University of the Witwatersrand

School of Chemical and Metallurgical Engineering

UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

A Data-driven Soft Sensor for Predicting Grade and Recovery in a Platinum Concentrator

MSc Research Dissertation

Prepared by

Tebogo Motsa

Student number: 2105985

Submitted to

School of Chemical and Metallurgical Engineering, Faculty of Engineering and Built Environment, University of the Witwatersrand, Johannesburg, South Africa

Supervisor: Prof. Antony Higginson

Co-Supervisor: Prof. Kevin Brooks

I _____Tebogo Teeno Motsa_____ (Student number: __2105985_____) am a student registered for __MSc Chemical Engineering_____ in the year___2023___. I hereby declare the following:

- I am aware that plagiarism (the use of someone else's work without their permission and/or without acknowledging the original source) is wrong.
- I confirm that the work submitted for assessment for the above course is my own unaided work except where I have explicitly indicated otherwise.
- I have followed the required conventions in referencing the thoughts and ideas of others.
- I understand that the University of the Witwatersrand may take disciplinary action against me if there is a belief that this in not my own unaided work or that I have failed to acknowledge the source of the ideas or words in my writing.

Signature: _____T.Motsa_____ Date: _____24/03/2025_____

Abstract

The timely and accurate prediction of the grade and recovery in flotation circuits is critical for optimising metallurgical operations, where even a 1% increase in throughput can yield significant profit improvements. Traditional laboratory analyses often cause significant delays that limit the ability to adapt to the rapid and dynamic changes inherent in these processes. This study investigated the feasibility of a data-driven soft sensor that uses machine learning to infer the grade and recovery in real time. A comprehensive dataset collected from the Anglo-American archive, consisting of key flotation process variables, was pre-processed using robust feature selection techniques. This ensured that only the most influential parameters, such as the reagent concentrations, sump conditions, and cell-level measurements, were retained. The performance of various machine learning models, linear regression, neural networks, random forest and extreme gradient boosting was evaluated to predict the grade of platinum group metals (PGMs) in a flotation circuit. Although linear regression provides a useful baseline understanding of the process, its poor performance reinforces the inherent nonlinearity of the process. Among nonlinear models, Random Forest achieved the best generalisation performance, demonstrating robust predictive accuracy on unseen data. Extreme gradient boosting (XGBoost), while exceptionally well capturing complex dynamics on the training set with a coefficient of determination (R2) near 99%, exhibited signs of overfitting, owing to its lower performance on the unseen data. Neural networks have also demonstrated strong generalisation underscoring the potential of deep learning in this application. Moreover, hyperparameter tuning is essential for optimising non-linear models for optimum performance. Random forest is a reliable model for the development of data-driven soft sensors to improve real-time process control in flotation circuits. Real-time predictions were achieved with minimal computational overhead, offering a viable alternative to the conventional time-consuming laboratory methods. This integrated soft sensor not only enhances process control but also provides operators and metallurgists with actionable insights that ultimately contribute to more efficient and responsive flotation operations.

Keywords: Soft Sensor, Flotation Circuit, Machine Learning, Real-Time Prediction, Artificial Neural Network, Random Forest, Extreme Gradient Boosting, Grade and Recovery, Feature Selection, Hyperparameter Tuning.

Acknowledgements

I express my sincere gratitude to everyone who has played a role in shaping my journey, both personally and academically. First and foremost, I thank Almighty God for His unwavering love, mercy, and guidance, and I am deeply grateful for the wisdom and humility imparted to me by the Holy Spirit.

A special thanks to Anglo American for fully funding this project and providing the necessary data for robust analysis and evaluation. I owe a deep debt of gratitude to my supervisors, Honorary Adjunct Professor Kevin Brooks and Professor Antony Higginson, as well as the Anglo-American team: Chris Steyn (Principal Advanced Control Engineer), Kudzai Changunda (Group Principal Flotation), Angus Morrison (Group Principal Flotation), Johan Steyn (Principal Platinum), Daniel Rabopane (Lead Engineer – APC), and De Villiers Groenewald (Lead Engineer). It was an honour and a privilege to work with you. Your expert guidance, patience, and unwavering encouragement have been invaluable throughout this research. Your mentorship has profoundly shaped my academic and professional growth, and I am committed to making you proud in all my future endeavours

I extend my heartfelt thanks to my beloved mother Lindiwe Mamba, and my grandmother Gcinaphi Mamba for their endless support, love, and prayers. Your constant encouragement made it possible for me to pursue and complete this master's program. I am also grateful to my entire family for their steadfast support and kindness. I also recognise and appreciate the support and encouragement I got from my friends. Finally, I would like to thank lecturers and support staff at Wits University, especially the late Prof. Josias Van Der Merwe and Mr. Paul Denhoed for their support and cooperation. Lastly a special thanks to the Innovative Process Solutions (IPS) family your contributions have been crucial in helping me complete my master's degree. Thank you all.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

The mining industry is a cornerstone of the global economy, providing essential materials for various sectors. Flotation circuits are among the most critical processes in the mining industry serving a pivotal role in mineral processing. They ensure that the valuable minerals are separated efficiently from the ore. The efficiency of this separation process is measured in terms of grade and recovery rates which are essential indicators of a circuit's performance. Engineers use these metrics to assess process performance and ensure efficiency, as they directly reflect yield. Many industries have invested heavily in advanced grade and recovery monitoring technologies, recognizing that even a 1% improvement in yield can translate into substantial profits. Traditionally, the information of ore grade and recovery in a plant relied on empirical models and operator experience. However, these methods often fall short in handling flotation circuit's complex and dynamic nature. The emergence of Industry 4.0 has led to a significant shift towards data-driven approaches, leveraging the power of machine learning and artificial intelligence to enhance predictive accuracy and operational efficiency.

The worldwide need for platinum-group metals (PGMs) commonly known as 4E, which involves platinum (Pt), palladium (Pd), rhodium (Rh), and gold (Au) greatly surpasses the availability of this minerals (Wilburn & Bleiwas, 2004). This supply-demand imbalance coupled with expanding industrial and technological applications of PGMs has driven intensified exploration efforts and spurred advancements in resource detection and deposit assessment methodologies. The limited and uneven geographical spread of these resources and elevated production expenses highlight the challenge of meeting the growing demand for PGMs. Anglo-American, situated in South Africa, is ranked as the leading producer of PGMs, and South Africa is the greatest producer of these metals globally, with reserves estimated to be approximately 65 million kilograms (Wilburn & Bleiwas, 2004). Flotation is crucial in ore processing industries, facilitating high-grade concentrate and high recovery.

Globally, surface mineral reserves have been declining, prompting mining companies to invest heavily in accessing deeper deposits (Ali et al., 2017). Newly discovered mineral resources are typically of lower grade, complicating the extraction process and making it more challenging to exploit these resources efficiently and economically. Additionally, the mining industry faces significant data utilization challenges, as only 10% of available data is leveraged to manage variability and optimize operations due to its disconnection from the value chain (Ali *et al.*,

2017). Mineral processing and metal extraction are also major contributors to greenhouse gas emissions, which has heightened the focus on sustainable and ethical mining practices aimed at reducing the environmental and material impact across the entire value chain (Ali et al., 2017). As the industry seeks economically viable minerals at greater depths, traditional methods are proving inadequate for efficiently extracting smaller ore bodies, making the process increasingly complex and time-consuming.

For centuries, the utilisation of froth flotation as a physicochemical approach for particle separation has remained among the most extensively employed methods around mineral processing, recycling, and water treatment (Shean & Cilliers, 2011). Flotation cells exhibit dynamic behaviours primarily distinguished by robust interconnections, time delays, and random disturbances (Eduardo *et al.*, 1995). The milling and flotation circuit is complex, with many variables from different process units affecting it to various extents. These interactions between variables make control and monitoring of the process more challenging. Optimising this process holds a key position in enhancing the efficiency of the plant due to increased throughput, therefore even slight improvements in the recovery of the valuable metals yield substantial economic benefits (Maldonado, Sbarbaro and Lizama, 2007).This aligns with the new Anglo-American strategy outlined in the Anglo-American Platinum Limited Integrated Report of 2018, where the mine reported significant improvements in PGM yields. The following figure provides Anglo-American summary for the adopted strategies for 2018 and 2019 financial year.

