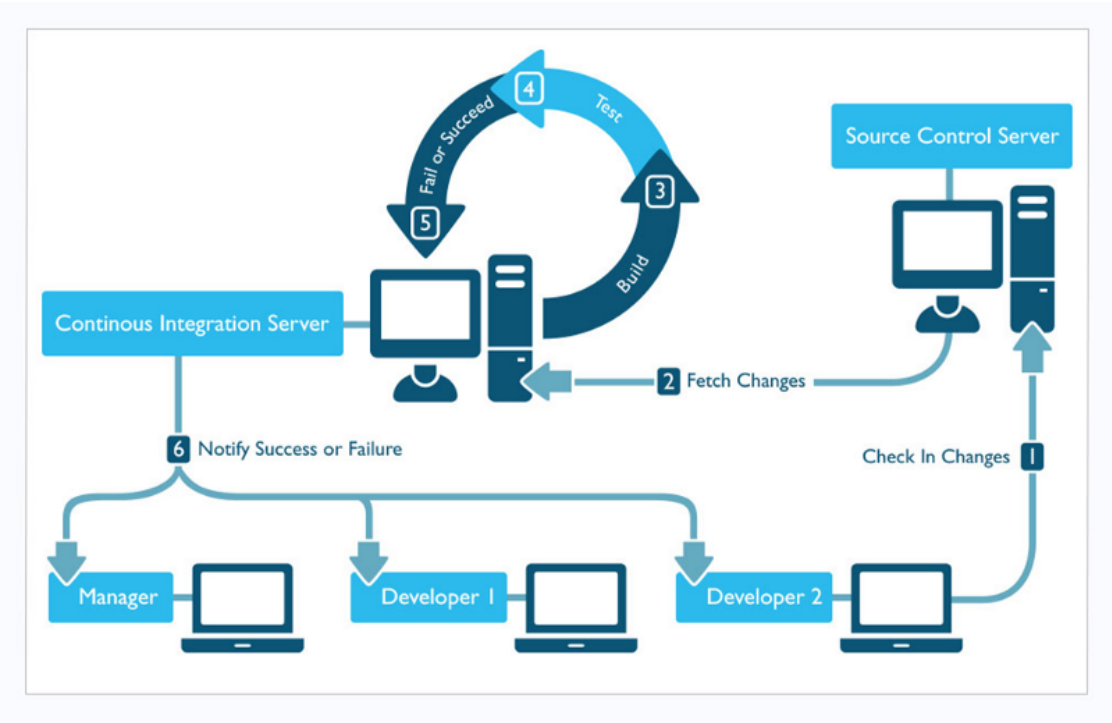

Figure 4.4: Continuous Integration (CI) is a phase in the software development cycle where code from different team members or different features are integrated together. This usually involves merging code (integration), building the application and carrying out basic tests all within an ephemeral environment [130]

changes to the delivery of production-ready builds—involves test automation and code release automation. At the end of that process, the operations team is able to deploy an app to production quickly and easily. With continuous delivery, the software is built so that it can be deployed to production at any time. The deployments can then be manually triggered or automated as well.

The final stage of a mature CI/CD pipeline is continuous deployment. Continuous deployment automates the release of an application to production as an extension of CD, which automates the release of a production-ready build to a code repository. Continuous deployment heavily relies on well-designed test automation as there is no manual gate at the pipeline level prior to production. In practice, continuous deployment means that a developer's change to a cloud application could go live within minutes of writing it, assuming it passes automated testing. It has thereby become much simpler to regularly receive and take into account customer input. When all of these CI/CD techniques are used, deployment of an application becomes less hazardous, making it easier to release application modifications incrementally rather than all at once. There is also a lot of upfront investment, however, since automated tests will need to be written to accommodate a variety of testing and release stages in the CI/CD pipeline.

Adopting CI/CD into software development tends to facilitate a number of positive changes. First, the users and customers will have a better experience since fewer bugs and problems occur in production. This leads to improved levels of customer satisfaction, confidence and reputation. Second, CI/CD can accelerate time-to-value. The deployment of software has not limitation of

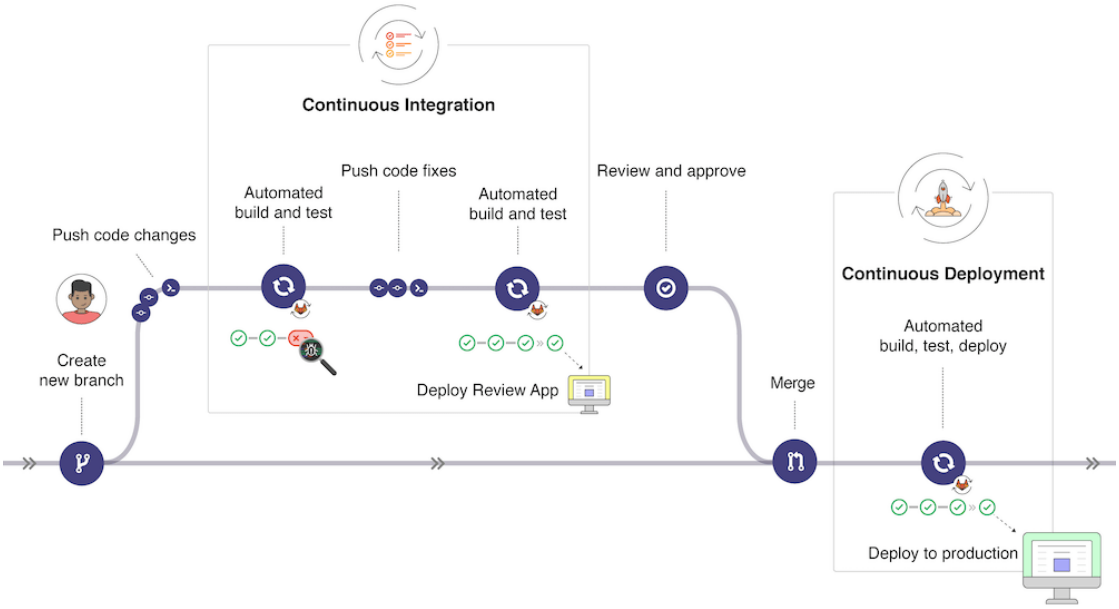

Figure 4.5: A CI/CD pipeline is a seamless way for us to make changes to code that are then automatically tested and pushed out for delivery and deployment. The goal is to eliminate downtime [51]

timing and location, which bring products and new features to the end-user faster. Ultimately, this facilitates optimal efficiency and effectiveness in task execution by decreasing tedious and time-consuming manual development work and legacy approval processes. Automation makes processes predictable and repeatable so that there is less opportunity for error from human intervention.

The idea behind the development of the EidoData web application was to enable its deployment in any public or private cloud. Several third-party services for CI/CD and public cloud services will be shortly introduced before outlining the entire development process. GitHub Actions is a major player in the CI/CD market. It can easily replace most CI/CD tools, especially if code is shipped as container images (cf. section 2.6.1). GitHub Actions offers a serverless platform that can handle the majority of development operations, which makes automating jobs easier. From a developmental perspective, transitioning between multiple services to diagnose build errors can compromise efficiency. Minimizing the number of environments engaged on a routine basis can optimize productivity. A workflow is an automated procedure that can be configured to execute one or more operations. Workflows are defined by a YAML file that is checked into the repository and performed when triggered by an event manually, or according to a set schedule. Upon each new commit submission to GitHub, the system initiates the GitHub Actions workflow process. GitHub Actions will build a docker image in its runner and push that image to a container registry. GitHub Actions will then connect to a public cloud provider, such as Google Cloud Platform, and deploy the image from the container registry to the cloud Kubernetes engine. Figure 4.6 shows the pipeline of using GitHub Actions to build, test and deploy a web application to the Google Cloud Platform.

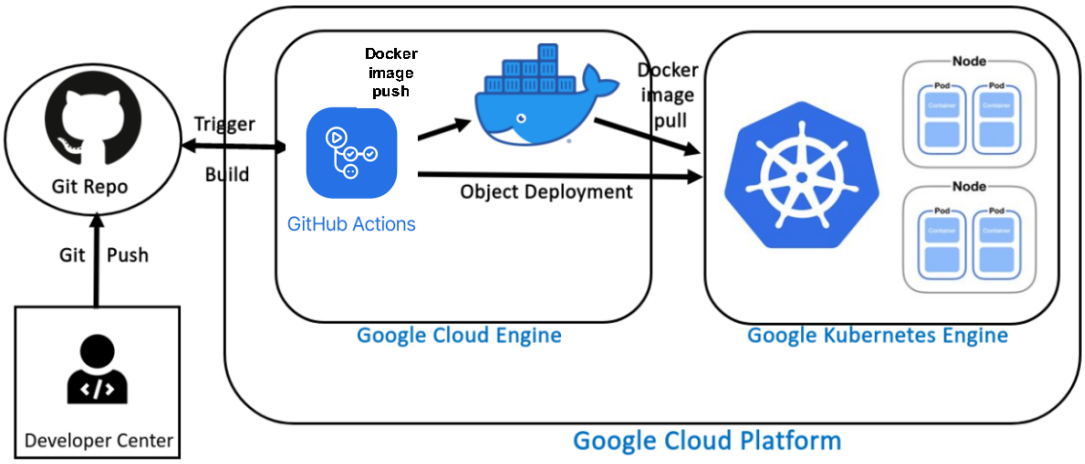

Figure 4.6: GitHub Actions help us to build the image, push it to Google container registry and trigger the deployment.

4.3.2 Features of EidoData web

Aside from running on the cloud, the EidoData web also provides some additional features, which are presented in this section. In terms of feature engineering (cf. section 2.4.1.2), EidoData web can combine two features through an arithmetic operation, which can be helpful in explaining variances in data. Creating a new feature through the interaction of existing ones is known as feature interaction. More specifically, feature interaction creates new features by multiplying two variable ($x_1 \cdot x_2$), while feature ratios create new features by calculating the ratios of existing features (x_1/x_2). In ML experiments the relationship between the dependent and independent variable is often assumed as linear. However this is not always the case: sometimes the relationship between dependent and independent variables is more complex. Creating polynomial features sometimes might help in capturing that relationship which otherwise may go unnoticed. EidoData web therefore also supports creating polynomial features from existing features. For instance, if an input sample is two dimensional and of the form $[x_1, x_2]$, the polynomial features with degree 2 are: $[1, x_1, x_2, x_1^2, x_1 \cdot x_2, x_2^2]$.

When a dataset contains features that are related to each other in some way, for example features recorded at some fixed time intervals, the new statistical features such as mean, median, variance and standard deviation for a group of such features can be created from existing features using the *group features* function in the EidoData web. *Feature binning* is a method of turning continuous variables into categorical values using a predefined number of bins. It is effective when a continuous feature has too many unique values or few extreme values outside the expected range. Such extreme values influence on the trained model, thereby affecting the prediction accuracy of the model. In EidoData web, continuous numeric features can be binned into intervals using *feature binning* function. The number of bins can be determined following Sturges' rules [149] or using K-Means clustering to convert continuous numeric features into categorical features.

A categorical feature with a lot of levels might sometimes be encountered in a dataset; these are known as high cardinality features. If these feature or features are encoded (cf. section 2.4.1.2) into numeric features, the resulting matrix is sparse. Due to the manifold increase in the number

of features and resulting increase in dataset size, this not only slows down the computations but also introduces noise. By merging the rare levels in the feature or features that have cardinality, a sparse matrix can be avoided. This can be achieved in EidoData web using the *combine rare levels* function. Creating clusters using the existing features from the data is an unsupervised ML technique (cf. section 2.4) to engineer and create new features. It uses an iterative approach to determine the number of clusters using a combination of the Calinski-Harabasz [16] and the silhouette criterion [137]. Each data point with the original features is assigned to a cluster. The assigned cluster label is then used as new feature in predicting target variable. This can be achieved in EidoData web using the *create clusters* function.

The web version of EidoData offers feature selection, which is another important distinction between it and the desktop version. Feature selection is a process used to select features in the dataset that contributes the most in predicting the target variable. Working with selected features instead of all the features reduces the risk of overfitting, improves accuracy, and decreases the training time. In EidoData web, this can be realised using the *feature selection* option. It uses a combination of several supervised feature selection techniques to select the subset of features that are most important for modeling. Multicollinearity is a phenomenon in which one feature variable in the dataset is highly linearly correlated with another feature variable in the same dataset. Multicollinearity increases the variance of the coefficients, thus making them unstable and noisy for linear models. One way to deal with multicollinearity is to drop one of the two features that are highly correlated with each other. This can be achieved in EidoData web using the *remove multicollinearity* function. Finally, a dataset may have a categorical feature with multiple levels, where the distribution of such levels is skewed and one level may dominate over other levels. In such cases, there is not much variation in the information provided by the feature. For a ML model, such feature may not add a lot of useful information and thus can be ignored for modeling. This can be done in EidoData web using the *ignore low variance* function. Figure 4.7 illustrates the interface of EidoData web for all aforementioned functions.

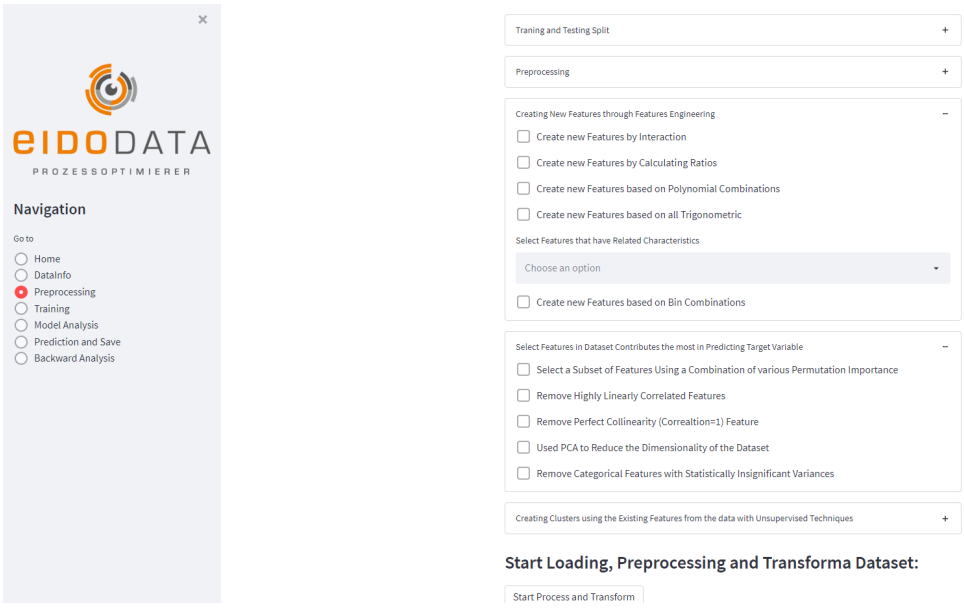

Figure 4.7: EidoData Web provides a friendly and intuitive interface for preprocessing, training and deploying a ML project.

Without having to worry about the hyperparameter setting for each ML algorithm, EidoData Web can train and evaluate the performance of almost all popular ML algorithms simultaneously. Besides the ML algorithms that were introduced in section 2.4.2, EidoData web also supports another 15 ML algorithms for regression. Figure 4.8 shows a list of available ML algorithms of EidoData web and their results together with the related metrics on an example dataset. Simultaneous execution of ML algorithms significantly speeds up the training process. It provides the single model that performs best based on the predefined criteria.