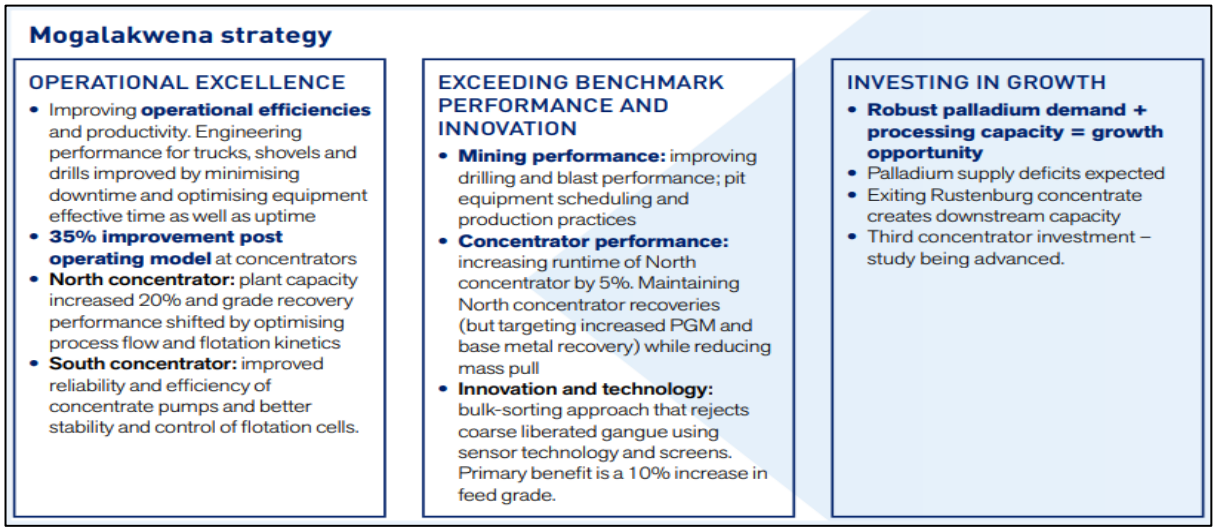


Figure 1: *Anglo American Mogalakwena Mine 2018 strategy summary (Ndlovu, 2021)*

The Mogalakwena operation prioritizes operational excellence, innovation, and sustainable growth, as reflected in its strategic framework (Figure 1). Recent advancements in concentrator

performance including significant improvements in operational models, capacity expansion, and enhanced reliability across plants underscore the importance of adopting predictive technologies to further optimize metallurgical processes. A soft sensor as a data-driven tool will assist address these needs by enabling real-time process insights, reducing variability, and improving flotation efficiency. Such innovations align with broader objectives of maximizing mineral recovery, reducing operational costs, and ensuring scalable production capabilities, particularly in response to growing global demand for critical metals like palladium. The soft sensor integration of historical and real-time data analytics approach supports Mogalakwena's commitment to operational agility and strategic growth in a competitive resource landscape.

Soft sensing refers to the different approaches and algorithms utilised to make informed predictions or estimates in industrial processes regarding product quality or physical quantities based on knowledge and available measurements (Fortuna, et al., 2007). The development of a soft sensor, a virtual instrument that estimates process variables not directly measurable, represents a leap forward in this domain. Utilizing real-time data and advanced analytics, soft sensors can provide timely and accurate predictions of grade and recovery, enabling operators to make informed decisions and optimize the flotation process. One of the key challenges in flotation cell operation is accurately determining the grade and recovery of the minerals (Jera & Bhondayi, 2021). The grade is the percentage of the valuable mineral in the concentrate, while the recovery is the percentage of the valuable mineral that is recovered from the feed ore. Furthermore, (Singh & Mohan Rao, 2005) emphasised that these inaccurate measurements can lead to sub-optimal process control, reduced metal yield, and increased operational costs. There is no doubt that accurate estimation of grade and recovery in platinum flotation processes is crucial for optimising the extraction of valuable 4E (Platinum Group Elements) metals.

Data processing in industry historical data is lacking and the most challenging, frequently surpassing the complexity of designing predictive models, yet it remains essential for constructing an effective predictive model, especially for flotation circuits given their dynamic behaviour. Traditional methods that are currently employed in industries to carry out this task involve physical sensors, but they have limitations (Jiang *et al.*, 2021). These conventional methods require manual sampling and laboratory analysis to obtain the hard-to-measure variables of the process which results in time delays and limited real-time process insight. Therefore, there is a need for developing alternative methods that can provide fast, accurate, and continuous estimation of the grade and recovery of a flotation circuit.

The development of a data-driven soft sensor for online grade and recovery inference, using advanced techniques holds significant promise for improving process efficiency, optimising resource utilisation, and enhancing overall profitability in platinum mining operations (McCoy & Auret, 2019). The proposed sensor will not only predict grade and recovery with high precision but also adapt to changing conditions within the flotation circuit, ensuring consistent performance. Data-driven soft sensors do not require physical sensors or detailed knowledge of the process dynamics. They can be built using various machine-learning techniques, such as artificial neural networks, support vector machines, decision trees, fuzzy logic, or genetic algorithms. The substantial volume of data being collected and retained within the process industries serves as a motivation in the data-driven soft sensor development along with its advancement (Pani & Mohanta, 2011). These types of sensors utilise statistical models which are trained using historical data from the local processes usually obtained from the physical sensors located in different parts of the plant.

Historical data is very important because these statistical models get to search for patterns and correlations and capture the nonlinear system dynamics between the input and the target output of the plant. These in turn assist the soft sensor to perform well in both novel and normal process conditions. Once trained and validated, data-driven soft sensors can provide real-time estimation of the grade and recovery of a flotation circuit based on the current and historical input data. Consequently, soft sensors are becoming a valuable tool for many applications in industrial fields such as nuclear plants, chemical plants, pulp and pepper industry, food processing, etc.

In the past decade, the concept of soft sensing technology has gained significant attention and momentum in academic circles and various process sectors, emerging as a highly promising approach for real-time forecasting (Jin *et al.,* 2016). On the other hand, data-based software sensors allow for real-time process monitoring and control, which is highly advantageous for enhancing flotation performance. This approach can effectively capture the evolving states of the system and generate a prediction by analysing both past and current process information. Additionally, it has the capability to swiftly identify anomalies in both instrumentation and the overall process, thereby guaranteeing optimal process operation. According to Jiang et al. (2021), these advancements in process control and monitoring are facilitated by the rapid growth and evolution of data science, computing and communication technology, statistical tools, and machine learning. Previous research indicates that integrating froth imaging with empirical and physical modelling can serve as the foundation for such a soft sensor (Jahedsaravani et al., 2023).

Numerous studies conducted over the past ten years have demonstrated that the froth's appearance carries a wealth of useful information that is essential to improving the flotation process efficiency (Nakhaei *et al.,* 2022). Hence, several soft sensors were developed to measure the froth visual and textural characteristics in flotation plants and are producing positive results (Jahedsaravani et al., 2023). The author provided additional insights, highlighting that these advanced sensors leverage machine vision learning algorithms to effectively process data, thereby enhancing their capability to model and predict flotation performance. This cutting-edge integration of sensor technology and predictive analytics represents a noteworthy development in the domain of flotation processes. The evolution of froth imaging and data-driven soft sensors is ongoing (Zheng, et al., 2023).

The potential for further optimisation is vast through integration with other advanced technologies such as big data analytics, the Internet of Things (IoT), and remote sensing, as indicated by Jahedsaravani et al, (2023). This promising convergence of technologies opens new horizons for improved efficiency and insight in various applications. Convolutional Neural Networks (CNNs) have shown great performance under deep neural networks in efficiently handling image processing tasks, including applications in the field of froth flotation studies (Jahedsaravani et al., 2023). Deep learning has a natural ability to uncover features and patterns from data on its own as compared to other machine learning approaches (Fu & Aldrich, 2020). It accomplishes this while using model structures that are skilled at capturing even the most complex behaviours. The implications of such a development are far-reaching, promising to revolutionize the way flotation circuits are monitored and controlled. By enhancing predictive capabilities, the industry can achieve higher yields, reduce waste, and minimize environmental impact, aligning with the sustainable development goals of the 21st century.

1.2 Problem Statement

Traditional process control and monitoring methods are limited in their ability to provide real-time plant data on concentrates and tailings, making it difficult to accurately predict recovery and grade on a shift or daily basis. The procedure of sampling and analysis is a time-consuming and laborious process and prevents real-time process insight. This results in a reduced yield due to limited real time information about the process behaviour. Online analysers are expensive and soft sensor based on easily available measurements could provide estimates where analysers are not economical to install.

1.3 Objectives of the Study

- To address the limitations of infrequent sample-based measurements by offering real-time insights into the dynamic behaviour of the plant.
- To evaluate the feasibility of machine learning models for predicting grades and recovery of platinum, palladium, rhodium, and gold based on measured process and froth parameters.
- To leverage the vast amounts of data collected in process industries which often go unused and are discarded, despite holding valuable insights into process dynamics and behaviours.
- To identify relevant high-frequency data and assay results that can form the basis for developing a robust, data-driven soft sensor.

1.4 Research Question

How can historical data from a flotation circuit be utilized through machine learning models to predict the grade and recovery in real-time. Which model provides the best performance in terms of accuracy, reliability, and computational efficiency?

1.5 Hypothesis

Machine learning techniques can be used to develop a predictive soft sensor capable of estimating flotation circuit grade and recovery in real time. Among the different machine learning models, there will be at least one that demonstrates superior performance in terms of prediction accuracy, reliability, and computational efficiency. The hypothesis will be tested by collecting and pre-processing a substantial dataset of historical process data from the flotation circuit. Identifying and integrating relevant process variables as model inputs. Implementing and comparing various machine learning algorithms, such as regression trees, support vector machines, and neural networks, to determine the most effective model. Evaluating the models' performance using appropriate metrics which includes the mean squared error (MSE), mean absolute error (MAE), and the R-squared (R^2).

1.6 Significance of the Study

The importance of this research is grounded in its potential to transform the mineral processing industry by introducing a novel approach for monitoring and optimising the flotation circuits. In utilising the already available historical data and machine learning algorithms, the soft sensor aims to provide real time predictions overcoming traditional empirical models limitation.

This approach enables optimized decision-making in mineral separation systems, directly enhancing the operational throughput and recovery efficiency. Real-time predictions enable operators to respond swiftly to changes in the flotation circuit, potentially reducing downtime and increasing throughput.

This efficiency gain translates to cost savings and higher productivity in mining operations. Accurate predictions of grade and recovery can help reduce the consumption of reagents and energy, thereby minimizing the environmental footprint of mining activities. This aligns with global efforts towards sustainable development and responsible resource management. By optimising the flotation process, the industry can achieve higher yields and better-quality products. This directly impacts profitability and competitiveness in the global market. This study contributes to the body of knowledge in the field of mineral processing by exploring the application of machine learning in an industrial context. It also opens new avenues for future research and technological innovation.

This study embodies the principles of Industry 4.0, showcasing how digitalization and smart technologies can be adopted in traditional industries such as mining. It serves as a model for other sectors seeking to harness the power of data analytics. The proposed soft sensor has the potential to establish a new standard for monitoring and controlling flotation circuits. This innovation delivers valuable improvements by advancing mining operations through enhanced technical capabilities, while aligning with global priorities for cost-optimized resource utilization and reduced ecological impact in industrial practices.

1.7 Scope and Limitations

This research covers the flotation circuit mainly the primary and secondary flotation cells that have an influence on the grade and recovery. It focuses on the already existing machine learning algorithms for regression tasks. It compares the performance of the different algorithms in predicting grade and recovery. It does not cover the deployment of the chosen algorithm and its maintenance in a real-time situation.

Chapter 2: Literature Review

2.1 Machine Learning in Industrial Processes

Machine learning involves creating computer programs that can learn and improve at tasks autonomously by using experience. Machine learning, a specialized domain within artificial intelligence (AI), centres on constructing computational systems that derive insights autonomously from datasets, eliminating dependence on predefined equations. These algorithms adaptively enhance their performance as they are exposed to more learning examples. According to Susmita (2009), machine learning is a prominent field within AI that plays a crucial role in digitalization solutions attracting significant interest in the digital landscape. It can be categorized into four primary types: supervised, unsupervised, semi-supervised, and reinforcement learning (Sah, 2020).

The rapid expansion of digitalization within manufacturing sectors is a key feature of the ongoing 4th Industrial Revolution, also known as Industry 4.0 (Fu & Aldrich, 2020). The mining industry in particular is undergoing a significant transformation in alignment with this trend. Machine learning is integral to this transformation especially in the development of soft sensors. In leveraging machine learning soft sensors can analyse vast amounts of process data to provide inferential measurements of variables that are difficult or impossible to measure directly. Soft sensors have emerged as critical tools for real-time process monitoring, control, and fault detection in modern industrial processes. These sensors ensure optimal operational efficiency, maintain product quality, and minimize downtime (Jin *et al.*, 2016). The knowledge derived from these data is obtained through machine learning, which has been evolving as a central component of artificial intelligence since the mid-20th century (Fu & Aldrich, 2020). Machine learning will assist industries to achieve higher levels of automation and intelligence. This enhances and drives forward the capabilities of soft sensors which will ultimately boosting the overall efficiency of industrial operations.

Hussain et al., (2024)explained that the development of these advanced sensors hinges on the fact that the values of desired outcomes or product quality are interconnected with other measurable process variables that can be monitored in real-time. According to Kadlec et al. (2009) in the initial stages of soft sensor applications, 'hard-to-measure' quality parameters were estimated using first-principles models or a combination of statistical and traditional methods. However, traditional statistical methods struggled to adequately capture spatial and temporal fluctuations in process data, resulting in unsatisfactory predictive performance (Perera *et al.*, 2023).

2.1.1 Advancements in Machine Learning for Industrial Processes

Industrial processes have historically relied on traditional methods for monitoring, control, and optimisation. These methods often involved manual data collection, human expertise, and relatively simple statistical models. While effective to some extent, these approaches had limitations in adapting to changing conditions, providing real-time insights, and dealing with large volumes of data. Machine learning emergence has brought transformative changes to industrial processes making them more efficient, accurate, and adaptive. Before its introduction industries primarily relied on process control systems that used predefined equations and manual adjustments. Techniques such as Statistical Process Control (SPC) and Proportional-Integral-Derivative (PID) controllers were commonplace. SPC involves using control charts to monitor process variability and ensure that it remains within acceptable limits. PID controllers, on the other hand, use feedback loops to maintain the desired process conditions. While these methods provided a basic level of control, they were often reactive rather than proactive and struggled with complex, non-linear relationships within process data (Montgomery et al., 2010).

The transition to machine learning in industrial processes began in the late 20th century, coinciding with advances in computing power and the availability of large datasets. Early applications of machine learning in industry focused on predictive maintenance, where algorithms were used to predict equipment failures based on historical data. This shift marked a significant departure from traditional methods and introducing a proactive approach to process management (Young & Rogers, 2019). One of the key milestones in this transition was the introduction of data-driven models that could learn from historical data and make predictions about future process states. These models moved beyond the limitations of predefined equations allowing for the capture of complex non-linear relationships. The techniques incorporated such as linear regression, neural networks, decision trees, and clustering algorithms became the foundation of early machine learning applications in industry (López et al., 2022).

The early 2000s marked a pivotal shift in machine learning, characterized by exponential progress in analytical methodologies. These advancements were catalysed by breakthroughs in computational power and the emergence of novel algorithmic architectures capable of autonomously adapting to complex datasets, superseding reliance on rigid, formula-driven models. Support Vector Machines (SVM) and Neural Networks (NN) gained popularity for their ability to handle high-dimensional data and their suitability in modelling complex relationships. These algorithms provided more accurate predictions and control capabilities, enabling industries to optimize their processes more effectively (Bishop & Nasrabadi, 2006).

The introduction of decision trees ensemble methods that involved Random Forests and Gradient Boosting Machines further enhanced the predictive power of machine learning models. Ensemble methods synthesize multiple computational models to strengthen predictive accuracy and system robustness, effectively countering critical challenges such as overfitting and prediction variability in data-driven applications. The Random Forest algorithm exemplifies this principle by integrating predictions from an ensemble of decision trees, thereby diminishing the influence of anomalies in individual models through aggregated decision-making (Breiman, 2001).

In recent years, the introduction of deep learning has revolutionized machine learning applications in industrial processes. Deep learning models, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have demonstrated exceptional performance in tasks such as image and sequence analysis. These models are capable of automatically extracting features from raw data, reducing the need for manual feature engineering and improving predictive accuracy (LeCun, Bengio and Hinton, 2015).

Today, machine learning is an integral part of industrial processes across various sectors, including manufacturing, mining, and energy. Soft sensors, which leverage machine learning algorithms to provide real-time inferential measurements, have become critical tools for process monitoring, control, and fault detection (Jin *et al.*, 2016). These sensors enable industries to achieve higher levels of operational efficiency, maintain product quality, and minimize downtime. The integration of Internet of Things (IoT) devices and the proliferation of big data have further propelled the adoption of machine learning in industrial processes. IoT devices provide continuous data streams from various process parameters, which can be analysed in real-time using machine learning models. This integration facilitates predictive maintenance, anomaly detection, and process optimisation, leading to significant cost savings and improved productivity (Lee *et al.*, 2007).

As machine learning continues to evolve, its applications in industrial processes are expected to expand further. The development of explainable AI aims to address the interpretability challenges associated with complex models, making it easier for industry professionals to understand and trust machine learning predictions. Additionally, advancements in transfer learning and federated learning are expected to enhance the ability of models to generalize across different datasets and collaborate without sharing sensitive data (Pan & Yang, 2009). Machine learning has introduced a proactive, data-driven approach to process monitoring, control, and optimisation, enabling industries to achieve unprecedented levels of efficiency and accuracy. As

technology continues to advance, the role of machine learning in industrial processes is set to become even more pivotal, driving innovation and competitiveness in the industrial sector.