Model	MAE	MSE	RMSE	R2	RMSLE
AdaBoost Regressor	0.8589	1.4234	1.1188	0.578	0.0646
Extra Trees Regressor	0.866	1.4668	1.148	0.5506	0.0663
Random Forest Regressor	0.8906	1.4838	1.1542	0.547	0.0667
Gradient Boosting Regressor	0.8778	1.4329	1.1545	0.5364	0.0664
Light Gradient Boosting Machine	0.9299	1.6593	1.225	0.4881	0.0705
K Neighbors Regressor	0.9482	1.6774	1.2234	0.487	0.0701
Bayesian Ridge	0.9678	1.6533	1.2491	0.4393	0.0707
Huber Regressor	0.9802	1.6849	1.2616	0.427	0.071
Ridge Regression	0.9965	1.8783	1.3145	0.4231	0.0747
Linear Regression	0.9593	1.7651	1.2831	0.4173	0.0729
Least Angle Regression	0.9669	1.7761	1.2882	0.4133	0.0732
Orthogonal Matching Pursuit	1.0593	2.1364	1.3952	0.3556	0.0797
CatBoost Regressor	0.5227	0.9621	0.7039	0.316	0.0429

Figure 4.8: A comprehensive variety of ML algorithms are offered by EidoData web. This graphic displays the results of each ML algorithm's training result together with the related metrics.

4.4 Summary

The investigation of traditional methodologies for scrap purchase optimization shows a number of shortcomings. The constraints inherent to these methods motivate the progression towards a data-driven paradigm. This new approach leverages the complex relationships between process data and energy consumption, founded on the analysis of a substantial volume of historical data.

The traditional methods, while effective to an extent, often lack the flexibility and adaptability necessary to handle complex variables and scenarios. This is particularly noticeable in the context of energy consumption, where the interplay between different factors is intricate and often unpredictable. Therefore, these traditional methods often reach their limits, emphasizing the need for more sophisticated tools capable of unearthing insights from complex data sets.

This is where data-driven methodologies come into the picture. Using ML algorithms, these methods are able to identify and quantify the relationships between process variables and energy consumption, providing more accurate and nuanced insights. The approach uses historic collected data to predict future outcomes, giving foundries a more efficient way to optimize their energy usage based on their specific needs.

To this end, two user-friendly ML software tools have been introduced, which demonstrate significant utility for the foundry industry. These tools are designed to be easy to use, even for non-experts, and provide a way to harness the power of ML without needing specialized knowledge or training. This enables individuals and companies to leverage ML in their operations, enhancing their ability to optimize their purchased scrap, reduce energy consumption, and ultimately improve their bottom line. The software tools are a contribution to the democratization of ML, opening up its vast potential to a wider user base and making its benefits more accessible.

Chapter 5

Surrogate gradient methods for optimization

The widespread popularity of data-driven models is reflected in their expanding deployment as alternatives to traditional analytical descriptions or simulations, as hinted at in chapter 4. Within the context of the foundry industry, these models are instrumental in forecasting diverse results such as energy consumption, correlated with specific quantitative traits that derive from observed process parameters as well as control variables.

Transitioning from a purely predictive to a prescriptive analytics framework represents a significant progression. The transition encompasses the utilization of data-driven models to optimize processes. A key aspect of this procedure involves determining a control variable value, for given process parameters, that would improve output quality in accordance with the process model. Thus, prescriptive analytics, as an extension of predictive analytics, not only anticipates outcomes, but also suggests the course of action for optimal results.

The progression of this task inherently necessitates the formulation of a constrained optimization problem (cf. Equation 2.5.1) pertinent to data-driven predictive algorithms. Nevertheless, it has been observed that a significant proportion of optimal models in the current research landscape exhibit a lack of consistency. Predominantly non-differentiable methods, such as gradient boosting or random forest (cf. section 2.4.2.1), are frequent choices but they often demonstrate discontinuities. The optimization of these models requires the application of derivative-free techniques as discussed in section 2.5.2. Such a requirement recognizes the intrinsic complexities associated with these methodologies. It further highlights the imperative to develop advanced techniques to navigate the intrinsic challenges posed by these models and to capitalize on their predictive capacities.

In the following, the utilization of surrogate models is explored (cf. section 2.5.4), specifically independently differentiable ML models (cf. section 2.4.2.2), during the optimization phase. While it is generally acknowledged that these surrogate models may not be as precise in emulating the actual process, their capacity to utilize derivative-based optimization techniques (cf. section 2.5.3) offers significant benefits with regard to computational efficacy.

This exploration encompasses both theoretical frameworks, evidenced by traditional benchmarks, and practical application, emphasized by the analysis of a dataset derived from an actual industrial environment. Despite the increased model error observed in these surrogate models, the computational benefits they provide may render them favorable, especially within the realm of real-time applications. This is particularly evident when evaluating the cost-benefit analysis of computational speed versus absolute accuracy, a factor of paramount importance in real-time

optimization scenarios.

5.1 Gradient-based optimization of surrogate models

In this study, the primary aim is to optimize the composition of procured scrap mixtures. Fundamentally, these scrap mixtures influence three key cost components in foundry operations.

The first cost component under consideration is the direct expenditure associated with the acquisition of the scraps. This cost is determined by the market prices of the different types of scrap materials, the quantity required, and the contractual terms with the scrap suppliers. This necessitates an understanding of the metal scrap market dynamics and price fluctuations, which can significantly influence the cost of procurement.

The second cost component pertains to the logistics and transportation of the acquired scraps from the point of purchase to the foundry location. The transportation cost is typically a function of the distance between the supplier and the foundry, the mode of transport, and the total weight of the scrap materials.

The third and final cost component is related to the composition of purchased scraps that are fed into the furnace for melting. This is of particular significance as it can have a substantial impact on energy consumption and, consequently, energy costs. The energy required for melting different types of scrap materials varies, and therefore, the composition of the scrap mix can significantly influence the total energy cost. More energy-efficient scrap mixes can contribute to lower overall production costs and promote sustainable foundry operations.

In the present study, ML methodologies were utilized to predict energy consumption related to distinct instances of steel casting and particular furnaces. This attempt is of particular interest as it facilitates better management of energy resources in foundry operations, contributing to more sustainable and cost-effective practices. Specifically, in the context of structured, tabular data applications, the XGBoost algorithm (cf. section 2.4.2.1) has exhibited superior performance in specific instances when compared to NN (cf. section 2.4.2.2). XGBoost, a tree-based ensemble method, can effectively handle heterogeneous features often present in tabular data and offers robustness against overfitting, rendering it a promising tool for such predictive tasks. However, the prediction function of XGBoost is inherently non-smooth and non-differentiable because it is based on decision trees [23]. A decision tree, by its very nature, makes predictions using a series of binary splits on the input features. Each of these splits introduces a discontinuity in the prediction function, which makes it non-smooth. For instance, if a decision tree splits a feature at a value of 0.5, the prediction for a value 0.4999 could be significantly different from the prediction for a value of 0.5001.

Furthermore, the prediction function of XGBoost is unsuitable for derivative-based method even locally because of its foundational structure in decision trees. These trees produce constant outputs within the area of each leaf node. Therefore, within the area defined by any leaf node, the output does not change regardless of changes in the input, until the input moves across a decision boundary. This unchanging output results in a derivative value of zero. In other words, the prediction function of XGBoost is piecewise constant, leading to its derivative being zero almost everywhere, except at the decision boundaries where it's not defined.

Contrarily, the architecture of a NN (cf. section 2.4.2.2) is structured as a combination of operations that are inherently differentiable. Leveraging the principles of the chain rule in calculus, the derivative of a composite function is computed as the product of the derivatives of the constituent functions. Consequently, this implies that the prediction function of a NN is inherently differentiable, both with respect to its inputs and the parameters that dictate the network's behavior.

In addition to employing the COBYQA (cf. section 2.5.2.2) and COBYLA (cf. section 2.5.2.1) algorithms, this study also utilizes the NGOpt method and NGOptRW method [94], which are supplied by the *Nevergrad* library [128], a comprehensive and advanced derivative-free optimization framework. NGOpt is a meta-algorithm that adapts to the problem at hand by selecting and combining various optimization algorithms available within *Nevergrad*. The primary goal of NGOpt is to provide robust optimization performance across a diverse set of problems. To achieve this, NGOpt employs a two-layer approach:

1. Algorithm selection: NGOpt starts by choosing a suitable algorithm (or a combination of algorithms) from its pool of available optimizers. This selection process is based on the performance of these optimizers on a set of benchmark problems, which are representative of a wide range of real-world optimization challenges. By analyzing the problem features and comparing them to the benchmarks, NGOpt selects the most promising algorithms for the given task.

2. Algorithm adaptation: After selecting the appropriate algorithm(s), NGOpt fine-tunes the chosen optimizer(s) to better adapt to the specific problem. This is achieved by adjusting the optimizer's parameters or through other problem-specific adaptations. The optimization process then proceeds using the adapted algorithm(s).

Unlike NGOpt, which is optimized for artificial benchmarks, NGOptRW has been specifically developed to better suit real-world applications. NGOptRW is adjusted to be more effective in handling real-world problems, making its performance more suitable and practical for real-life scenarios [9].

5.1.1 General problem description

In the following, we will assume that $f, \hat{f}, \tilde{f} \colon \Omega \times \Phi \to \mathbb{R}$ are functions on a domain $D = \Omega \times \Phi \subset \mathbb{R}^N$ such that

i) $\|f - \hat{f}\|_{L^2(\Omega\times\Phi)} < \|f - \tilde{f}\|_{L^2(\Omega\times\Phi)}$,

ii) $\tilde{f}(\cdot, \phi) \colon \Omega \to \mathbb{R}$ is continuously differentiable.

In the given context, f denotes the theoretical function representing the actual process. This function, in an idealized sense, captures the true relationship governing the process under investigation. Meanwhile, $\hat{f}$ is an approximation of f that is suitably close with respect to the $\|\,.\,\|_{L^2(\Omega\times\Phi)}$. In other words, $\hat{f}$ serves as a data-driven model whose mean square error (MSE, cf. section 2.4.1.3), a measure of the model's predictive performance, is sufficiently low.

Furthermore, $\tilde{f} \in C^1(D)$, acts as a differentiable surrogate for $\hat{f}$. However, it is assumed that while $\tilde{f}$ retains the differentiability property, it provides a less accurate approximation of f compared to $\hat{f}$. This trade-off is often necessary when the original function or its direct approximation is non-differentiable, and a differentiable representation is required for certain analysis or optimization procedures.

For given parameters $\phi \in \Phi$, we would like to find

$$x^* = \operatorname*{argmin}_{x\in\Omega} f(x, \phi)\,.$$

In practice, of course, the functional relation f is usually unknown. As an approximation, we try to determine

$$\widehat{x}^* = \operatorname*{argmin}_{x\in\Omega} \widehat{f}(x, \phi)$$

instead. However, if $\widehat{f}$ is not sufficiently regular, derivative-free methods are required to compute a suitable approximation $\widehat{x} \approx \widehat{x}^*$. Due to the difficulties associated with derivative-free optimization, a close approximation of $\widehat{x}^*$ would generally require a high amount of computational resources which, for example in real-time applications, might not be readily available. On the other hand, using classical derivative-based methods, determining

$$\tilde{x} \approx \tilde{x}^* = \underset{x \in \Omega}{\operatorname{argmin}} \, \tilde{f}(x, \phi)$$

proves a much simpler task due to regularity assumption ii), although according to i), a higher deviation between $f(\widetilde{x}^*, \phi)$ and $f(x^*, \phi) = \min f(\cdot, \phi)$ needs to be expected. A simple example of this method is shown Figure 5.1.

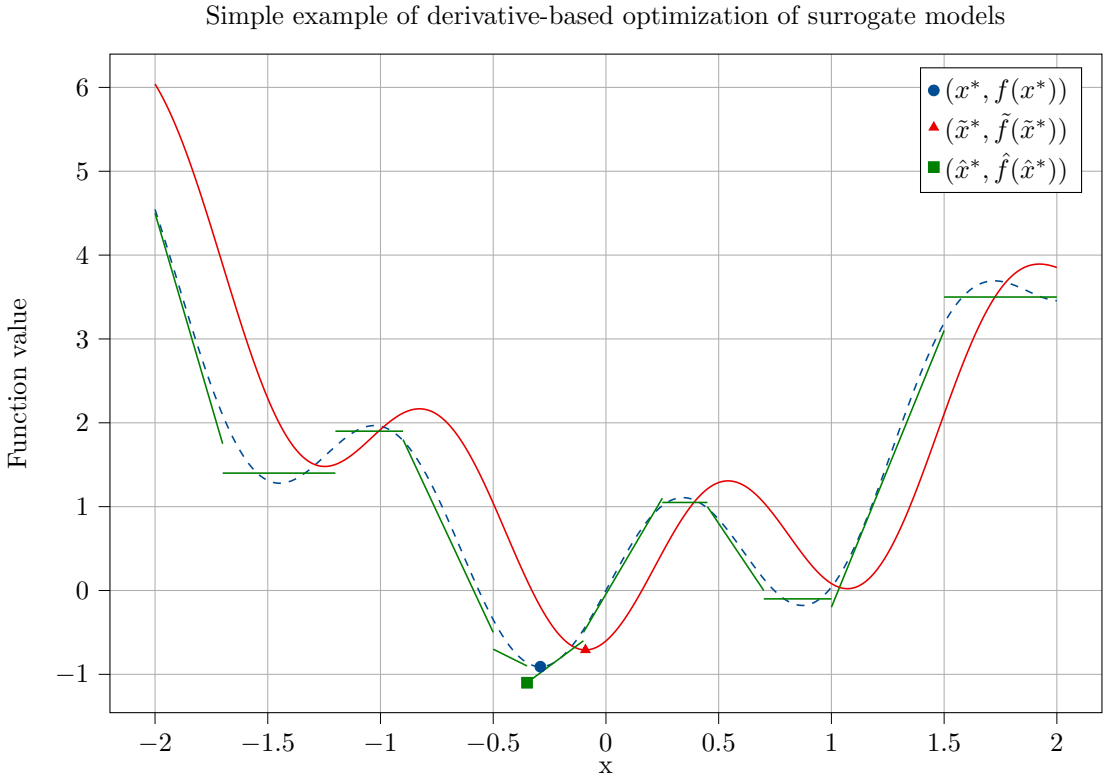

Figure 5.1: A simple demonstration of using the differentiable surrogate model.The blue curve represents the original function f. The green dashed curve represents the piecewise approximation $\hat{f}$. The red dash-dot curve represents the differentiable surrogate $\tilde{f}$.

5.1.2 Application to machine learning

More specifically, as depicted in Figure 5.2, the methodology can be delineated into two primary phases: training and optimization. In the training phase, two ML models, specifically XGBoost (non-differentiable) and a NN (differentiable), are trained. Subsequent to this training, a comparative analysis is conducted based on the MSE to ascertain the relative performance of the two models. Transitioning to the optimization phase, the procedure distinguishes two cases based

on which is the superior model: if the NN exhibits a lower MSE, the optimization of the input variable X is undertaken using derivative-based optimization techniques, such as SLSQP. The objective function for this optimization is derived from the prediction function of the NN. The final output is then the NN's prediction on the optimized input feature, denoted as X^*.

Conversely, if XGBoost demonstrates a superior performance metric, the NN is leveraged as a surrogate model. This is primarily due to the NN's capability to furnish derivative information pertinent to the loss function. The optimization of X still employs derivative-based optimization techniques with the NN's prediction function serving as the objective. However, post-optimization, the prediction on the optimized input X^* is conducted using XGBoost, attributed to its lower MSE.

5.1.3 Main objective

Our primary objective of to conduct a comprehensive examination of the consequential trade-off between a model's predictive accuracy and the computational resources necessitated for its optimization. Specifically, we seek to clarify whether within the paradigm of data-driven modeling applied to industrial processes, circumstances can arise where it may be advantageous to utilize a model characterized by lower accuracy but equipped with more regular, or smoothly varying, behavior for the purposes of optimization. This balance between precision and computational efficiency is a critical aspect of model selection in the context of real-world applications, where resource limitations often constrain the complexity of the models that can be feasibly employed. Our study seeks to provide valuable insights into these trade-offs, informing decision-making in the practical application of data-driven models to industrial processes. This understanding could guide the choice of modeling approach, potentially favoring simpler, more computationally tractable models that may offer sufficient accuracy for the task at hand.

5.2 Numerical experiments

In this investigation, the proposed methodology was employed on two distinct datasets: one being a popular benchmark dataset used in related studies, and the other being a dataset procured from real-world operations in the foundry industry. The successful application of our method across these diverse datasets highlights its robustness and proficiency in managing varied data types and domains, thereby validating its suitability for broad application in different settings.

5.2.1 Rosenbrock function

The Rosenbrock function, as initially proposed by Rosenbrock in 1960 [136], represents a well-established benchmark problem within the realm of numerical optimization. Since its inception, this function has been extensively employed as a standard measure to evaluate and compare the efficacy of various optimization algorithms. The initial definition of the function is as follows:

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2,$$

with a global minimum at (1, 1), as shown in Figure 5.3. We will be using a multidimensional extension of this problem [53], which is given by

$$f(x) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n-1} \left[(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2\right] .$$

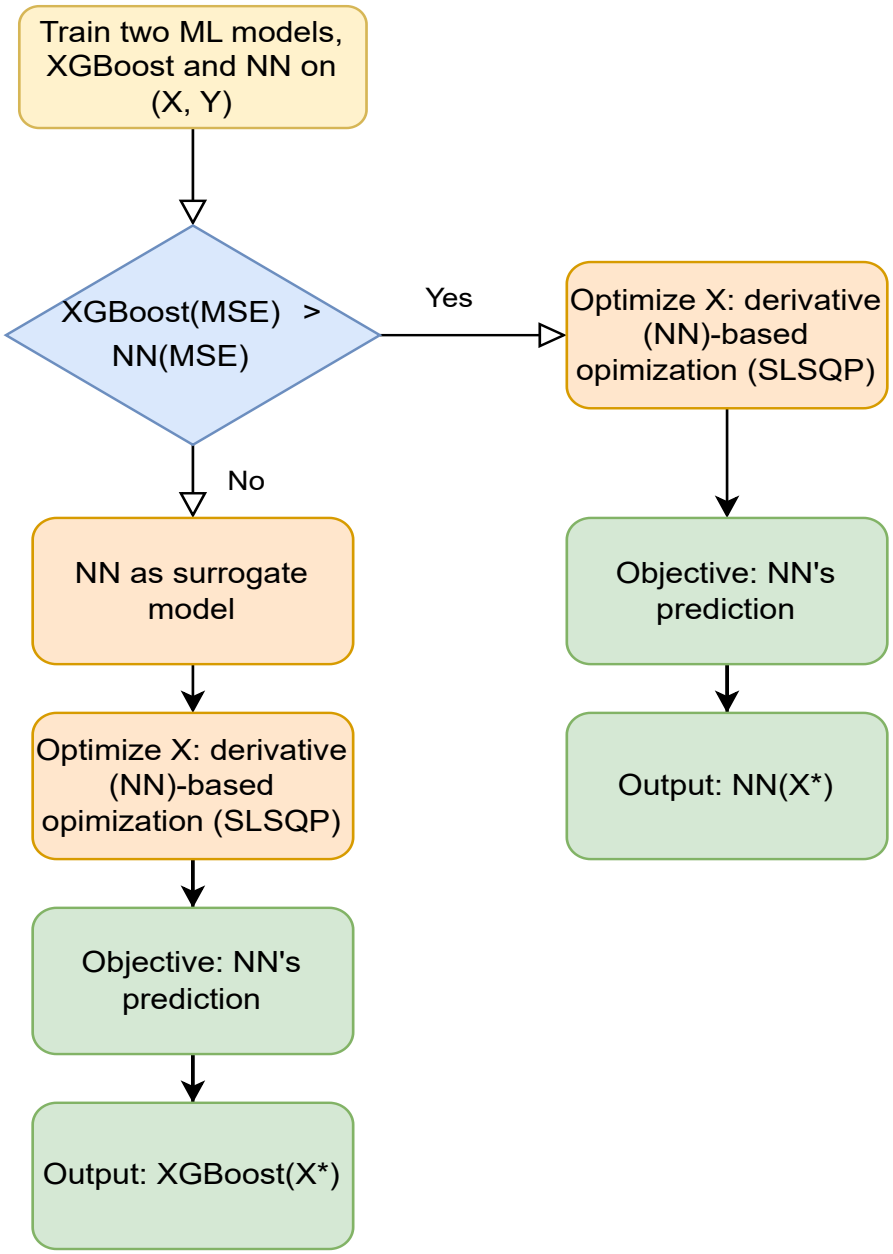

Figure 5.2: Flowchart for the derivative-based optimization of surrogate models method.

Additionally, we select $m \in \{5, 10\}$, with m representing the quantity of variables that are randomly designated as constant. The corresponding equality constraints are

$$\sum_{i=1}^{5} x_i = 25 \qquad \text{and} \qquad \sum_{i=1}^{10} x_i = 50\,, \tag{5.2.1}$$

respectively.

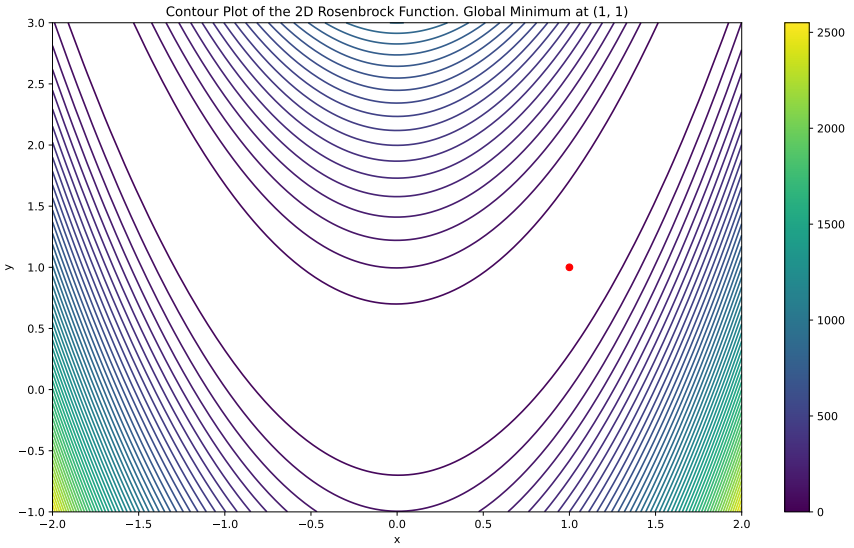

Figure 5.3: Contour plot for 2D Rosenbrock function.

In the context of the Rosenbrock test function, we initially designate $n = 10$ as the number of variables and choose $m = 5$ constant variables. The optimization of the input variables is facilitated through the utilization of XGBoost model ($\hat{f}$) and NN model ($\tilde{f}$) as objective functions, implementing derivative-free (cf. section 2.5.2) and derivative-based optimization (cf. section 2.5.3) methodologies respectively. Following the optimization process, we proceed to compute the Rosenbrock function's value using the optimized variables.

Subsequently, we extend our analysis to include the Rosenbrock function with $n = 15$, while maintaining $m = 5$ or $m = 10$ constant variables. The intention behind varying the count of variables and constants is to examine the performance of the optimization algorithms under different complexities, which is critical in establishing their robustness and applicability in diverse scenarios. To further strengthen the reliability and validity of our findings, we conducted 35 independent simulations for each configuration.

More specifically, we generate a synthetic dataset comprising 50,000 data points using a truncated normal distribution within the range $[0, 10]$. Each dataset corresponds to a specific dimensionality n. The Rosenbrock function values are then computed for each data point, and the datasets are split into training and testing datasets with a ratio of 90% and 10%, respectively. The training phase involves fitting both XGBoost and NN models to the training data. The models are tuned to approximate the Rosenbrock function, learning the underlying patterns in the data.

In this synthetic example, the Rosenbrock function itself represents the underlying functional relationships within the simulated process, which is unknown in practice, with the m fixed parameter values as the observable quantities influencing the process and the $n - m$ variables as the controllable parameters. The ML models, which were trained on the limited dataset (representing observations from the process), exhibit significant differences in their accuracy, with an MSE-measure of 23,296 for the XGBoost model and MSE $= 117,658$ for the NN model. Since the non-continuous XGBoost model provides a better fit to the data, these two ML models

can be used to investigate different combinations of optimization algorithms, as described in section 5.1.

More specifically, the following optimization strategies are employed for comparison:

- the XGBoost model is optimized directly with the gradient-free methods NGOpt and NGOptRW;
- the XGBoost model is optimized with SLSQP using the gradients of the NN model function;
- the NN model function is optimized with SLSQP;
- the original Rosenbrock function itself is optimized with SLSQP and NGOptRW.

The results derived from these numerical experiments are subsequently presented in Table 5.1. Figures 5.4 – 5.6 provide a comparative analysis of the results, employing both line plots and box plots. Furthermore, Figures 5.7 – 5.9 illustrate the processing time associated with each optimization method and deviations from the constraints, respectively. All computations were performed using an Intel Xeon Gold 6140 Processor with 128 GB and an NVIDIA GeForce RTX 3090.

Table 5.1: Optimization outcome for the Rosenbrock function. The dataset comprises 50,000 data points, which were partitioned into a ratio of 90% for training and 10% for testing. The MSE was assessed using the test dataset to evaluate the model's performance. When the objective function is defined as the Rosenbrock function, the optimization algorithms operate directly on this function. In such scenarios, ML models do not contribute to the optimization process.

n	m	MSE	Objective	Gradient	Optimizer	Time (s)	Rosenbrock	Constraint deviations
10	5	23 296	XGBoost	-	NGOpt	35.89	650 579	0.5452 %
			XGBoost	-	NGOptRW	13.18	555 887	0.5472%
		117 658	XGBoost	NN	SLSQP	0.58	386 743	0.0002%
			NN	NN	SLSQP	0.28	386 118	0.0003%
		0	Rosenbrock	Rosenbrock	SLSQP	0.04	381 696	0.0%
			Rosenbrock	-	NGOptRW	16.16	524 740	0.5233%
15	5	37 696	XGBoost	-	NGOpt	35.87	979 301	0.4834%
			XGBoost	-	NGOptRW	13.96	801 651	0.4581%
		148 261	XGBoost	NN	SLSQP	0.67	691 395	0.1376%
			NN	NN	SLSQP	0.27	696 118	0.0006%
		0	Rosenbrock	Rosenbrock	SLSQP	0.04	673 113	0.0 %
			Rosenbrock	-	NGOptRW	15.37	835 557	0.5017%
15	10	37 696	XGBoost	-	NGOpt	36.00	1 466 969	0.5629%
			XGBoost	-	NGOptRW	13.65	966 270	0.5391%
		148 261	XGBoost	NN	SLSQP	1.25	532 398	0.0767%
			NN	NN	SLSQP	0.34	536 971	0.0001%
		0	Rosenbrock	Rosenbrock	SLSQP	0.10	494 890	0.0 %
			Rosenbrock	-	NGOptRW	14.50	897 084	0.4879%

For $n = 10$ and $m = 5$, using the XGBoost model with the NGOpt optimization method required a computational time of 35.89 seconds and yielded a Rosenbrock value of 650,579.

Using NGOptRW, the computation time was reduced to 13.18 seconds, and the Rosenbrock value improved to 555,887. However, a significant enhancement in performance was observed when the SLSQP method was employed instead, using the gradients of the NN model function for optimization: the computation time was reduced to 0.58 seconds, i.e., by more than an order of magnitude. The Rosenbrock objective value at the found optimum was 386,743, again showing a significant improvement over the derivative-free optimization using purely the (more accurate) XGBoost model. A further reduction in time to 0.28 seconds – with a comparable Rosenbrock value of 386,118 – was observed when using the NN model for the SLSQP target function, in combination with its own gradients. For comparison, even the direct optimization of the Rosenbrock function itself with SLSQP resulted in a comparable value of 381,696; however, the computational time was only 0.04 seconds in this case. Finally, employing NGOptRW directly on the Rosenbrock function took considerably longer (16.16 seconds) and resulted in a higher Rosenbrock value of 524,740.

For $n = 10$ and $m = 5$, the XGBoost model with NGOpt required a computational time of 35.87 seconds, yielding a Rosenbrock value of 979,301. When NGOptRW was applied with XGBoost, the time decreased to 13.96 seconds, and the Rosenbrock value improved to 801,651. The combination of XGBoost with NN gradients and SLSQP further optimized the performance, reducing the time to 0.67 seconds and achieving a Rosenbrock value of 691,395. Utilizing NN with its own gradients and SLSQP resulted in a time of 0.27 seconds and a Rosenbrock value of 696,118. As with the previous combination, the direct application of SLSQP to the Rosenbrock function was the most effective, requiring only 0.04 seconds to achieve a Rosenbrock value of 673,113. The direct use of NGOptRW on the function, while less efficient than SLSQP, yielded a Rosenbrock value of 835,557 in a time of 15.37 seconds.

In the final scenario with $n = 15$ and $m = 10$, the XGBoost model combined with NGOpt took 36.00 seconds and achieved a Rosenbrock value of 1,466,969. Using NGOptRW with XGBoost reduced the time to 13.65 seconds, resulting in a Rosenbrock value of 966,270. Combining XGBoost with NN gradients and SLSQP significantly improved efficiency, reducing the time to 1.25 seconds and yielding a Rosenbrock value of 532,398. When NN with its own gradients was used with SLSQP, the optimization took 0.34 seconds, achieving a Rosenbrock value of 536,971. Again, the direct application of SLSQP to the Rosenbrock function was highly efficient, requiring only 0.10 seconds to reach a Rosenbrock value of 494,890. When NGOptRW was applied directly, the Rosenbrock value was higher at 897,084, achieved in a time of 14.50 seconds.

Overall, The substantial difference in MSE values between XGBoost and the NN justifies the use of the NN as a surrogate model for gradient information. The proposed method, particularly when employing the NN for gradient information, offers a promising avenue for efficient and effective optimization on problems like the Rosenbrock function.

To further validate the efficacy of the proposed approach, one can investigate the initial values utilized by the optimization algorithms. For instance, one might initially employ XGBoost as the objective function and NGOptRW as the optimization algorithm. During each optimization, the NN is then used as the objective function, and the output x from the SLSQP serves as the initial value for the SLSQP optimizer. Conversely, one might begin with the NN as the objective function and SLSQP as the optimization algorithm. Subsequently, the generated x from SLSQP is employed as the initial value when XGBoost is the objective function, and this x is input into the NGOptRW optimizer. The final step involves a comparative analysis of the Rosenbrock values and computational times from both procedures. The result is shown in Table 5.2.