2.1.2 Data-Driven Models in Mineral Processing

Machine learning is about creating computer programs that can learn and get better at tasks by themselves, using experience. It can be categorized into four primary categories namely, supervised, unsupervised, semi-supervised, and reinforcement learning (Sah,2020). Supervised learning is applied when the data available has both input and output while unsupervised only the input is required. On the other hand, reinforcement learning is an action-based decision learning where the algorithm learns to select actions based on given situations, and its choices influence both current and future situations (François-Lavet et al, 2018). According to McCoy & Auret, (2019) supervised learning is the widely used form of machine learning which involves providing an algorithm with labelled training data and having it make predictions on new, unlabelled data by using patterns it learns from the labelled examples. In data-based modelling, easily measurable variables serve as input data, which is then processed using various modelling techniques, these techniques help estimate difficult-to-measure variables as shown in figure 2.

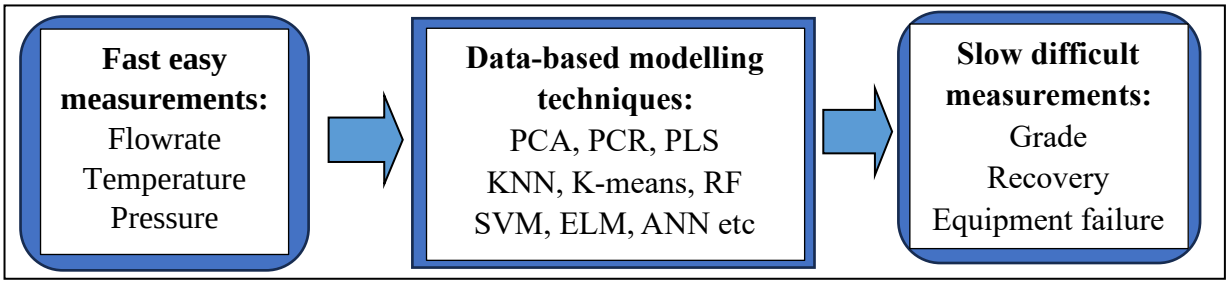


Figure 2: *Regular information flow in data-based modelling tasks, showing example inputs and results (McCoy & Auret,2019).*

Semi-supervised learning combines elements of both supervised and unsupervised approaches, utilizing a small amount of labelled data alongside a larger set of unlabelled data. This hybrid method can be particularly useful in scenarios where obtaining labelled data is expensive or time-consuming. Transfer learning, another important machine learning paradigm, involves applying knowledge gained from one task to a different but related task, potentially reducing the amount of training data required for the new task.

There are two distinct model approaches for soft sensor development which include white-box models based on physical process knowledge and black-box or data-driven models constructed solely from empirical process data (Souza et al., 2016). Chitralekha et al, 2010 mentioned the grey-box model which incorporates both the prior knowledge and empirical process data for the model development. When utilizing black-box modelling, the initial decision to make is the type

of model to employ, presenting a choice between a linear or nonlinear model. Several authors suggest giving precedence to the utilization of a linear model as the primary choice, citing its architectural simplicity and computational efficiency compared to nonlinear models.

In the development of data-driven soft sensors, the approach involves offline modelling using recorded historical data where the collected historical recordings are treated and then employed for model identification. However, this task is not trivial as emphasized by authors (Dong & McAvoy, 1996), that historical data is data-rich yet information-poor. According to Dong & McAvoy (1996) the historical data must encompass all potential future states and conditions of the process, encompassing not just operational states but also those linked to environmental shifts, alterations in process input materials, and more. In an attempt to solve this problem many authors have proposed adaptive techniques which can be incorporated in the model to ensure that soft sensors learn new process states and adapt accordingly in real- time.

According to many authors particularly François-Lavet et al. (2018) stated that data collected from real-world industrial applications comes with a set of recognized challenges that need to be managed. These challenges include factors like sampling intervals, outliers, missing data, and more. These problems are normally brought by faults in equipment (physical sensors), different delays in measuring frequency of different sensors employed, and incorrect readings from instrumentation. The performance of soft sensors is directly impacted by the quality of the historical data (Jiang *et al.,* 2021). Therefore, it is very crucial that these problems be managed before training to ensure the reliability and accuracy of the soft sensor is improved.

2.1.3 Prediction Grade and Recovery

Machine learning techniques application in the flotation process is increasing due to their ability to overcome the current challenges faced by process experts in trying to control or improve its performance. Recent studies have demonstrated their effectiveness in improving process insight and control. For instance, Maldonado et al. (2020) developed a soft sensor for estimating flotation concentrate grade using principal component regression combined with adaptive filtering, enabling more robust predictions under dynamic conditions. Similarly, Wei and Craig (2022) reviewed the integration of soft sensors in grinding and flotation circuits, noting that modern data-driven models, such as artificial neural networks and random forests, outperform traditional regression approaches in handling nonlinearity and noise. In a copper flotation circuit, Gharabaghi et al. (2021) implemented a soft sensor that combined froth image features with air flow and pH readings to predict recovery, which was later embedded into a model predictive control (MPC) framework.

These developments align with the rise of adaptive control architectures such as reinforcement learning and adaptive MPC that adjust control strategies based on continuous model updating (Chen et al., 2023). The literature reflects a growing trend towards integrating soft sensors into closed-loop adaptive frameworks, offering real-time decision-making support and operational stability in complex, multivariable mineral processes. A review of the use of machine learning in mineral processing from McCoy & Auret (2019) found that there are 95 instances of machine learning implementation in flotation, with 16 of those involving data-driven modelling. Additionally, they identified 75 cases in the domain of machine vision as it can be seen in the following table.

Table 1: *A summary of implementing machine learning techniques in processes (McCoy & Auret, 2019)*

Process application	Data-based modelling	Fault detection and diagnosis	Machine vision	Total
Flotation	16	8	71	95
Ore particle sizing	2	0	26	28
Milling circuits	15	7	3	25
Concentration	2	6	3	11
Smelting/furnaces	12	8	0	20
Hydrometallurgy	1	3	0	4
Other	5	1	4	10

In table 1 it can also be seen that the top three leading process applications of machine learning are flotation, ore particle sizing and milling circuit which are all beneficiary processes for valuable metal extraction. This highlights the necessity and the potential of improvement of performance in processes with machine learning as if all the sub-processes are well controlled and monitored then improved production will be achieved. Therefore, machine learning in flotation circuits will significantly improve the extraction of the valuable metals hence the increase in research around this field. Traditional offline methods for acquiring grade measurements through analytical laboratory techniques are unsuitable for dynamic operations due to their long sampling and measurement durations. The artificial intelligence discovery is believed to enhance the performance of the already existing online sample analysers such as online X-ray Fluorescence which have been incorporated into multiple facilities to infer grade and tailings measurement with improved response times (Popli *et al.*, 2018). This will introduce machine vision and improve their high maintenance requirement, help reduce the high initial cost, solve the difficult calibration problem, and errors in capturing data.

Machine vision has the capability to swiftly and precisely capture features of froth, including physical aspects like bubble size, and dynamic properties such as froth velocity from digital images (Aldrich *et al.*, 2010). These outcomes can then be conveyed to operators or employed as inputs for process control systems. Currently, numerous industrial sites globally have incorporated machine vision technology, and its progress is consistently fuelled by advancements in computer technology. Mehrabi et al., (2014) created a machine vision system to oversee working conditions, encompassing bubble size distribution, the count of bubbles, froth velocity, and stability. Machine vision was followed by data-modelling (soft sensors) which capitalizes the historical process data stored by the online analysers. Originally, Soft Sensors were examined within the context of chemometrics, a discipline focused on statistical methods employed to extract information from datasets comprising a multitude of measured variables (Wold, 1995).

2.2 Machine Learning Techniques for Soft Sensor Development

Machine learning (ML) techniques have significantly advanced the development of soft sensors for industrial processes, particularly in the field of mineral processing. Soft sensors provide inferential measurements of unmeasured or difficult-to-measure variables are indispensable for real-time process monitoring and control (Kadlec et al., 2009). This section delves into the different learning strategies of machine learning algorithms. This includes supervised learning, unsupervised learning, reinforcement learning, and deep learning. We will further explore the concepts of black-box and white-box approaches as machine learning development strategies. The distinction between black-box and white-box models is crucial in the context of soft sensor development.

The black-box approach in machine learning refers to models or algorithms where the internal workings of how the algorithm generalizes are not easily interpretable or transparent. In this approach, the focus is on input-output relationship according to the given inputs. The model produces outputs without providing an explanation of how the result was derived. Common black-box models include deep neural networks, ensemble methods, and support vector machines, characterized by complex computations or layers of transformations that are challenging to analyse directly (Fabra-Boluda *et al.*, 2024).

These models are effective at making predictions or identifying patterns in data. However, the lack of transparency in their decision-making process poses challenges for developers in debugging and ensuring model fairness or bias mitigation. LeCun & Hinton (2015) emphasized the importance of model interpretability and the lack of interpretability for most models is a

drawback in industrial applications where understanding the model's decision-making process is crucial for validation and building trust. To address this Fabra-Boluda et al., (2024) proposed an innovative strategy that systematically queries a black-box model to label an artificially generated dataset which is then utilized to train various surrogate models from different families each designed to approximate the behaviour of the original black-box model.