Upon analyzing the results across different combinations, it is evident that the optimization strategy employing initial values from SLSQP significantly influences the performance metrics. Specifically, for all tested combinations, the second approach, which utilizes the optimized x from SLSQP as the initial value, consistently demonstrates superior performance in terms of

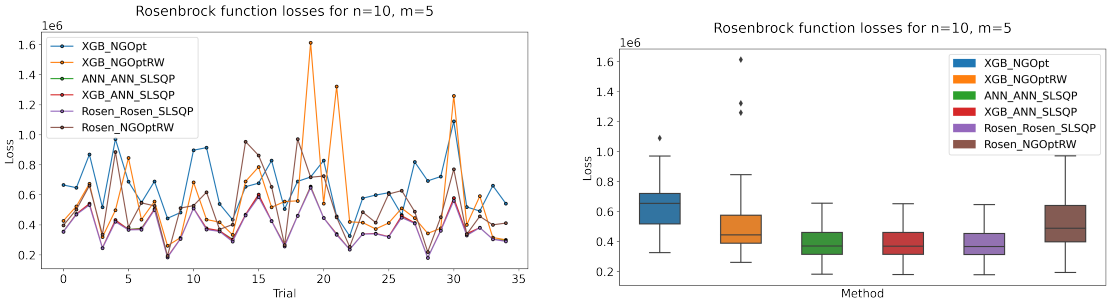

Figure 5.4: Rosenbrock function losses for n=10, m=5

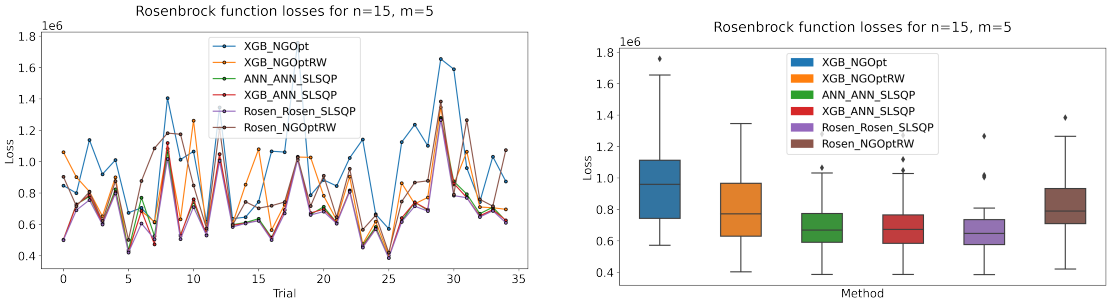

Figure 5.5: Rosenbrock function losses for n=15, m=5

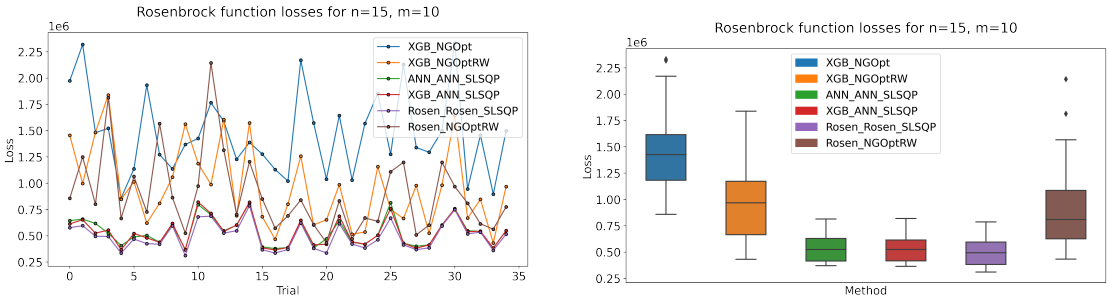

Figure 5.6: Rosenbrock function losses for n=15, m=10

Table 5.2: Comparison of results for initial values across different combinations.

Combination	Initial Value	XGBoost + NGOptRW		Initial Value	NN + SLSQP	
		Losses	Time (s)		Losses	Time (s)
(10, 5)	-	608 937.46	16.24	-	386 114.90	0.265
	SLSQP	371 116.98	14.25	NGOptRW	386 118.35	0.269
(15, 5)	-	924 567.41	15.39	-	696 115.78	0.251
	SLSQP	436 499.53	11.91	NGOptRW	696 118.40	0.265
(15, 10)	-	1 081 367.38	13.70	-	536 966.28	0.326
	SLSQP	498 501.25	11.69	NGOptRW	536 971.13	0.318

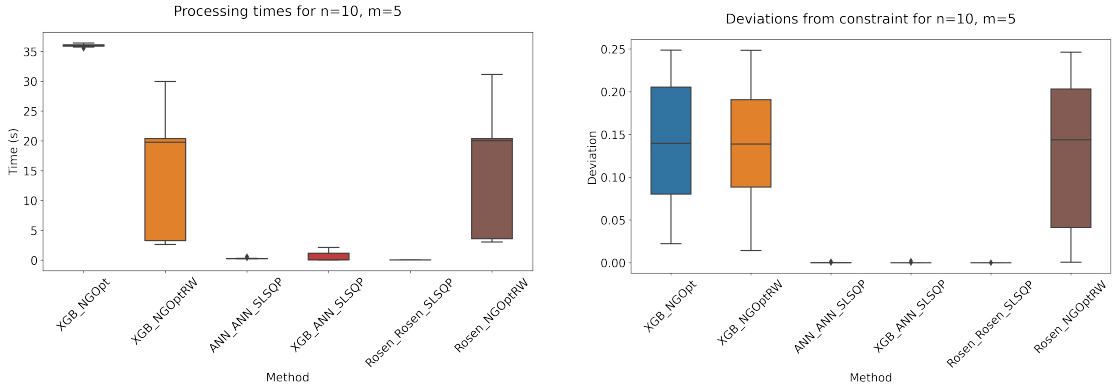

Figure 5.7: Processing time and deviations from constraint for n=10, m=5

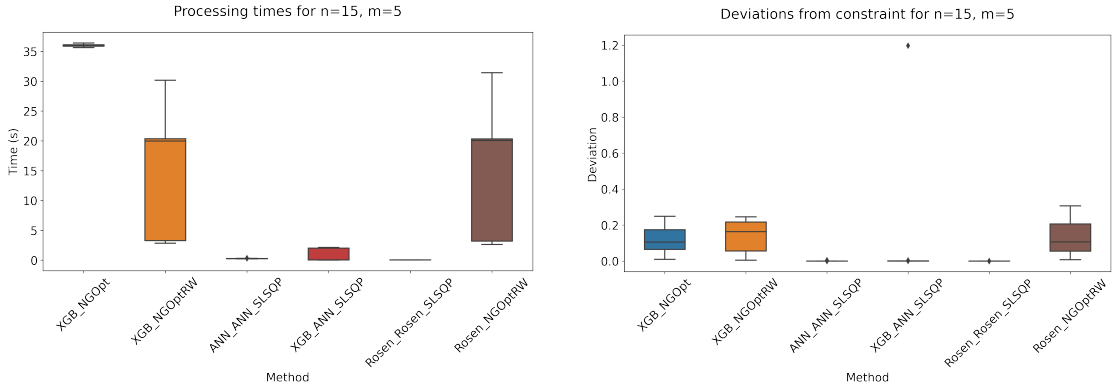

Figure 5.8: Processing time and deviations from constraint for n=15, m=5

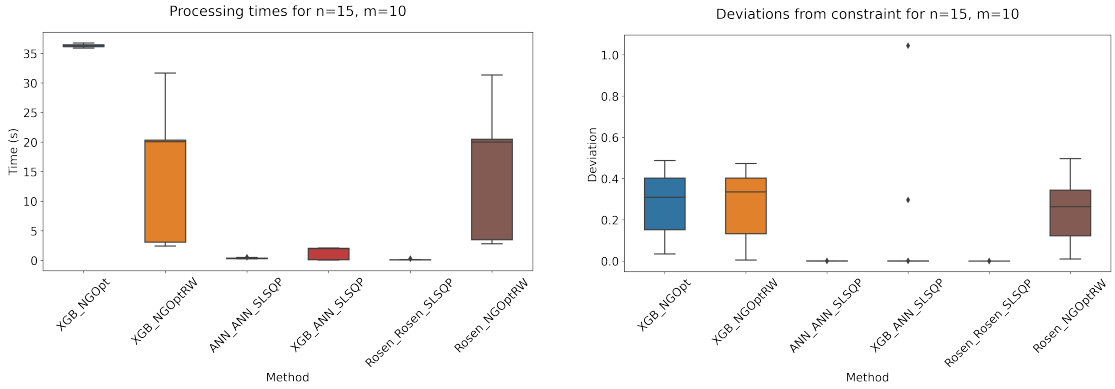

Figure 5.9: Processing time and deviations from constraint for n=15, m=10

both losses and computational time. For the combination (10, 5), the second approach reduced the NGOptRW losses from 608,937.46 to 371,116.98. Similarly, the computational time was reduced from 16.24 seconds to 14.25 seconds, indicating a more efficient optimization process. The combination (15, 5) exhibited a reduction in NGOptRW losses from 924,567.41 to 436499.53. The process time also saw a reduction from 15.39 seconds to 11.91 seconds. Lastly, for the

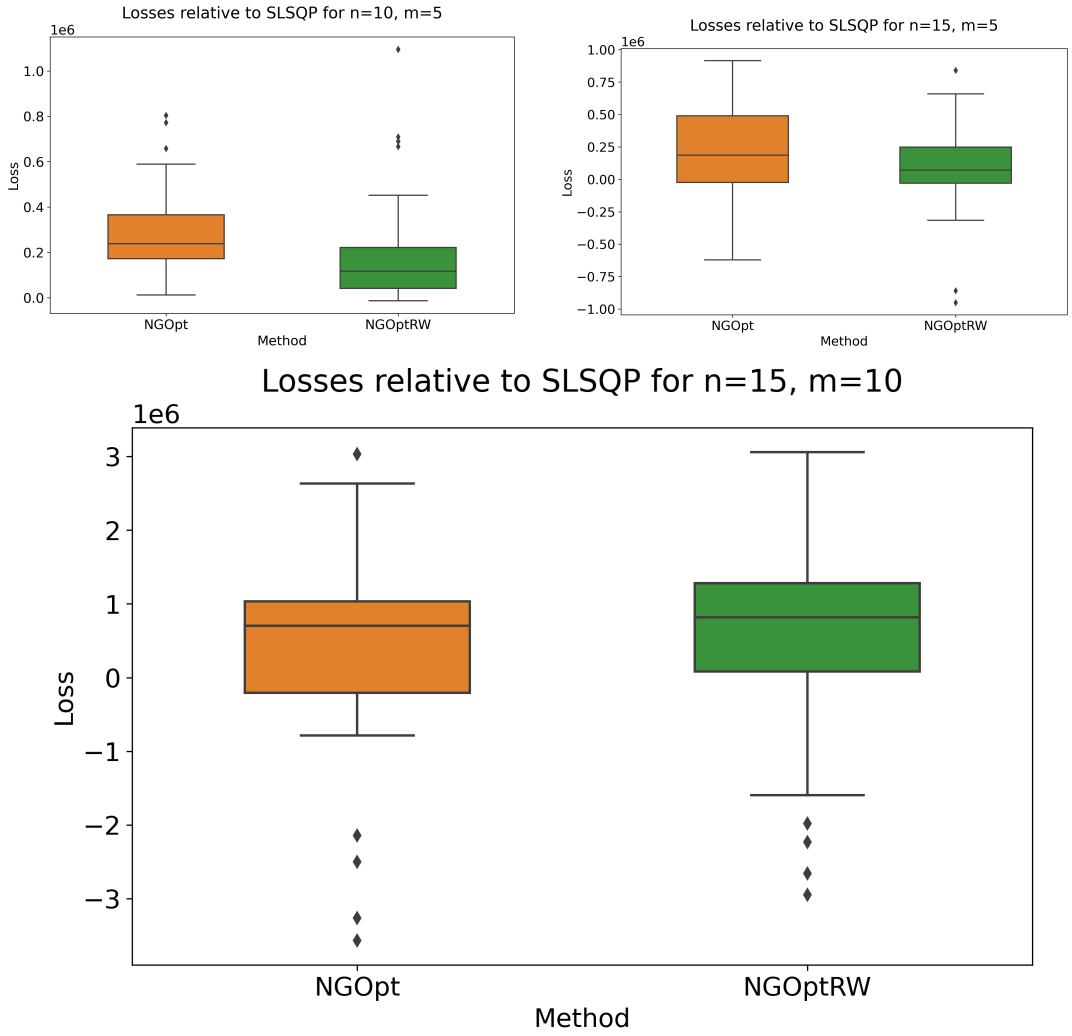

Figure 5.10: Derivative-free methods losses relative to SLSQP

combination (15, 10), the NGOptRW losses decreased from 1,081,367.38 to 498501.25. The computational time was optimized from 13.70 seconds to 11.69 seconds.

NN, being differentiable, can provide gradient information. This gradient information is crucial for optimization algorithms like SLSQP that leverage gradient-based methods. The gradient essentially provides a direction in which the function increases most rapidly. By knowing this direction, optimization algorithms can make more informed steps, leading to faster convergence and potentially better solutions. The choice of initial values can determine which minimum the optimization algorithm converges to. By using the output from one optimization process (SLSQP in this case) as the initial value for another, we're essentially providing a "warm start", which can lead to faster convergence and potentially better solutions. The significant reduction in losses when using initial values from SLSQP suggests that the optimization landscape (as modeled by the NN surrogate) has regions of lower loss that are more easily accessible from these initial values. Mathematically, this implies that the trajectory followed by the optimization algorithm,

when started from these initial values, leads more directly to regions of the search space with lower objective function values. The reduction in computational time can be attributed to faster convergence. This faster convergence is likely due to the combination of gradient information from the NN and the improved initial values from SLSQP. Derivative-based methods can take larger, more informed steps, reducing the number of iterations required to converge to a solution.

5.2.2 Foundry dataset

For real-world foundry applications, the analysis of energy consumption during melting in a medium sized steel foundry was conducted. The study employs a dataset of 3500 individual melts comprising three distinct types of steel - cold work, tool, and high speed - with the target variable of interest being energy consumption, measured in kilowatt hours (kWh). The dataset features 85 distinct attributes, classified into three groups - process data, batch data, and induction furnace data. The batch data offers information on the scrap's type and quantity used in the melting process, along with its unique chemical composition. In the foundry, the raw materials are divided into the various groups of alloys, purchased scrap and internal scrap. The induction furnace data, on the other hand, offers insights into melting time, power and energy, while process data identifies any obstacles that occurred during the melting process or given time periods in the process, such as the time between finished melt and feedback from the spectral laboratory regarding the measured chemical analysis. Sourcing scrap from various suppliers remains crucial to the melting process, with the aim being to minimize the predicted energy consumption based on the amount of each type of scrap used in a melt batch. For each batch, the scrap's chemical composition must adhere to specific chemical composition standards to achieve the desired steel result. Moreover, the weight of the scrap used in the melting process should not exceed the final steel weight.

As specified (cf. section 1.1), the total cost is the sum of three components,

$$f(x) = p(x) + m(x) + l(x)\,. \tag{5.2.2}$$

We derive the energy expenditure function $m(x)$ utilizing data-driven methodologies. The immediate procurement cost $p(x)$ is given by the product of scrap weights and their respective market prices. To streamline our analysis within the context of this study, we incorporate the transportation expenditure $l(x)$, as follows:

$$l_j(x) = \begin{cases} 0, & \sum_j x_j \leq 10, \\ 2.5 \cdot (x_j - 10), & 10 < \sum_j x_j \leq 50, \\ 100, & \text{otherwise}\,, \end{cases} \tag{5.2.3}$$

where x_j means the scrap purchasing weight from scrap provider j. The aim of this investigation using the foundry data set was to optimize total cost $f(x)$ by adjusting the quantities of purchasing scrap materials present in a melt batch, while keeping process parameters and furnace characteristics constant in the model. In this context, this means that parameters that are not control variables are kept constant or within pre-defined ranges for optimization. Examples include the tapping temperature, which typically needs to be between 1550 and 1650°C depending on the steel grade; the melt mass, which cannot exceed 7 tonnes due to the capacity of the induction furnace used; the tapping mass, which depends on the type of semi-finished product to be cast with the given melt; or time periods, such as the time frame described earlier between the final melt and its chemical composition evaluation, which depends on how much time human workers in the laboratory need to evaluate the melt sample. We also added constraints to the model based on the stock of different types of scrap available in the foundry.