Unlike black-box models, white-box models prioritize transparency and interpretability. In a white-box approach, the model's internal mechanisms and decision-making processes are fully accessible and understandable. This transparency allows developers to clearly understand how the model generalizes from inputs to outputs, facilitating easier debugging, validation, and trust-building. Typical examples of white-box models include decision trees, linear regression, and rule-based systems, all of which are relatively straightforward to analyse and explain. While white-box models may be less flexible than black-box models in capturing complex patterns in data, their ease of interpretability makes them a preferred choice in scenarios where transparency, fairness, and ease of model maintenance are prioritized. However, they may not capture complex, non-linear relationships as effectively as black-box models. In practice, the choice between black-box and white-box approaches depends on the specific requirements of the application, including the need for accuracy, interpretability, and computational resources (Breiman, 2001). Hybrid approaches like the grey box which combine elements of both are also gaining traction offering a balance between predictive performance and interpretability (Hastie, Tibshirani and Friedman, 2009).

The integration of machine learning techniques into soft sensor development has revolutionized industrial process monitoring and control (Kadlec et al.,2009). Supervised learning has been widely adopted due to its ability to learn from labelled data and make accurate predictions on new unseen data (Rexie *et al.,* 2023). This method is highly valuable for creating reliable soft sensors that can infer difficult to measure variables from easily obtainable process data, thereby enhancing process control strategies and ensuring higher efficiency and product quality (Kay *et al.,* 2024).

In addition to supervised learning, unsupervised learning techniques are employed to identify hidden patterns or intrinsic structures in data without predefined labels, aiding in feature extraction, anomaly detection, and data clustering (Jolliffe, 2002). Reinforcement learning is another crucial approach where an agent learns to make decisions by interacting with an

environment, receiving feedback in the form of rewards or penalties, and aiming to maximize cumulative rewards over time (Ai & Dltf, 2024). Each of these machine learning paradigms offers unique strengths and applications, making them integral to the development of advanced soft sensors. The following section will focus on supervised learning, as it is the approach of our research, and will explore these methodologies in greater detail, highlighting their applications, strengths, and limitations.

2.2.1 Supervised Learning Techniques

Supervised learning is a fundamental paradigm in machine learning where a model is trained using a labelled dataset, meaning each training example is paired with an output label. The objective of supervised learning is to learn a mapping function from inputs to outputs that can generalize well to unseen data, thus enabling accurate predictions on new, unseen examples. This process involves minimizing a loss function that quantifies the discrepancy between the model's predictions and the actual labels. Common techniques include regression for predicting continuous values and classification for predicting discrete labels. The supervised learning framework is widely applicable across various domains such as image recognition, natural language processing, and particularly in developing soft sensors for industrial applications (Russell & Norvig, 2016).

Regression and classification are two core applications of supervised learning, each serving distinct purposes across various industries. Regression algorithms predict continuous numerical values and are widely used in sectors such as finance for stock price prediction, healthcare for predicting patient outcomes, and manufacturing for quality control and process optimisation. In the mining industry regression models assist in predicting grade and recovery rates in mineral processing plants thereby aiding in real-time decision-making and efficiency improvement (Herrera *et al.*, 2023). On the other hand, classification algorithms predict discrete labels and are essential in applications like fraud detection in banking, where transactions are classified as fraudulent or legitimate, and in medical diagnosis, where symptoms are classified to determine the presence of diseases. In telecommunications, classification models help predict customer churn by categorizing customers based on their likelihood of continuing service. Both regression and classification enable industries to leverage data for informed predictions and decisions, driving operational excellence and innovation (Haykin, 1999).

Supervised learning encompasses a variety of algorithms tailored for both regression and classification tasks, each suited to specific data types and problem requirements. Common

regression algorithms include Linear Regression, Support Vector Regression (SVR), Random Forest Regression, and Gaussian Process Regression (GPR). These algorithms are designed to predict continuous values and find applications in diverse industries such as finance for stock price predictions, healthcare for patient outcome estimations, and manufacturing for process optimisation. Popular classification algorithms include Logistic Regression, Support Vector Machines (SVM), Decision Trees, Random Forests, and Neural Networks. These models are employed for predicting discrete labels in tasks like fraud detection in banking, medical diagnosis, and customer churn prediction in telecommunications. Additionally, some algorithms, such as Random Forests and SVM, are versatile and can be adapted for both regression and classification tasks. By utilizing these algorithms, industries can enhance predictive accuracy and make well-informed decisions, thereby supporting a wide range of applications (Haykin, 1999).

In the context of platinum concentrators, supervised learning techniques play a crucial role in predicting process variables such as grade and recovery, which are essential for optimising the flotation circuit. Soft sensors, which are models that infer process states from available measurements, rely heavily on these techniques. For instance, regression algorithms can predict the recovery rates by mapping operational parameters and sensor readings to output variables, providing real-time insights that enhance process efficiency and control. Furthermore, advanced methods like Support Vector Machines (SVM) and Artificial Neural Networks (ANNs) offer robust solutions by capturing complex non-linear relationships in the data, thereby improving prediction accuracy and reliability (Haykin, 1999).

This capability is indispensable for the dynamic and multifaceted nature of mineral processing, where accurate predictions can lead to significant economic and operational benefits. Given the significant role that supervised learning techniques play in optimising the flotation circuit of platinum concentrators, it is crucial to delve deeper into the specific regression techniques employed in this domain. Regression algorithms are particularly valuable as they provide continuous output predictions, which are essential for real-time process control and efficiency enhancement. This research will explore various regression techniques in detail, focusing on their applications, strengths, and limitations within the context of mineral processing. By examining these regression techniques, this research aims to identify the most effective methods for developing soft sensors to predict grade and recovery rates, ultimately contributing to the optimisation of the flotation circuit in platinum concentrators (Breiman, 2001).

2.2.2 Unsupervised Learning

Unsupervised machine learning is a technique in which a model learns patterns and relationships from unlabelled data without predefined target variables (Goodfellow et al.,2016). Unlike supervised learning, where the model is trained using input-output pairs, unsupervised learning aims to discover hidden structures, patterns, or groupings in data. The core principle behind unsupervised learning is that it identifies inherent similarities within the data, clustering similar instances or reducing the dimensionality of the data to make complex datasets more interpretable (Murphy, 2022). Common unsupervised learning algorithms include Clustering (e.g., K-means, hierarchical clustering, DBSCAN), which groups similar data points; Dimensionality Reduction techniques (e.g., Principal Component Analysis (PCA), t-Distributed Stochastic Neighbour Embedding (t-SNE)) reduce the number of features while retaining essential patterns in the data. Association Rule Learning for example Apriori, Eclat algorithms which identify relationships and associations within the dataset. These algorithms are essential in scenarios where the dataset lacks labelled examples or when the goal is to uncover unknown patterns rather than make direct predictions (Bishop & Nasrabadi, 2006).

In the mineral processing industry, unsupervised learning techniques have become increasingly valuable for process optimisation, anomaly detection, and quality control (McCoy & Auret, 2019). Clustering algorithms, are widely used to group mineral samples based on their composition or properties, enabling process engineers to better understand ore variability and improve separation efficiency. Additionally, dimensionality reduction methods like PCA help in analysing and visualizing high-dimensional data from sensors. This is important for monitoring the performance of equipment like crushers, flotation cells, and grinding mills (Wills & Finch, 2015). This approach helps extract important features from complex process data which can then be used to optimize process parameters. Unsupervised learning techniques, such as anomaly detection algorithms, are increasingly leveraged in the mining industry for predictive maintenance. These systems analyse real-time sensor data from vital process equipment to detect deviations from normal operational patterns. Identifying the subtle anomalies early assist operators to schedule maintenance proactively, mitigating unplanned downtime and avoiding catastrophic failures (Rodriguez-Galiano *et al.*, 2015). This results in reduced downtime, cost savings, and improved operational efficiency. Overall, the use of unsupervised learning in the mineral processing industry supports better decision-making, enhances process understanding, and provides opportunities for significant performance improvements.

2.2.3 Reinforcement Learning

Reinforcement Learning (RL) operates as a computational paradigm within machine learning, wherein an autonomous agent develops decision-making strategies through continuous interaction with a dynamic environment. This framework is orchestrated to optimize a cumulative reward metric across iterative cycles, prioritizing long-term operational efficacy over immediate outcomes (Sutton and Barto, 2018). Unlike supervised learning, where labelled data is provided, or unsupervised learning, where patterns are discovered from unlabelled data, RL is characterized by learning from experience through trial and error. The agent takes actions in a given state, receives feedback in the form of rewards or penalties, and uses this feedback to adjust its future actions. This approach is structured around three foundational elements namely the state, which encapsulates the environment's real-time conditions. The action which is defined by the spectrum of viable decisions the agent may implement. Lastly the reward which is a performance metric that evaluates the efficacy of state transitions initiated by agent decisions (Pack Kaelbling *et al.*, 1996). Popular RL algorithms include Q-Learning, a value-based method that seeks to learn the optimal action-value function, and Policy Gradient Methods, which directly optimize the policy that maps states to actions. RL has applications in situations that require sequential decision-making, and where the goal is to learn optimal policies over time through interactions.