Due to the limited capability of NGOpt and NGOptRW in handling equality constraints effectively, the foundry dataset optimization exclusively employs the COBYQA and COBYLA optimizer for derivative-free optimization. This decision ensures a more reliable and efficient optimization process tailored to the specific requirements of the dataset. The XGBoost algorithm exhibits an MSE value of 120.85, while the NN model attains an MSE value of 163.11, signifying a discrepancy in their predictive performance. To optimize the quantities of scrap and reduce energy use, we first use derivative-free methods, specifically COBYLA and COBYQA, to adjust the input parameters, using the XGBoost model as the objective function. Afterwards, the scrap composition is optimized using SLSQP with the NN model (the differentiable surrogate model) as the objective. In both cases, the final energy consumption is computed using the XGBoost model as the best current available model for the actual energy consumption.

The findings of this analysis are illustrated in Table 5.3 and Figure 5.11. Overall, 600 independent simulations were conducted to ensure the robustness of the results. Consistent with the case of the Rosenbrock function, the NN model utilizing the SLSQP optimizer yields marginally decreased total cost. The XGBoost model with the COBYQA optimizer showed no constraint violations and a total cost of 4437.8153. However, it required a considerably longer processing time of 491.787 seconds. When employing the COBYLA optimizer with the XGBoost model, the processing time was reduced to 58.999 seconds, albeit at the cost of a slightly increased total objective value of 4494.9615 and a small constraint violation of 0.00998%. On the other hand, the NN model with the SLSQP optimizer demonstrated competitive results in terms of total cost (4430.6569) and constraint violations (0.00291%). Moreover, it exhibited a significantly shorter processing time of 15.731 seconds, showcasing its efficiency in terms of computational resources.

The two-sided t-test was performed to assess the equality of means between the SLSQP optimizer and both the COBYLA and COBYQA optimizers. The t-test resulted in a p-value of 7.037e-45 for the comparison between SLSQP and COBYLA, indicating a highly significant difference between the means of these two optimizers. In contrast, the p-value for the comparison between SLSQP and COBYQA was 0.00769, which still suggests a statistically significant difference between their means, albeit less pronounced than that between SLSQP and COBYLA. In conclusion, the results of the two-sided t-tests provide strong evidence that there are statistically significant differences in the performance of the SLSQP optimizer compared to both the COBYLA and COBYQA optimizers.

MSE	RRMSE	Objective	Gradient	Optimizer	Time (s)	Energy consumption	Constraint deviations	P-value
120.85	0.0140	XGBoost	-	COBYQA	491.787	4437.8153	0.0%	0.0076
		XGBoost	-	COBYLA	58.999	4494.9615	0.00998%	7.023e-45
163.11	0.0296	NN	NN	SLSQP	15.731	4430.6569	0.00291%	

Table 5.3: Foundry dataset optimization result.

Upon examination, it is evident that the derivative-based surrogate modeling approach exhibits superior performance in comparison to derivative-free data-driven methodologies, with regard to both computational efficiency and the accuracy of the ultimate outcome.

5.3 Summary

The above analysis provides a detailed introduction to the concept of derivative-based optimization of surrogate models, establishing the foundational principles. Two numerical experiments

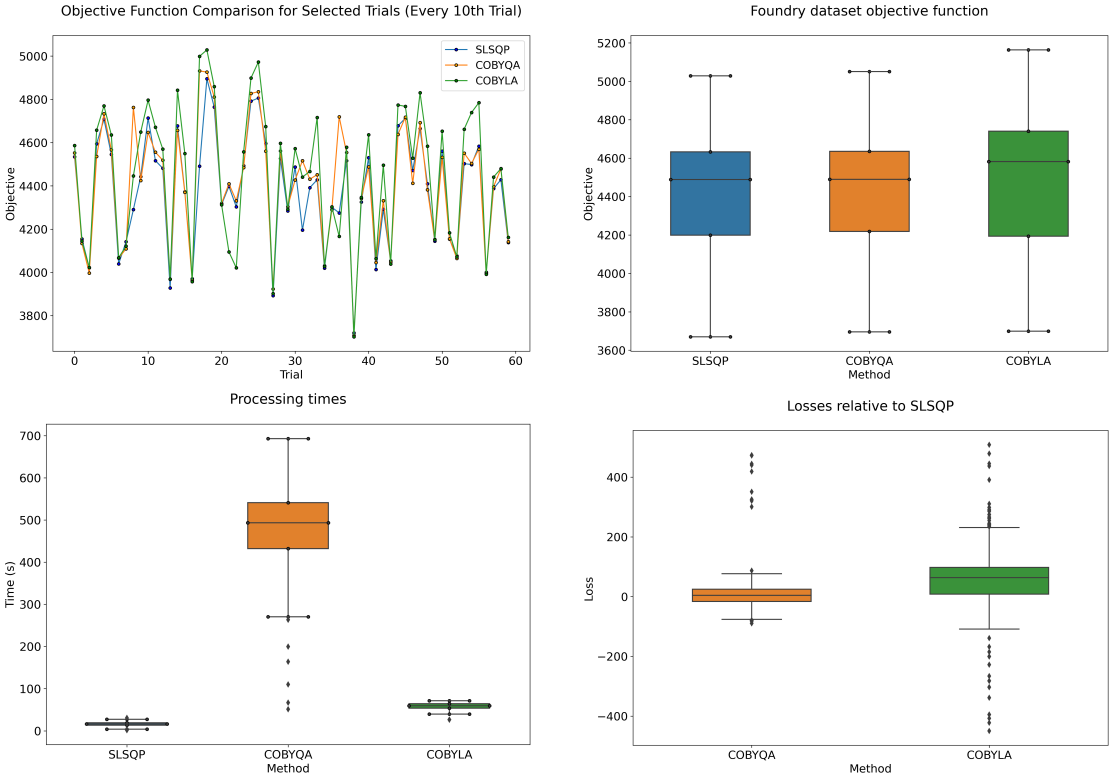

Figure 5.11: Foundry dataset optimization results.

are conducted to illustrate the practical applications of employing alternately trained, independent differentiable ML models as surrogate models. These experiments utilize both benchmark datasets and real-world foundry datasets, demonstrating the wide applicability of these techniques.

While the results suggest that the trained differentiable data-driven models may display a marginally lower level of accuracy, the advantages of using derivative-based optimization methods significantly enhance computational performance. It is these performance improvements that present a compelling case for the use of these methods, despite the potential for additional model error, as shown by the significant differences in the MSE for the investigated cases.

In particular, for real-time applications, the benefits of improved computational performance can outweigh the minor disadvantages in the attained objective values. This finding highlights the value of these methods in practical, real-world applications where speed and efficiency are often critical factors. The results thereby establish a promising avenue for future research and application in the optimization of surrogate models.

Chapter 6

Real-world scrap procurement simulations

Building on the insights collected from chapter 5, we further consider the practical implications of the finding within the foundry industry. It is well-established that the foundry industry is characterized by complex processes and systems that require efficient decision-making. Given the findings of the previous chapter, while differentiable data-driven models may exhibit a slightly reduced accuracy, their inherent computational efficiency makes them a viable alternative for real-time applications in the foundry setting. The potential marginal increase in the ML model error is outweighed by the speed and efficiency gains, which are crucial in an industry that operates on tight timelines and thin margins.

In the following, simulations of real-world scenarios within the foundry scrap procurement process are presented, exploring how derivative-based optimization methods can be integrated into existing systems and processes. Thereby, we investigate specific use cases, demonstrating how the benefits of improved computational performance can be harnessed to address real-world challenges and reduce overall operational cost.

6.1 Validation of the algorithm in real-world scenarios

As described in section 5.2.2, the foundry dataset's attributes can be divided into three groups: process data, batch data and induction furnace data. The batch data consists of three raw materials that are fed into the furnace, which are alloys, purchased scrap and internal scrap. In the following, the remaining input variables will be considered fixed parameters which, depending on their selected values, simulate the furnace conditions in a specific foundry. As previously, we train both a non-differentiable XGBoost model and a differentiable NN model on the dataset, with the output value given by the energy consumption.

As demonstrated in the previous chapter, any optimization involving data-driven models of the foundry process can benefit from the additional efficiency provided by surrogate gradient methods. In particular, these methods allow for a highly performant optimization of scrap purchasing decisions, i.e. the intended use of the scrap procurement platform described in chapter 3. Since this market platform is not actively in use yet, we will consider a simulation of the exchange between foundries and scrap providers in the following.

Recall that the general aim of using the ML models based on the foundry data described

above is to minimize the cost function f with

$$f(x) = p(x) + m(x) + l(x)\,,$$

where x is the scrap composition, $p(x)$ denotes the total purchase price of the scrap metal, $l(x)$ denotes the transportation cost and

$$m(x) = c \cdot \widehat{m}(x)\,, \tag{6.1.1}$$

where $\widehat{m}$ is the data-driven model of the energy consumption (in kWh) and c is the effective energy cost per kWh used in the process. The transportation cost $l(x)$, which accounts for varying transportation cost scenarios based on the total weight of scrap purchased from different providers, is given by

$$l(x) = \sum_j l_j(x)\,, \quad l_j(x) = \begin{cases} 0, & \sum_i x_{ji} \leq t_1, \\ 2.5 \cdot (x_{ji} - t_1), & t_1 < \sum_i x_{ji} \leq t_2, \\ t_3, & \text{otherwise}, \end{cases} \tag{6.1.2}$$

where x_{ji} is the amount of scrap of type i purchased from provider j. The individual transport cost function l_j transitions from a no-cost scenario (for negligible amounts of scrap, which are assumed to be not actually purchased for computational reasons) to a linearly increasing cost, and then to a flat rate cost as the weight of scrap purchased increases, as shown in Figure 6.1.

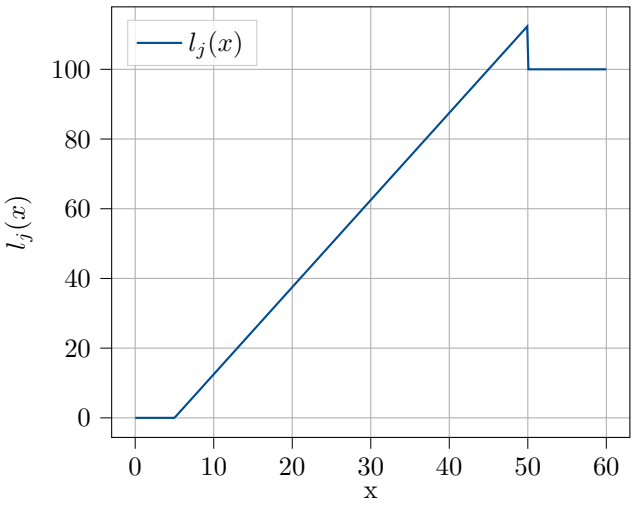

Figure 6.1: Plot of the function $l_j(x)$ for $[t_1, t_2, t_3] = [5, 50, 100]$.

The simulation is based on time-discrete interactions between the foundries and the scrap suppliers: each foundry requests an optimal order of scrap from the suppliers, who offer different stock at different prices. This process is repeated for multiple time steps. Each purchasing decision is based on the current price and availability of scrap, the individual foundry's operation conditions (simulated via different fixed parameter values for the data-driven model), the transport costs, the energy price per kWh and, of course, the predicted energy consumption, which depends on the scrap composition. Different optimization strategies are then employed to make an optimal purchasing decision in each time step, minimizing f under the constraints

given by the availability of scrap and the chemical composition required for the produced steel type (which we will assume to be 1.2379 throughout this chapter.)

Additionally, constraints are placed on the quantity of total scrap, ensuring it does not surpass 7 tons. In the context of this study, acceptable quantities range between 5 to 7 tons. Factoring in the prevailing electricity rates for the foundry industry in Germany in 2023, the electricity cost is benchmarked at 40 cent/KWh [83]. To further enhance the fidelity of the simulation to real-world dynamics, upon the determination of an optimal purchased scraps list by the algorithm, subtractions are made from the relevant scrap providers. This approach ensures that the available scrap and its corresponding quantity are updated in real-time throughout the simulation.

Obviously, a large-scale simulation of the interaction between foundries and scrap suppliers requires a large number of purchasing decisions, each of which requires a full optimization of the scrap composition. Therefore, a performant simulation is strongly dependent on a computationally efficient optimization strategy.

6.1.1 Preliminary results

Due to the newly trained models and the weight factor c introduced into the objective function in Equation (6.1.1), we again compare the performance of the two main algorithms from the previous chapter in Table 6.1 and Figure 6.2. In total, a comprehensive set of 250 independent simulations were executed to ascertain statistical significance. Note that once again, the surrogate gradient method, i.e. the optimization based on the less accurate, but differentiable NN model, outperforms the derivative-free optimization of the actual objective function given by the XGBoost model.

Table 6.1: NN and XGBoost results in real-world scenarios.

R^2	Objective	Optimizer	Time(s)	Total Cost	P-value
0.58	XGBoost	COBYQA	38.38	1690.94	9.64e-15
0.49	NN	SLSQP	7.09	1662.33	

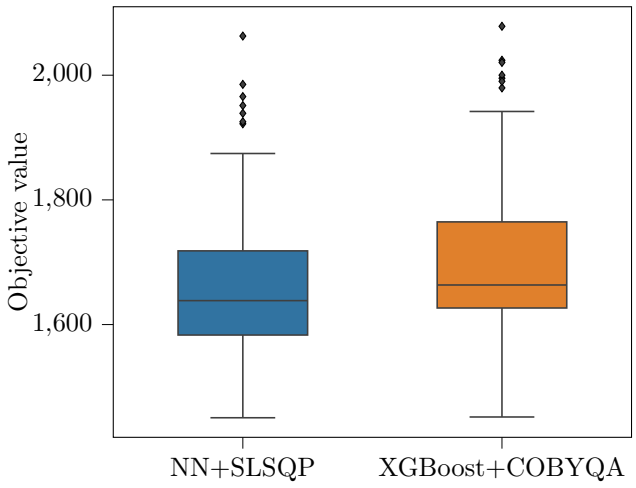

Figure 6.2: Results for gradient-free and surrogate gradient optimization.

Furthermore, in order to demonstrate that the optimization with respect to the full costs (including the energy costs) provides an actual benefit over the pure minimization of the scrap purchase price, we compare the final costs of a batch for two different scrap compositions: one is obtained as the optimum for the full cost function itself, as computed with SLSQP and the NN model; the other is calculated as the minimizer of the pure purchasing and transport costs. Table 6.2 shows that the difference between the average resulting costs over 250 simulations, while small, is indeed statistically significant.

Table 6.2: SLSQP optimize results in real-world scenarios with and without derivative information.