In the mineral processing industry, RL has shown significant potential in process optimisation, real-time control, and adaptive decision-making (McCoy and Auret, 2019a). For instance, in flotation processes which are complex and require dynamic control of variables like reagent dosages, airflow rates, and cell levels to maximize recovery and grade. This approach can be applied to develop intelligent controllers that continuously adapt to changing ore properties and operational conditions, thus optimising the performance of the flotation process (Zhou *et al.*, 2020). Additionally, RL has applications in optimising grinding circuits, which are energy-intensive processes that require efficient control to balance throughput, energy consumption, and product quality. By using RL-based controllers, mineral processing plants can achieve more robust and efficient operations, adapting to disturbances and improving the economic efficiency of the overall process. This technology also has potential applications in predictive maintenance, equipment health monitoring, and automated decision-making, further enhancing productivity in mineral processing. RL is particularly suitable for applications requiring sequential decision-making and adaptive control.

2.2.4 Deep Learning

This is another subset of machine learning that focuses on training neural networks with many layers to model complex patterns and relationships in data (Goodfellow, Bengio and Courville, 2016). The basic building block of deep learning is the artificial neural network (ANN), which consists of an input layer, multiple hidden layers, and an output layer. Each layer contains neurons that are interconnected, and these connections are weighted, allowing the network to learn the importance of different features. During training, the model uses techniques like backpropagation to adjust these weights to minimize error in its predictions.

This hierarchical structure enables deep networks to learn abstract representations and features directly from raw data, which makes them powerful for tasks like image classification, speech recognition, and natural language processing (LeCun, Bengio and Hinton, 2015). Popular deep learning architectures include Convolutional Neural Networks (CNNs), which are effective for image and video analysis. Recurrent Neural Networks (RNNs) are suitable for sequential data like time series or text and Transformers which are highly efficient for language modelling and sequence-to-sequence tasks. Deep learning models often require large amounts of data and computational power but have demonstrated superior performance in extracting complex patterns and features from high-dimensional data.

In the mineral processing industry, deep learning has found applications in several areas, including process optimisation, predictive maintenance, and anomaly detection (Huang *et al.*, 2015). CNNs are particularly effective for analysing images in mineralogy, such as identifying different mineral types or predicting ore characteristics based on digital images of rock samples (Cai *et al.,* 2024). RNNs and Long Short-Term Memory (LSTM) networks are used to model time-series data from sensors in mineral processing plants to predict equipment failures or optimize control systems in processes like grinding, flotation, and thickening. Furthermore, deep learning models are employed for analysing sensor data to detect faults or operational inefficiencies in real-time, providing opportunities for better process control and efficiency improvement (Dayo-Olupona *et al.,* 2023). This can result in significant cost savings and enhanced productivity by enabling proactive maintenance and better decision-making. Overall, deep learning is transforming the mineral processing industry by enabling more accurate predictions, intelligent automation, and the discovery of patterns that were previously too complex to identify using traditional approaches.

An extension of deep learning is transfer learning, which is particularly valuable for complex deep models that are difficult and time-consuming to build from scratch. Transfer learning enables developers to leverage pre-trained models that have been trained on large datasets for similar tasks. This simplifies the development process and reducing the need for extensive training data. Deep learning models, such as Convolutional Neural Networks (CNNs) for image classification or Recurrent Neural Networks (RNNs) for sequence data, require significant computational resources and domain expertise to design and train (Yosinski et al., 2014). Transfer learning addresses this by allowing the use of an existing model that has already learned low-level features like edges, textures, and shapes in the case of image data, or temporal dependencies in sequence data. The model can then be fine-tuned or adapted to a new, but related, task, making it particularly useful when data availability is limited, or when training from scratch would be impractical.

2.3 Exploration of Regression Algorithms

Regression algorithms are foundational techniques in supervised learning, crucial for developing soft sensors in industrial process control. They are widely used in industrial application where continuous monitoring and control of process is crucial (Guo *et al.*, 2020). Real-time prediction of important performance metrics is crucial in industries such as mining, oil and gas, chemical manufacture, and food production, since it helps to ensure process stability. These algorithms predict continuous outputs by modelling the relationship between input variables and a target variable. Regression algorithms make it possible to forecast important performance metrics in the context of process control, including grade, recovery rates, and other process parameters. This makes real-time monitoring and optimisation possible.

Regression models under supervised learning provide a mathematical framework to predict these continuous variables based on historical data, allowing for better process optimisation, fault detection, and decision-making. By modelling the relationship between easily measurable inputs and target outputs complex process variables readily available cost effectively. The advancements in machine learning and data-driven approaches, the range of regression algorithms has expanded, from traditional linear models to more sophisticated methods like support vector regression and neural networks. There are several types of regression algorithms, each with distinct applications and advantages, depending on the complexity of the process, the availability of data, and the level of accuracy required. This section explores various regression algorithms, their applications in industrial process control, and their importance in developing effective soft sensors for improving operational efficiency and performance.

2.3.1 Linear regression

Linear regression is one of the simplest and most interpretable regression techniques. It models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. It is a straight line or a straight hyperplane in higher dimensions that is fitted to the data to explain the existing relationship between variables and the target independent variable. It involves fitting a straight line or a straight hyperplane in higher dimensions to the data in order to capture and explain the relationship between the input variables and the target independent variable.

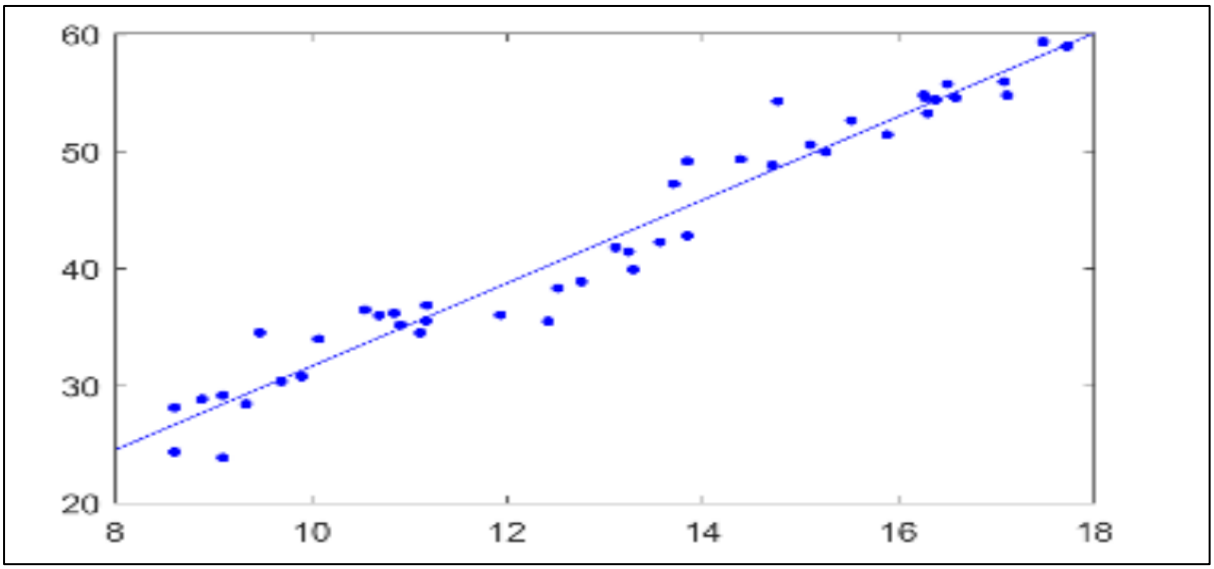


Figure 3: *An example of simple linear regression with a single feature or variable (Shanthamallu et al., 2017).*

To make predictions, the observed pattern is used due to the linear dependency between the variables and the target. Figure 3 illustrates how the straight line is fitted to the data which can also be in a sigmoid curve depending on the structure and pattern of the data. This method is applicable when the relationship between the variables in the dataset is linear. Despite its simplicity, linear regression is often used as a baseline model in many applications. Linear regression is divided into two main categories: simple linear regression, which involves a single independent variable, and multiple linear regression (MLR), where two or more independent variables are used to predict the outcome (Maulud et al., 2020). Linear regression is often the first port of call when tackling regression problems in machine learning.

It serves as a foundational technique for several reasons. Firstly, it is relatively simple to understand and implement, making it an excellent starting point for industrial problems. Secondly, even if the underlying relationship between variables is non-linear, linear regression can still provide valuable insights. In analysing the fit of the model, we can assess whether a

non-linear approach might be more suitable (Montgomery et al., 2010). It assumes a linear relationship between a dependent variable (y) and one or more independent variables (x). This relationship is expressed by the following equation (Maulud et al., 2020):

$$f(x) = \beta_0 + \beta_1 x + \varepsilon \qquad (1)$$

where:

- $f(x)$ is the target variable or y
- β_0 is the intercept (the y-axis value where the line crosses)
- β_1 is the coefficient for the independent variable representing the slope of the line
- ε is the error term, accounting for any unexplained variance

For multiple linear regression the equation become extended as follows:

$$f(x) = \beta_0 + \beta_1 x_1 + \cdots \beta_n x_n \qquad (2)$$

where x_1 to x_n are the independent variables and β_1 to β_n are the coefficients associated with each. The intercept β_0 remains a constant as the value of the dependent variable when all independent variables are zero. Allowing multiple predictors in MLR provides a more flexible and powerful model for understanding complex relationships between variables. The core objective of linear regression is to find the optimal values for the coefficients that minimize the difference between the predicted y values (based on the equation) and the actual y values in the data. This difference is typically measured using the mean squared error (MSE) function. Linear regression offers several advantages as well as notable limitations. Firstly, it is highly interpretable, as the coefficients directly translate to the impact of each independent variable on the dependent variable, making the model easier to understand. Secondly, it is computationally efficient and requires minimal data pre-processing compared to more complex models. Additionally, linear regression serves as a valuable baseline model, even if the true relationship is non-linear, providing a benchmark for comparison with more sophisticated algorithms.

The primary limitation is the assumption of linearity; if the underlying relationship is non-linear, linear regression will not capture the true pattern, leading to inaccurate predictions. Furthermore, without proper regularization techniques, linear regression models can overfit the training data, resulting in poor performance on unseen data. In conclusion, linear regression offers a solid foundation for regression analysis due to its simplicity, interpretability, and utility as a baseline model. However, it is crucial to be aware of its limitations, particularly the assumption of linearity.

2.3.2 Support Vector Regression

Support Vector Regression (SVR) extends the principles of Support Vector Machines (SVM) to regression problems. These algorithms are commonly employed for classification tasks in various studies and have been adapted for use in regression problems in the form of SVR (Smola & Schölkopf, 2004). It works by finding a hyperplane in a high-dimensional space that best fits the input data while minimizing a margin of error. The hyperplane is defined using support vectors which are specific training data points that lie outside the boundaries of the margin or tube (Awad & Khanna, 2015).

These support vectors are critical in determining the position and orientation of the hyperplane. Support Vector Regression (SVR) approaches the problem of function approximation by framing it as an optimisation task. The goal is to identify the narrowest possible tube or margin around the predicted surface that still contains most of the data points as shown in figure 4. Simultaneously, SVR aims to minimize the prediction error defined as the difference between the predicted outputs and the actual or desired outputs. The optimisation balances fitting the data within this tube while keeping the prediction error small, resulting in a robust and generalizable model.

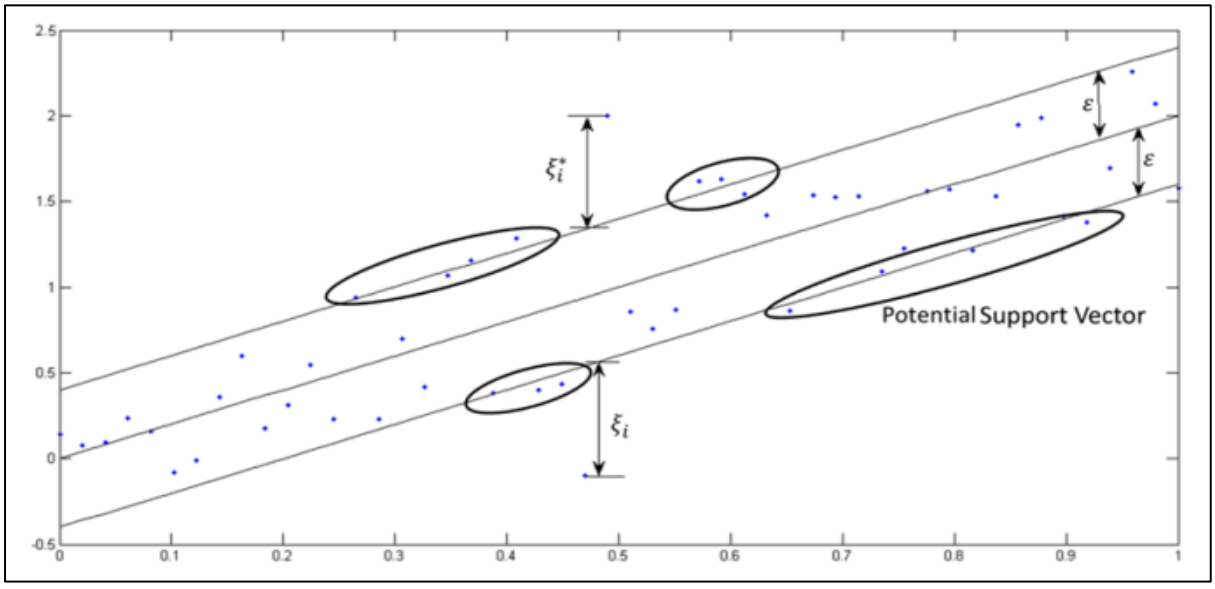


Figure 4*: Linear SVR in one-dimensional* (Awad & Khanna, 2015).

The primary goal of SVR is to find a function that deviates from the actual observed values by a value no greater than a specified margin, while simultaneously being as flat as possible. This approach ensures that the model is robust and generalizes well to unseen data. SVR uses a concept known as the kernel trick to transform the input data into a higher-dimensional space where a linear relationship can be established. The most commonly used kernels include the linear, polynomial, and radial basis function (RBF) kernels.

In mapping the data into this higher-dimensional space, SVR can capture complex, non-linear relationships between the input variables and the target variable, which is a significant advantage over linear regression (Cortes et al., 1995). Asymmetric loss function applies equal penalties for both overestimation and underestimation.

Vapnik's ε-insensitive method assist in creating a tube of minimal radius which is flexible around the estimated function. Errors with absolute values smaller than a specified threshold (ε) are disregarded if found above and below the estimated function. Since points that fall outside the tube are penalized, those within the tube are not penalized. The following are the formulars are responsible for the algorithm to make a prediction.

$$f(x) = \begin{bmatrix} x \\ 1 \end{bmatrix} \begin{bmatrix} \omega \\ b \end{bmatrix}^T = \omega^T x + b \;\; x, w \in \mathbb{R}^{M+1} \quad \ldots\ldots 3$$

$$min_w \; \frac{1}{2}\|w\|^2 \quad \ldots\ldots 4$$

$$f(x, w) = \sum_{i=1}^{M} w_i x^i, x \in \mathbb{R}, \;\; \omega \in \mathbb{R}^M \quad \ldots\ldots 5$$

All the equations are sourced from (Awad & Khanna, 2015) equation 3 represents the linear prediction function where ω is the weight vector, *b* is the bias term, and x is the input data. The equation defines the relationship between the input features and the predicted output by fitting a linear hyperplane to the data. The following equation 4 minimizes the magnitude of the weight vector. Minimizing helps to find the optimal hyperplane that balances maximizing the margin which is the space between the support vectors and minimizing the prediction error. When weights are small the model generalizes better preventing overfitting.

The last equation is the summation of the weighted inputs in the prediction process. Each w_i weight corresponds to the contribution of each x^i feature in the input. The summation helps calculate the predicted output based on all input features in multiple dimensions. M represents vector w increases its elements become nonzero leading to higher-order solutions. Figure 5 illustrates the algorithm performance with different orders. As the order increases the performance also increases but at very high orders the is a risk of overfitting.