Objective	Optimizer	Total Cost	P-value
$l(x) + p(x) + m(x)$	$l(x) + p(x)$	1864.28	1.97e-14
$l(x) + p(x) + m(x)$	$l(x) + p(x) + m(x)$	1847.11	

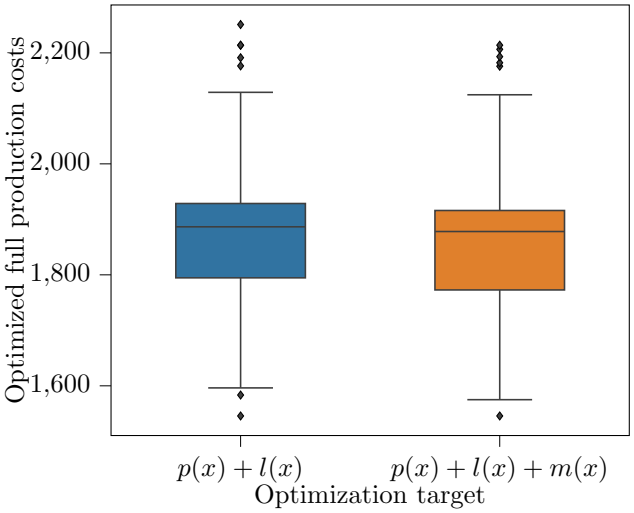

Figure 6.3: Optimization of the cost function with and without the energy term and the resulting full production costs.

6.2 Time discrete simulations

Table 6.3: Transport cost parameter settings for total energy consumption comparison.

Foundries	Scrap providers	$[t_1, t_2, t_3]$
8	10	$[0, 50, 80]$, $[5, 50, 100]$, $[80, 100, 100]$

In all our simulations, a hypothetical scenario is considered wherein eight foundries and ten scrap providers are interacting. Each foundry simultaneously forwards its purchasing requests to

the simulation program. It is noteworthy that process data and induction furnace data exhibit variability among the different foundries. To efficiently manage and process these concurrent requests, the simulation operates in parallel. This parallel operation facilitates the optimization of external scrap purchasing lists. The optimization is conducted employing the derivative-based surrogate model designed for the distinct operational and process characteristics of each foundry.

In the first simulation, the objective is to analyze how the optimization responds to higher transportation costs, hypothesizing that an increase in transportation cost would correspondingly drive the energy cost (i.e. the energy consumption) $m(x)$ higher post-optimization. As listed in Table 6.3, three sets of coefficients t_1, t_2, t_3 for the transport costs (cf. Equation (6.1.2)) are compared to study the effect of different transportation cost structures on the optimization outcome.

During the simulation, the average total energy consumption per batch is computed for each of the individual foundries over 50 requests. Figure 6.4 illustrates a clear trend: as the transportation cost increases, there is a corresponding rise in energy consumption. As the optimization is performed using SLSQP, which utilizes derivative information from $f(x)$ (cf. Equation 5.2.2) to find the optimal scrap purchase list x, the outcome should ideally balance out all three costs involved. However, with higher values of transport cost coefficients $[t_1, t_2, t_3]$, the balance tilts, and the optimizer has to navigate through a steeper cost gradient contributed by $l(x)$. The simulation result shows that a higher transportation cost could drive the system to select scrap which is not optimal in terms of energy consumption to avoid the increased transportation cost.

In our next simulation, we investigate the impact of transportation costs on the number of requests between foundries and scrap providers. Two extreme scenarios are considered for comparison to understand the range of behaviors the system may exhibit. The transportation cost parameters are organized in Table 6.4 with each foundry and scrap provider pair having associated cost coefficients t_1, t_2 and t_3. In this case, for 8 foundries and 10 scrap providers, the coefficients are set at the extremes for $t_3([100, 1000])$ to observe how the system responds to a wide range of flat rate transportation costs after a certain threshold of scrap weight is surpassed.

Table 6.4: Transport cost parameter settings for average request comparison.

Foundries	Scrap Providers	$[t_1, t_2, t_3]$
8	10	$[0, 5, 100]$, $[0, 5, 1000]$

Again, the simulation is performed over a length of 50 requests. Figure 6.5 shows the simulation result. In the low transport cost scenario $t_3 = 100$, the flat rate cost after the second threshold might not significantly deter requests between the foundry and scrap provider. This lower transportation cost could encourage a higher number of requests for scrap material between the foundry and the scrap provider, as the total cost $f(x)$ may still remain within a reasonable range. However, with $t_3 = 1000$ the substantial transportation cost could drastically reduce the number of requests as the cost of transporting scrap becomes prohibitively expensive.

This comparative analysis between the two scenarios highlights the sensitivity of the request frequency to the transportation cost structure, particularly the flat rate cost t_3. A higher t_3 value seemingly discourages a higher number of requests due to the substantial cost implications, as evidenced by the drop in average requests from 5.69 to 2.24 as t_3 increases from 100 to 1000. This showcases the critical role of transportation cost in influencing operational decisions and interactions between the foundry and scrap provider within this system. In particular, increased transport costs could lead to the environmentally preferable result that scrap is procured from fewer distinct sources and additional transports are avoided.

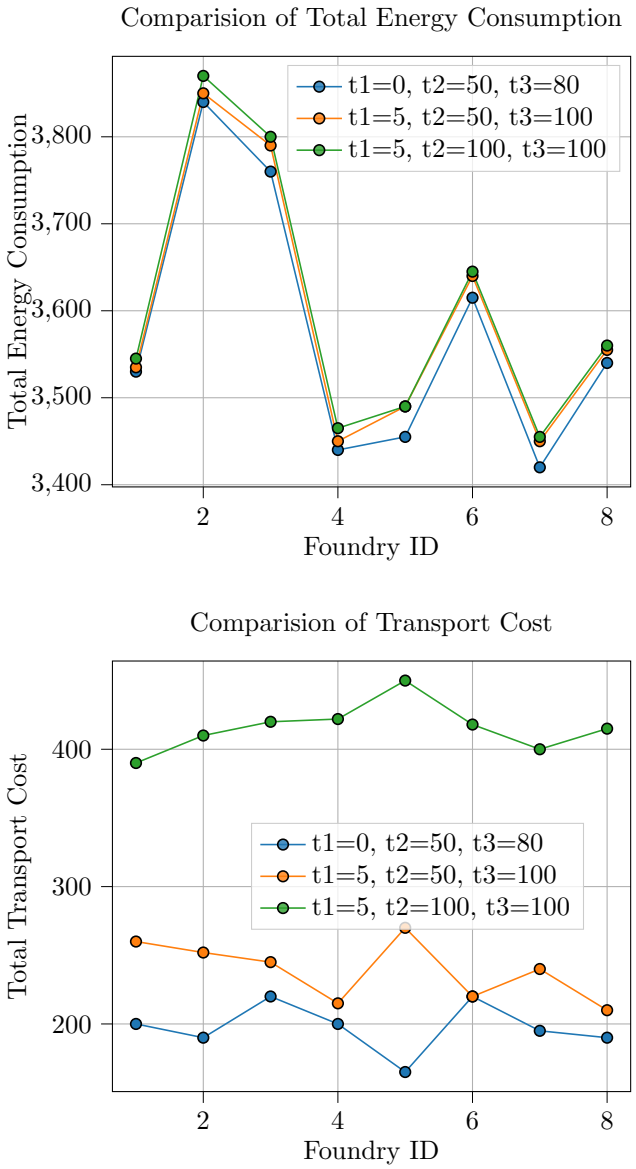

Figure 6.4: Comparison of total energy consumption with respect to the transport cost.

Finally, we consider the impact of the electricity price on the optimal scrap purchase decision. Since the energy cost is the product of electricity price and energy consumption, the electricity price can be considered a penalty factor to the energy cost $m(x)$, affecting the average energy consumption in the system after cost optimization. We simulate how fluctuations of the electricity price can affect the energy consumption of foundry processes by selecting the values shown in Table 6.5.

The simulation result, averaged over 50 requests, is shown in Figure 6.6. As the electricity price increases, it acts as a penalty factor to the energy cost $m(x)$, thereby potentially increasing

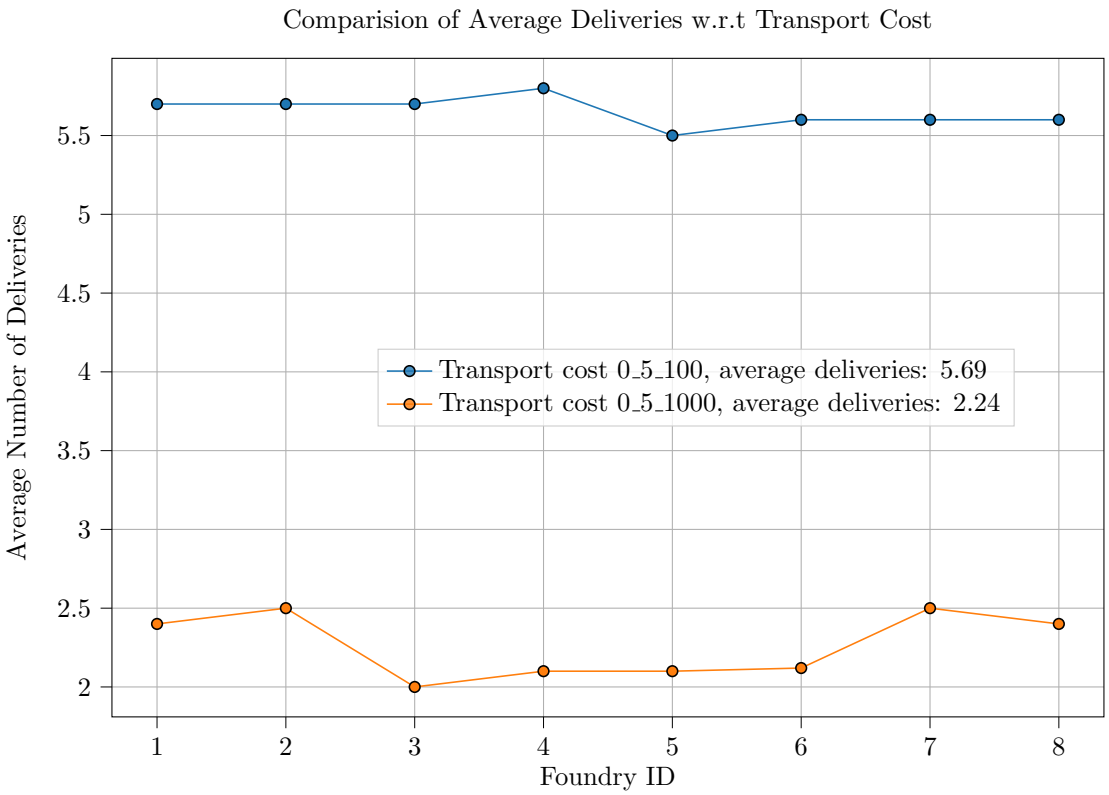

Figure 6.5: Comparison of purchase request with respect to the transport cost.

Table 6.5: Electricity price parameter settings for average energy consumption comparison.

Foundries	Scrap Providers	Electricity Price (Euro/KWh)
8	10	0.4, 0.8, 1.0

the total cost $f(x)$ if the energy consumption $m(x)$ remains constant. This higher penalty for energy consumption thus encourages the optimizer to look for solutions x that, among other things, might lead to lower energy consumption to balance out the extra cost from higher electricity prices. The optimizer is therefore navigating the trade-off between the electricity price penalty, transportation cost, and scrap procurement cost to find the most cost-effective scrap list x under different electricity price scenarios. As the electricity price increases, the optimizer should prioritize solutions that slightly reduce energy consumption to mitigate the higher penalty associated with energy cost, thereby working towards minimizing the energy consumption $m(x)$; this assumption is confirmed by the simulation results shown in Figure 6.6.

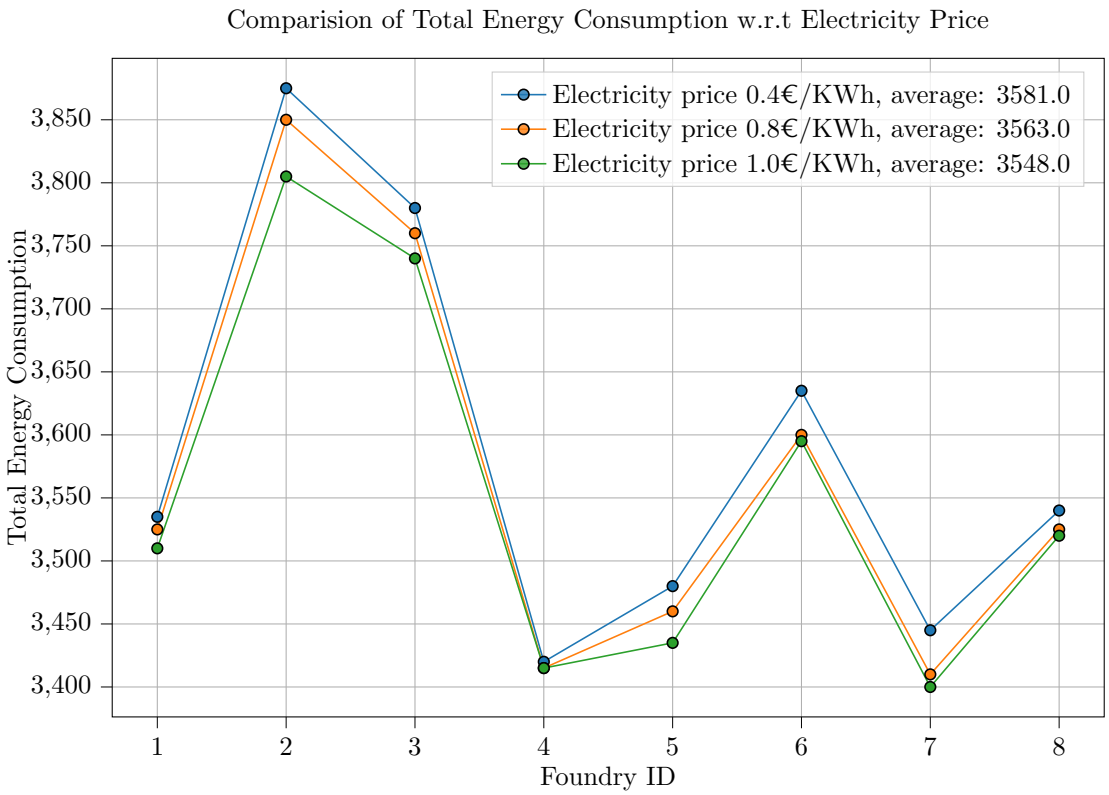

Figure 6.6: Comparison of average energy consumption with respect to the electricity price.

6.3 Summary

Using the optimization methods developed in chapter 5, it is feasible to perform large-scale simulations of the market place platform introduced in chapter 3. In particular, we can investigate the influence of factors like transportation cost and energy prices on the market participants' behaviour, assuming the purchase decisions are made in an optimized way as described in chapter 5.

The simulations indicate that in this case, the energy price indeed influences the purchasing decisions of the foundries. Including the predicted energy costs into the decision process therefore would not only lower the production costs for the foundries, but could also lead to a more sustainable production of steel due to the accordingly reduced energy consumption. Furthermore, the transport costs are shown to have significant influence on the purchasing decision: whereas higher transportation fees might lead to more responsible procurement strategies in terms of avoiding multiple sources of raw materials, it may lead to higher energy consumption due to the optimization trade-off between the energy costs and the transport costs.

Chapter 7

Conclusions

7.1 Thesis summary

This thesis aims to tackle challenges in the foundry industry, focusing on improving scrap procurement and energy consumption through ML and numerical optimization techniques. Chapter 2 begins with an exploration of the foundry and metal recycling sectors, identifying the core challenge of optimizing scrap purchasing. Basic concepts of ML and numerical optimization are introduced as potential solutions to this challenge.

In chapter 3, a platform designed to aid in scrap procurement is presented. This platform is built on ML models and a technique known as surrogate gradient optimization, which are used for simulation purposes. These methods simplify the optimization process, making it easier to find cost-effective scrap purchasing strategies.

We then introduce two additional software tools, EidoData desktop and EidoData web, in chapter 4. These tools are developed to make ML more accessible to individuals in the foundry industry, even without a deep understanding of ML concepts. They provide a user-friendly interface for importing data, selecting ML algorithms, and deploying models, essentially lowering the barrier to entry for ML applications in the industry.

In chapter 5, the aforementioned surrogate gradient methods are first discussed in the very general setting of numerical optimization. Afterwards, their effectiveness in improving the performance for the optimization of data-driven models is demonstrated on two examples: the classical benchmark Rosenbrock function and a real-world dataset from a German foundry.

The effectiveness of the surrogate gradient methods is further demonstrated in chapter 6, where the scrap procurement platform is simulated to investigate the influence of certain cost parameters on the interaction between foundries and scrap suppliers. These simulations would have be unfeasible in larger scales with classical derivative-free optimization techniques. Overall, the simulations show the value of both the general concept of surrogate gradient optimization and the inclusion of process parameters like the energy consumption into the scrap purchase decision process.

7.2 Thesis contributions

This thesis aims to tackle some of the challenges in the foundry industry, particularly focusing on scrap procurement and energy consumption. The work done in this thesis is broken down into three main contributions:

- Developing machine learning software (EidoData desktop and EidoData web): A major part of this thesis is the development of two machine learning software tools named EidoData desktop and EidoData web. These tools are designed to make machine learning more accessible to people in the foundry industry. They provide a simple way for users to bring in their data, choose machine learning algorithms, and use these algorithms to find solutions to their problems. The main goal here is to make it easier for people who lack experience with machine learning to use it in their work.

- Formulation of surrogate gradient optimization methods: Building on the theoretical scaffold of ML and numerical optimization, a novel differentiable surrogate optimization method is formulated. This method plays a crucial role in simplifying the optimization process, thereby making it more tractable and applicable to the foundry industry's scrap procurement challenges. By constructing smooth surrogate functions that approximate the non-smooth cost functions inherent in the optimization landscape, the method significantly enhances the computational efficiency and practicality of deriving optimal scrap purchasing strategies.

- Combining software and algorithms for real-world scrap purchasing optimization simulations: Lastly, this thesis investigates the combination of the developed software tools and the surrogate gradient optimization method to simulate real-world scrap purchasing optimization. This integration offers a practical approach to navigating the complex decision-making process associated with scrap procurement. The developed virtual environment closely resembles the real-world dynamics of scrap trading and presents a powerful tool for decision-makers, enabling a detailed understanding of the complex interactions between scrap procurement, energy consumption, and overall operational cost, and laying the groundwork for more cost-effective and environmentally sustainable practices in the foundry industry.

7.3 Future works

Based on the foundational contribution of this work, several directions for future work present themselves that could significantly build upon and advance these initial achievements:

- For the software EidoData desktop and EidoData web: Firstly, we can further develop a more intuitive GUI with features like drag-and-drop, real-time data visualization, and interactive tutorials that can simplify user interactions. Second, we can incorporate advanced AutoML algorithms to automate the ML process, making it accessible to individuals with limited ML expertise. Third, we can provide options for customizable ML pipelines within the software that can address a diverse range of use cases and user preferences, covering data preprocessing to model deployment. Furthermore, we can enhance cross-platform compatibility that ensures seamless operation across various operating systems and devices, extending the software's usability.

- For the surrogate gradient optimization method: Firstly, we can enhance the algorithm to manage a wider range of optimization problems and explore its potential to adapt to other industrial areas besides foundry scrap procurement, which could broaden its applicability. Secondly, we can delve into the development of more precise or alternative surrogate functions to improve the approximation of non-smooth cost functions, which may lead to better optimization results. Thirdly, we can optimize the method to achieve better scalability and efficiency, which is crucial for handling larger datasets and more

complex optimization landscapes common in real-world scenarios. Lastly, we can develop real-time optimization capabilities to adapt to dynamic changes in problem parameters or the operational environment, a feature that is often needed in industrial settings.

- For the time discrete simulation of scrap procurement: There are multiple other impact factors which have yet to be investigated, such as the number of participating foundries and scrap suppliers and the frequency in which the scrap is re-stocked. The market dynamics could, in particular, be investigated with respect to the stability of the economic process, i.e. it could be determined under which conditions the demand of the foundries will generally be met by the suppliers.

References

[1] A. F. Agarap. "Deep learning using rectified linear units (relu)". *arXiv preprint arXiv:1803.08375* (2018).

[2] S. Agatonovic-Kustrin and R. Beresford. "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research". *Journal of pharmaceutical and biomedical analysis* 22.5 (2000). Pp. 717–727.

[3] A. Agnihotri and N. Batra. "Exploring Bayesian Optimization". *Distill* (2020). https://distill.pub/2020/bayesian-optimization. DOI: 10.23915/distill.00026.

[4] A. Angelopoulos, E. T. Michailidis, N. Nomikos, P. Trakadas, A. Hatziefremidis, S. Voliotis, and T. Zahariadis. "Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects". *Sensors* 20.1 (2019). P. 109.

[5] C. Audet and W. Hare. *Model-based methods in derivative-free nonsmooth optimization*. Springer, 2020.

[6] E. Ayerbe, M. Berecibar, S. Clark, A. A. Franco, and J. Ruhland. "Digitalization of battery manufacturing: current status, challenges, and opportunities". *Advanced Energy Materials* 12.17 (2022). P. 2102696.

[7] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. "A comparison of mechanisms for improving TCP performance over wireless links". *IEEE/ACM transactions on networking* 5.6 (1997). Pp. 756–769.

[8] H. B. Barlow. "Unsupervised learning". *Neural computation* 1.3 (1989). Pp. 295–311.

[9] P. Bennet, D. Langevin, C. Essoual, A. Khaireh-Walieh, O. Teytaud, P. Wiecha, and A. Moreau. "An illustrated tutorial on global optimization in nanophotonics". *arXiv preprint arXiv:2309.09760* (2023).

[10] J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization." *Journal of machine learning research* 13.2 (2012).

[11] P. T. Boggs and J. W. Tolle. "Sequential quadratic programming". *Acta numerica* 4 (1995). Pp. 1–51.

[12] E. Brochu, V. M. Cora, and N. De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". *arXiv preprint arXiv:1012.2599* (2010).

[13] J. Brown. *Foseco ferrous foundryman's handbook*. Butterworth-Heinemann, 2000.

[14] Bvse. *Der Schrottmarkt 2020 - Rückblick auf ein schwieriges Pandemiejahr*. https://www.bvse.de/fachbereiche-schrott-e-schrott-kfz/metallschrott/marktbericht-stahlmetall.html. 2020.

[15] R. Byrd. "Robust trust region methods for constrained optimization". *Third SIAM Conference on Optimization, Houston, Texas*. 1987.

[16] T. Caliński and J. Harabasz. "A dendrite method for cluster analysis". *Communications in Statistics-theory and Methods* 3.1 (1974). Pp. 1–27.

[17] C. Carrión. "Kubernetes scheduling: Taxonomy, ongoing issues and challenges". *ACM Computing Surveys* 55.7 (2022). Pp. 1–37.

[18] R. Caruana and A. Niculescu-Mizil. "An empirical comparison of supervised learning algorithms". *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 161–168.

[19] M. R. Celis. *A trust region strategy for nonlinear equality constrained optimization*. Tech. rep. 1985.

[20] S. Chakraborty, S. Adhikari, and R. Ganguli. "The role of surrogate models in the development of digital twins of dynamic systems". *Applied Mathematical Modelling* 90 (2021). Pp. 662–681.

[21] R. Chandrappa and D. B. Das. *Solid waste management: Principles and practice*. Springer Science & Business Media, 2012.

[22] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. "Deepdriving: Learning affordance for direct perception in autonomous driving". *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2722–2730.

[23] T. Chen and C. Guestrin. "Xgboost: A scalable tree boosting system". *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[24] T. F. Coleman and A. R. Conn. "Nonlinear programming via an exact penalty function: Asymptotic analysis". *Mathematical programming* 24.1 (1982). Pp. 123–136.

[25] T. F. Coleman and A. R. Conn. "Nonlinear programming via an exact penalty function: Global analysis". *Mathematical Programming* 24.1 (1982). Pp. 137–161.

[26] H. Colpaert. *Metallography of steels: interpretation of structure and the effects of processing*. Asm International, 2018.

[27] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.

[28] R. Courant. "Variational methods for the solution of problems of equilibrium and vibrations" (1943).

[29] S. Dalquist and T. Gutowski. "Life cycle analysis of conventional manufacturing techniques: sand casting". *ASME International mechanical engineering congress and exposition*. Vol. 47136. 2004, pp. 631–641.

[30] L. Dartnell. *The knowledge: How to rebuild our world from scratch*. Random House, 2014.

[31] Django Software Foundation. *Django*. Version 2.2. URL: `%5Curl % 7Bhttps : / / djangoproject.com%7D` (visited on 10/08/2022).

[32] P. Domingos. "A few useful things to know about machine learning". *Communications of the ACM* 55.10 (2012). Pp. 78–87.

[33] V. Dunjko and H. J. Briegel. "Machine learning & artificial intelligence in the quantum domain: a review of recent progress". *Reports on Progress in Physics* 81.7 (2018). P. 074001.

[34] G. Dutta and R. Fourer. "A survey of mathematical programming applications in integrated steel plants". *Manufacturing & Service Operations Management* 3.4 (2001). Pp. 387–400.

[35] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola. "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data". *arXiv preprint arXiv:2003.06505* (2020).

[36] H. J. Escalante, M. Montes, and L. E. Sucar. "Particle swarm model selection." *Journal of Machine Learning Research* 10.2 (2009).

[37] European IPPC Bureau. *Best Available Techniques (BAT) Reference Document for the Smitheries and Foundries Industry*. URL: %5Curl%7Bhttps://eippcb.jrc.ec.europa.eu/sites/default/files/2022-02/SF_BREF_D1_web.pdf%7D.

[38] R. Fakoor, J. W. Mueller, N. Erickson, P. Chaudhari, and A. J. Smola. "Fast, Accurate, and Simple Models for Tabular Data via Augmented Distillation". *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 8671–8681. URL: https://proceedings.neurips.cc/paper/2020/file/62d75fb2e3075506e8837d8f55021ab1-Paper.pdf.

[39] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter. "Auto-sklearn 2.0: Hands-free automl via meta-learning". *arXiv preprint arXiv:2007.04074* (2020).

[40] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. "Efficient and robust automated machine learning". *Advances in neural information processing systems* 28 (2015).

[41] R. Fletcher. "Second order corrections for non-differentiable optimization". *Numerical Analysis: Proceedings of the 9th Biennial Conference Held at Dundee, Scotland, June 23–26, 1981*. Springer. 2006, pp. 85–114.

[42] R. Fletcher, N. I. Gould, S. Leyffer, P. L. Toint, and A. Wächter. "Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming". *SIAM Journal on Optimization* 13.3 (2002). Pp. 635–659.

[43] flower dev doc. *Flower*. URL: %5Curl%7Bhttps://flower.readthedocs.io/en/latest/%7D.

[44] A. I. Forrester, N. W. Bressloff, and A. J. Keane. "Optimization using surrogate models and partially converged computational fluid dynamics simulations". *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 462.2071 (2006). Pp. 2177–2204.

[45] M. Fowler. *Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch*. Addison-Wesley, 2012.

[46] P. I. Frazier and S. C. Clark. "Parallel global optimization using an improved multi-points expected improvement criterion". *INFORMS Optimization Society Conference, Miami FL*. Vol. 26. 2012.

[47] J. H. Friedman. "Greedy function approximation: a gradient boosting machine". *Annals of statistics* (2001). Pp. 1189–1232.

[48] M. Fukushima. "A successive quadratic programming algorithm with global and superlinear convergence properties". *Mathematical Programming* 35.3 (1986). Pp. 253–264.

[49] P. E. Gill, W. Murray, and M. A. Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization". *SIAM review* 47.1 (2005). Pp. 99–131.

[50] P. E. Gill and E. Wong. "Sequential quadratic programming methods". *Mixed integer nonlinear programming*. Springer, 2011, pp. 147–224.

[51] Gitlab. *What is CI/CD?* `https://docs.gitlab.com/ee/ci/introduction/img/gitlab_workflow_example_11_9.png`. 2022. (Visited on 12/07/2022).

[52] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. First. MIT press, 2016.

[53] J. Goodman and J. Weare. "Ensemble samplers with affine invariance". *Communications in applied mathematics and computational science* 5.1 (2010). Pp. 65–80.

[54] G. Goodwin, M. M. Seron, and J. A. De Doná. *Constrained control and estimation: an optimisation approach*. Springer Science & Business Media, 2006.

[55] D. Gorissen, T. Dhaene, and F. De Turck. "Evolutionary model type selection for global surrogate modeling". *Journal of Machine Learning Research* 10 (2009). Pp. 2039–2078.

[56] M. G. Grant, K. C. Kaiser, S. M. Cantacuzene, and T. Chen. "Optimization of Oxygen Steelmaking in Nonconventional EAF Operations". *AISTECH-CONFERENCE PROCEEDINGS-*. Vol. 1. Citeseer. 2005, p. 545.

[57] H. Greenspan, B. Van Ginneken, and R. M. Summers. "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique". *IEEE transactions on medical imaging* 35.5 (2016). Pp. 1153–1159.

[58] T. P. G. D. Group. *Documentation PostgreSQL 10.3*. Ed. by T. P. G. D. Group. 2018.

[59] D. R. Gunasegaram, A. Murphy, A. Barnard, T. DebRoy, M. Matthews, L. Ladani, and D. Gu. "Towards developing multiscale-multiphysics models and their surrogates for digital twins of metal additive manufacturing". *Additive Manufacturing* 46 (2021). P. 102089.

[60] I. Guyon et al. "Analysis of the AutoML challenge series". *Automated Machine Learning* (2019). P. 177.

[61] S. .-. Han and O. L. Mangasarian. "Exact penalty functions in nonlinear programming". *Mathematical programming* 17 (1979). Pp. 251–269.

[62] S.-P. Han. "Superlinearly convergent variable metric algorithms for general nonlinear programming problems". *Mathematical Programming* 11.1 (1976). Pp. 263–282.

[63] S.-P. Han. "A globally convergent method for nonlinear programming". *Journal of optimization theory and applications* 22.3 (1977). Pp. 297–309.

[64] M. R. Hestenes. "Multiplier and gradient methods". *Journal of optimization theory and applications* 4.5 (1969). Pp. 303–320.

[65] F. Hutter, T. Stützle, K. Leyton-Brown, and H. H. Hoos. "ParamILS: an automatic algorithm configuration framework". *arXiv e-prints* (2014). arXiv–1401.

[66] I. Ilankoon, Y. Ghorbani, M. N. Chong, G. Herath, T. Moyo, and J. Petersen. "E-waste in the international context–A review of trade flows, regulations, hazards, waste management strategies and technologies for value recovery". *Waste management* 82 (2018). Pp. 258–275.

[67] D. Inc. *Package Software into Standardized Units for Development, Shipment and Deployment*. `https://www.docker.com/resources/what-container/`. 2022.

[68] U. D. the Interior. *Mineral Commodity Summaries, January 2021: Iron and Steel Scrap*. `https://pubs.usgs.gov/periodicals/mcs2021/mcs2021-iron-steel-scrap.pdf`. 2021.

[69] Introduction to Celery. *Celery*. URL: %5Curl%7Bhttps://docs.celeryq.dev/en/stable/getting-started/introduction.html%7D (visited on 11/08/2022).

[70] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[71] R. Jin and G. Agrawal. "Communication and memory efficient parallel decision tree construction". *Proceedings of the 2003 SIAM international conference on data mining*. SIAM. 2003, pp. 119–129.

[72] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, perspectives, and prospects". *Science* 349.6245 (2015). Pp. 255–260.

[73] G. Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". *Advances in neural information processing systems* 30 (2017).

[74] F. Kheiri. "A review on optimization methods applied in energy-efficient building geometry and envelope design". *Renewable and Sustainable Energy Reviews* 92 (2018). Pp. 897–920.

[75] V. Klee and G. J. Minty. "How good is the simplex algorithm". *Inequalities* 3.3 (1972). Pp. 159–175.

[76] J. Kober, J. A. Bagnell, and J. Peters. "Reinforcement learning in robotics: A survey". *The International Journal of Robotics Research* 32.11 (2013). Pp. 1238–1274.

[77] D. Kraft. "A software package for sequential quadratic programming". *Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt* (1988).

[78] O. Kramer, D. E. Ciaurri, and S. Koziel. "Derivative-free optimization". *Computational optimization, methods and algorithms*. Springer, 2011, pp. 61–83.

[79] M. Kuss, C. E. Rasmussen, and R. Herbrich. "Assessing Approximate Inference for Binary Gaussian Process Classification." *Journal of machine learning research* 6.10 (2005).

[80] J. Larson, M. Menickelly, and S. M. Wild. "Derivative-free optimization methods". *Acta Numerica* 28 (2019). Pp. 287–404.

[81] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. "Efficient backprop". *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.

[82] P. Li, Q. Wu, and C. Burges. "Mcrank: Learning to rank using multiple classification and gradient boosting". *Advances in neural information processing systems* 20 (2007).

[83] M. Lieberwirth and H. Hobbie. "Decarbonizing the industry sector and its effect on electricity transmission grid operation—Implications from a model based analysis for Germany". *Journal of Cleaner Production* 402 (2023). P. 136757.

[84] P.-L. Liu and A. Der Kiureghian. "Optimization algorithms for structural reliability". *Structural safety* 9.3 (1991). Pp. 161–177.

[85] I. Loshchilov. "LM-CMA: An alternative to L-BFGS for large-scale black box optimization". *Evolutionary computation* 25.1 (2017). Pp. 143–171.

[86] S. M. Lundberg and S.-I. Lee. "A unified approach to interpreting model predictions". *Advances in neural information processing systems* 30 (2017).

[87] R. D. MacRosty and C. L. Swartz. "Dynamic modeling of an industrial electric arc furnace". *Industrial & engineering chemistry research* 44.21 (2005). Pp. 8067–8083.

[88] N. Maratos. "Exact penalty function algorithms for finite dimensional and control optimization problems" (1978).

[89] J. R. Martins and A. Ning. *Engineering design optimization*. Cambridge University Press, 2021.

[90] D. Q. Mayne. "On the use of exact penalty functions to determine step length in optimization algorithms". *Numerical Analysis: Proceedings of the 8th Biennial Conference Held at Dundee, Scotland, June 26–29, 1979*. Springer. 2006, pp. 98–109.

[91] D. Q. Mayne and E. Polak. *A surperlinearly convergent algorithm for constrained optimization problems*. Springer, 1982.

[92] S. Mehrotra. "On the implementation of a primal-dual interior point method". *SIAM Journal on optimization* 2.4 (1992). Pp. 575–601.

[93] W. Metalle. *GEMEINSAM AUFBRECHEN Der Geschäftsbericht der Nichteisen-Metallindustrie*. `https://www.wvmetalle.de/index.php?eID=dumpFile&t=f&f=282762&token=b83d45f6a5de8ba60c9bfde1e17e91a0851da87a`. 2022.

[94] L. Meunier et al. "Black-box optimization revisited: Improving algorithm selection wizards through massive benchmarking". *IEEE Transactions on Evolutionary Computation* 26.3 (2021). Pp. 490–500.

[95] microsoft. *Clean Architecture*. `https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures`. 2022. (Visited on 07/07/2022).

[96] I. Miletic, R. Garbaty, S. Waterfall, and M. Mathewson. "Model-based optimization of scrap steel purchasing". *IFAC Proceedings Volumes* 40.11 (2007). Pp. 263–266.

[97] I. Miletic, R. Garbaty, S. Waterfall, and M. Mathewson. *Steel scrap purchasing optimization and supply management*. American Institute of Chemical Engineers, 2006.

[98] J. Mockus. "Application of Bayesian approach to numerical methods of global and stochastic optimization". *Journal of Global Optimization* 4.4 (1994). Pp. 347–365.

[99] M. Moustapha and B. Sudret. "Surrogate-assisted reliability-based design optimization: a survey and a unified modular framework". *Structural and Multidisciplinary Optimization* 60 (2019). Pp. 2157–2176.

[100] J. P. Mueller and L. Massaron. *Machine learning for dummies*. Third. New York: John Wiley & Sons, 2021.

[101] P. Mullinger and B. Jenkins. *Industrial and process furnaces: principles, design and operation*. Butterworth-Heinemann, 2022.

[102] N. Nagesha and P. Balachandra. "Barriers to energy efficiency in small industry clusters: Multi-criteria-based prioritization using the analytic hierarchy process". *Energy* 31.12 (2006). Pp. 1969–1983.

[103] A. T. Nakhjiri, H. Sanaeepur, A. E. Amooghin, and M. M. A. Shirazi. "Recovery of precious metals from industrial wastewater towards resource recovery and environmental sustainability: A critical review". *Desalination* 527 (2022). P. 115510.

[104] J. Neumann, M. Petranikova, M. Meeus, J. D. Gamarra, R. Younesi, M. Winter, and S. Nowak. "Recycling of lithium-ion batteries—current state of the art, circular economy, and next generation recycling". *Advanced energy materials* 12.17 (2022). P. 2102917.

[105] A. Ng. *How large do the dev/test sets need to be?* Tech. rep. ... DeepLearning AI, 2018.

[106] A.-T. Nguyen, S. Reiter, and P. Rigo. "A review on simulation-based optimization methods applied to building performance analysis". *Applied energy* 113 (2014). Pp. 1043–1058.

[107] M. A. Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, USA, 2015.

[108] L. Niu and Y. Yuan. "NEW TRUST-REGION ALGORITHM FOR NONLINEAR CONSTRAINED OPTIMIZATION". *Journal of Computational Mathematics* (2010). Pp. 72–86.

[109] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.

[110] a. Nordic. *Docker architecture*. `https://nordicapis.com/`. 2022.

[111] H. Oettel. "Ground, Polished and Viewed in Close-Up: the First Six Decades of 'Schumann'1". *Practical Metallography* 50.9 (2013). Pp. 588–606.

[112] E. O. Omojokun. *Trust region algorithms for optimization with nonlinear equality and inequality constraints*. University of Colorado at Boulder, 1989.

[113] U. G. Palomares and O. L. Mangasarian. "Superlinearly convergent quasi-Newton algorithms for nonlinearly constrained optimization problems". *Mathematical Programming* 11.1 (1976). Pp. 1–13.

[114] F. Pedregosa et al. "Scikit-learn: Machine learning in Python". *the Journal of machine Learning research* 12 (2011). Pp. 2825–2830.

[115] I. Polmear. *Light alloys: from traditional alloys to nanocrystals*. Elsevier, 2005.

[116] I. Polmear, D. StJohn, J.-F. Nie, and M. Qian. *Light alloys: metallurgy of the light metals*. Butterworth-Heinemann, 2017.

[117] M. J. Powell. "A method for nonlinear constraints in minimization problems". *Optimization* (1969). Pp. 283–298.

[118] M. J. Powell. "Algorithms for nonlinear constraints that use Lagrangian functions". *Mathematical programming* 14 (1978). Pp. 224–248.

[119] M. J. Powell. "The convergence of variable metric methods for nonlinearly constrained optimization calculations". *Nonlinear programming 3*. Elsevier, 1978, pp. 27–63.

[120] M. J. Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[121] M. J. Powell. "A fast algorithm for nonlinearly constrained optimization calculations". *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*. Springer. 2006, pp. 144–157.

[122] M. Powell and Y.-X. Yuan. "A trust region algorithm for equality constrained optimization". *Math. Program.* 49.1 (1991). Pp. 189–211.

[123] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. "CatBoost: unbiased boosting with categorical features". *Advances in neural information processing systems* 31 (2018).

[124] B. Pshenichnyi. "Algorithms for general mathematical programming problems". *Cybernetics* 6.5 (1970). Pp. 677–684.

[125] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker. "Surrogate-based analysis and optimization". *Progress in aerospace sciences* 41.1 (2005). Pp. 1–28.

[126] T. M. Ragonneau. "Model-Based Derivative-Free Optimization Methods and Software". *arXiv preprint arXiv:2210.12018* (2022).

[127] S. Ranka and V. Singh. "CLOUDS: A decision tree classifier for large datasets". *Proceedings of the 4th knowledge discovery and data mining conference*. Vol. 2. 8. 1998.

[128] J. Rapin and O. Teytaud. *Nevergrad - A gradient-free optimization platform*. `https://GitHub.com/FacebookResearch/Nevergrad`. 2018.

[129] L. Rathba. *Model Fitting for an Electric Arc Furnace*. 2004.

[130] redhat. *Continuous Integration: A "Typical" Process*. `https://developers.redhat.com/blog/2017/09/06/continuous-integration-a-typical-process`. 2022. (Visited on 12/07/2022).

[131] Redis Enterprise Software technical overview. *Redis*. URL: `%5Curl%7Bhttps://docs.redis.com/latest/rs/technology-behind-redis-enterprise/%7D` (visited on 11/08/2022).

[132] J. R. Rice. "The algorithm selection problem". *Advances in computers*. Vol. 15. Elsevier, 1976, pp. 65–118.

[133] S. M. Robinson. "A quadratically-convergent algorithm for general nonlinear programming problems". *Mathematical programming* 3 (1972). Pp. 145–156.

[134] S. M. Robinson. "Extension of Newton's method to nonlinear functions with values in a cone". *Numerische Mathematik* 19.4 (1972). Pp. 341–347.

[135] R. T. Rockafellar. "A dual approach to solving nonlinear programming problems by unconstrained optimization". *Mathematical programming* 5.1 (1973). Pp. 354–373.

[136] H. Rosenbrock. "An automatic method for finding the greatest or least value of a function". *The computer journal* 3.3 (1960). Pp. 175–184.

[137] P. J. Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". *Journal of computational and applied mathematics* 20 (1987). Pp. 53–65.

[138] E. Sandberg. "Energy and scrap optimisation of electric arc furnaces by statistical analysis of process data". PhD thesis. Luleå tekniska universitet, 2005.

[139] I. Santos, J. Nieves, P. Bringas, and Y. Penya. "Machine-learning-based defect prediction in highprecision foundry production". *Structural Steel and Castings: Shapes and Standards, Properties and Applications* (2010). Pp. 259–276.

[140] K. Schittkowski. "NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems". *Annals of operations research* 5 (1986). Pp. 485–500.

[141] B. Schölkopf, A. Smola, and K.-R. Müller. "Kernel principal component analysis". *International conference on artificial neural networks*. Springer. 1997, pp. 583–588.

[142] T. Simpson, V. Toropov, V. Balabanov, and F. Viana. "Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come-or not". *12th AIAA/ISSMO multidisciplinary analysis and optimization conference*. 2008, p. 5802.

[143] K. A. Smith-Miles. "Cross-disciplinary perspectives on meta-learning for algorithm selection". *ACM Computing Surveys (CSUR)* 41.1 (2009). Pp. 1–25.

[144] A. F. Society. *Castings in Our World Factsheet*. `https://afsinc.s3.amazonaws.com/Documents/Marketing/Castings_Enduse.pdf`. 2020.

[145] D. Solomatine, L. M. See, and R. Abrahart. "Data-driven modelling: concepts, approaches and experiences". *Practical hydroinformatics: Computational intelligence and technological developments in water applications* (2008). Pp. 17–30.

[146] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors". *Proceedings of the 2Nd ACM SIGOPS/EuroSys european conference on computer systems 2007*. 2007, pp. 275–287.

[147] B. Srinivasan, L. T. Biegler, and D. Bonvin. "Tracking the necessary conditions of optimality with changing set of active constraints using a barrier-penalty function". *Computers & Chemical Engineering* 32.3 (2008). Pp. 572–579.

[148] W. Stahl. *Rohstahlproduktion in Deutschland*. `https://www.stahl-online.de/medieninformationen/rohstahlproduktion-in-deutschland-maerz-2022/`. 2022.

[149] H. A. Sturges. "The choice of a class interval". *Journal of the american statistical association* 21.153 (1926). Pp. 65–66.

[150] Q. Sun, B. Pfahringer, and M. Mayo. "Full model selection in the space of data mining operators". *Proceedings of the 14th annual conference companion on genetic and evolutionary computation*. 2012, pp. 1503–1504.

[151] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms". *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 847–855.

[152] M. E. Tipping and C. M. Bishop. "Mixtures of probabilistic principal component analyzers". *Neural computation* 11.2 (1999). Pp. 443–482.

[153] Y. N. Toulouevski and I. Y. Zinurov. "Innovation in electric arc furnaces". *Holland Landing, Canada* 2 (2010).

[154] P. Tüfekci. "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods". *International Journal of Electrical Power & Energy Systems* 60 (2014). Pp. 126–140.

[155] T. C. Uyan, K. Otto, M. S. Silva, P. Vilaça, and E. Armakan. "Industry 4.0 foundry data management and supervised machine learning in low-pressure die casting quality improvement". *International Journal of Metalcasting* 17.1 (2023). Pp. 414–429.

[156] A. Vardi. "A trust region algorithm for equality constrained minimization: convergence properties and implementation". *SIAM Journal on Numerical Analysis* 22.3 (1985). Pp. 575–591.

[157] J. Wang, Y. Zhang, K. Cui, T. Fu, J. Gao, S. Hussain, and T. S. AlGarni. "Pyrometallurgical recovery of zinc and valuable metals from electric arc furnace dust–a review". *Journal of Cleaner Production* 298 (2021). P. 126788.

[158] X. Wang and Y. Yuan. "An augmented Lagrangian trust region method for equality constrained optimization". *Optimization Methods and Software* 30.3 (2015). Pp. 559–582.

[159] R. B. Wilson. "A simplicial algorithm for concave programming". *Ph. D. Dissertation, Graduate School of Bussiness Administration* (1963).

[160] WorldSteelAssociation. *2021 World Steel in Figures*. `https://worldsteel.org/wp-content/uploads/2021-World-Steel-in-Figures.pdf`. 2021.

[161] D. Xin, E. Y. Wu, D. J.-L. Lee, N. Salehi, and A. Parameswaran. "Whither automl? understanding the role of automation in machine learning workflows". *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–16.

[162] xteam. *INTRODUCTION TO KUBERNETES ARCHITECTURE*. `https://x-team.com/blog/introduction-kubernetes-architecture/`. 2022.

[163] S. Zhang, C. Zhang, and Q. Yang. "Data preparation for data mining". *Applied artificial intelligence* 17.5-6 (2003). Pp. 375–381.

[164] B. Zhou, T. Pychynski, M. Reischl, E. Kharlamov, and R. Mikut. "Machine learning with domain knowledge for predictive quality monitoring in resistance spot welding". *Journal of Intelligent Manufacturing* 33.4 (2022). Pp. 1139–1163.

[165] Z.-H. Zhou. *Machine learning*. Springer Nature, 2021.

[166] X. J. Zhu. "Semi-supervised learning literature survey" (2005).

[167] M. Zinkevich, M. Weimer, L. Li, and A. Smola. "Parallelized stochastic gradient descent". *Advances in neural information processing systems* 23 (2010).

[168] M.-A. Zöller and M. F. Huber. "Benchmark and survey of automated machine learning frameworks". *Journal of artificial intelligence research* 70 (2021). Pp. 409–472.