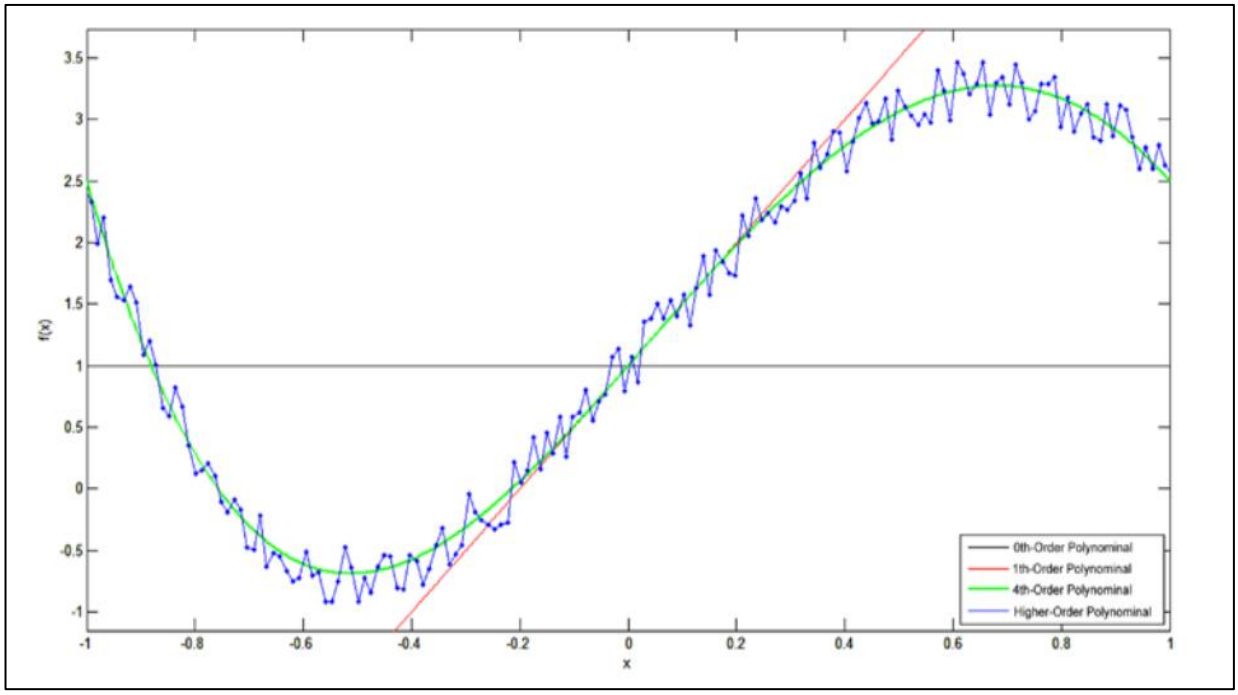


Figure 5: *SVR prediction for various orders* (Awad & Khanna, 2015).

Several loss functions can be used in the model such as linear, quadratic, and Huber loss functions (Awad & Khanna, 2015). The Huber loss function provides a smoother alternative compared to the linear and quadratic functions. It penalizes all deviations from the desired output with penalties which increases with an increase in error. The selection of a loss function depends on prior knowledge about the noise distribution, model sparsity requirements, and computational complexity during training. While the loss functions discussed are symmetrical and convex, ensuring a unique solution, asymmetrical functions can also be used to prioritize either underestimation or overestimation.

SVR is effective in capturing non-linear relationships by using kernel functions to map input data into higher-dimensional spaces. It's robustness to overfitting and its ability to handle high-dimensional data make it suitable for complex industrial processes where the relationship between input and output variables is not straightforward (Cortes et al., 1995).The advantages of SVR include its ability to handle non-linear data, robustness against overfitting with appropriate regularization, and flexibility in choosing different kernel functions. However, notable disadvantages include computational complexity for large datasets and the need for careful parameter tuning that tends to be time-consuming. Additionally, SVR models are less interpretable than linear regression models, particularly when non-linear kernels are used. Its computational demands and the need for meticulous parameter tuning are important considerations for practical applications

2.3.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are advanced machine learning models that are inspired by how the human brain's neural network works. They consist of interconnected layers of neurons, or nodes, which process input data in a non-linear fashion to learn complex patterns and relationships within the data. The primary justification for the increased application of Artificial Neural Networks (ANNs) in the field of extractive metallurgy is the declining ore grades. This decline has led to more complex extraction processes, which demand advanced tools for prediction and optimisation to maintain efficiency.

The most frequently used ANN algorithm in this context was found to be the Levenberg-Marquardt Algorithm (LMA), primarily due to its capability to handle large volumes of data and deliver results within a short period. LMA's efficiency in processing complex datasets makes it an ideal choice for optimising extraction processes in response to the increasing complexity driven by declining ore quality. This shift towards ANN and LMA reflects the industry's need for more adaptive and high-performance computational tools. Research has shown that these models are highly effective for regression tasks in industrial applications, where they excel at modelling complex relationships between input variables (Haykin, 1999).

This capability enables accurate prediction of critical output parameters, including product quality, process efficiency, and overall system performance architecture of ANNs typically includes an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to neurons in the subsequent layer, with each connection assigned a weight that is adjusted during training to minimize the error in the network's predictions. The training process involves using algorithms such as backpropagation, which iteratively updates the weights to reduce the difference between the predicted outputs and the actual observed values. This learning process allows ANNs to capture non-linear and high-dimensional relationships in the data, which are often challenging for traditional linear models to handle effectively. Artificial Neural Networks are particularly advantageous for regression tasks in industrial processes due to their ability to model complex, non-linear relationships through continual learning.

Their multi-layer structure and non-linear activation functions allow for accurate modelling across dynamic system states, effectively handling the evolving complexities and wide operational range typical of industrial processes (Grote-Ramm *et al.*, 2023). Like the human brain, they can adapt to various problems by modifying their architecture, such as altering the number

of hidden layers and neurons, to suit specific tasks effectively. Beaglehole et al, (2024) further illustrated that neural networks are capable of learning relevant features or patterns from raw input data without the need for manual feature engineering using a mechanism called Average Gradient Outer Product (AGOP) that characterizes how neural networks learn these features across various architectures.

A neural network structure is presented in figure 6 consists of layers of interconnected nodes or neurons. Each neuron in a layer takes an input, applies weight and bias, and then passes the result through a non-linear activation function before sending it to the next layer. Generally, these algorithms have an input layer to receive the data followed by one or more hidden layers to perform computations and learn patterns, and an output layer to produce the final prediction. The connections between the neurons are characterized by weights that adjust during training to minimize the error between the predicted and actual outputs. This structure enables the network to model complex relationships and learn directly from the data by tuning the weights across layers.

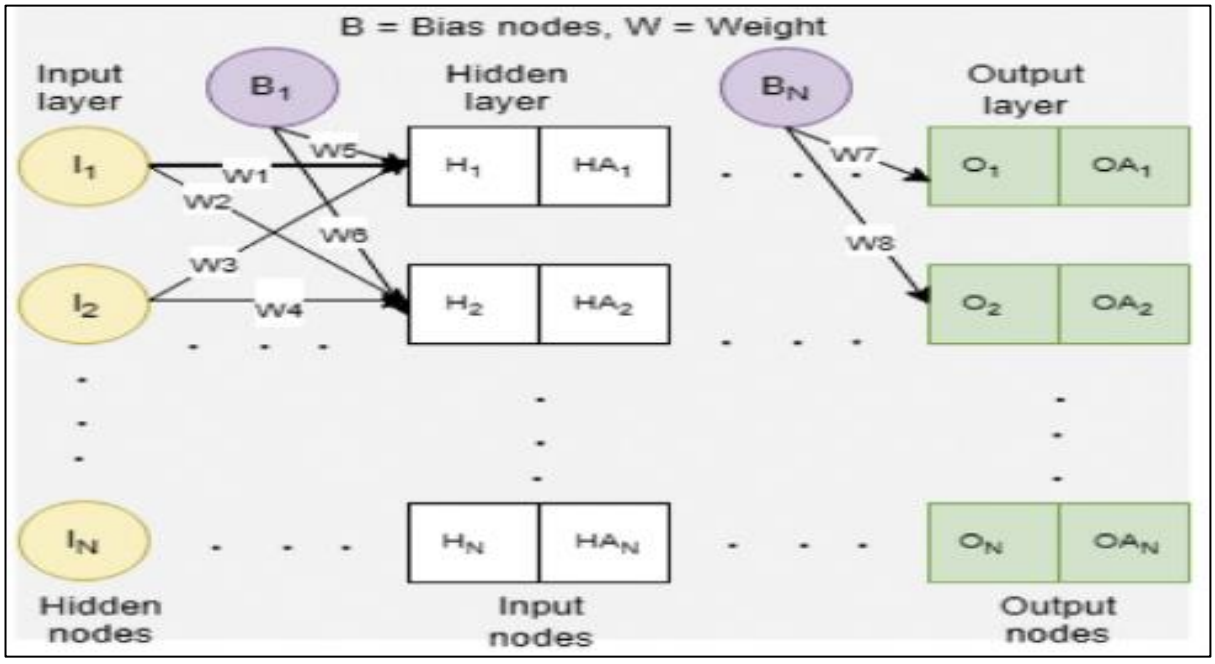


Figure 6: *Basic structure of a neural network* (Isaac Abiodun *et al.*, 2018).

In neural networks, data propagates sequentially from the input layer through hidden layers to the output layer. Figure 7 illustrates these distinct architectural layers and the directional flow of information, demonstrating how the system progresses toward generalisation. The input layer ingests raw data normally numerical values, text or images and transmits it without modification. Hidden layers process this information via weighted connections and activation functions extracting hierarchical features progressing from basic patterns to complex

representations. The output layer then synthesizes these features to generate task-specific predictions or classifications.

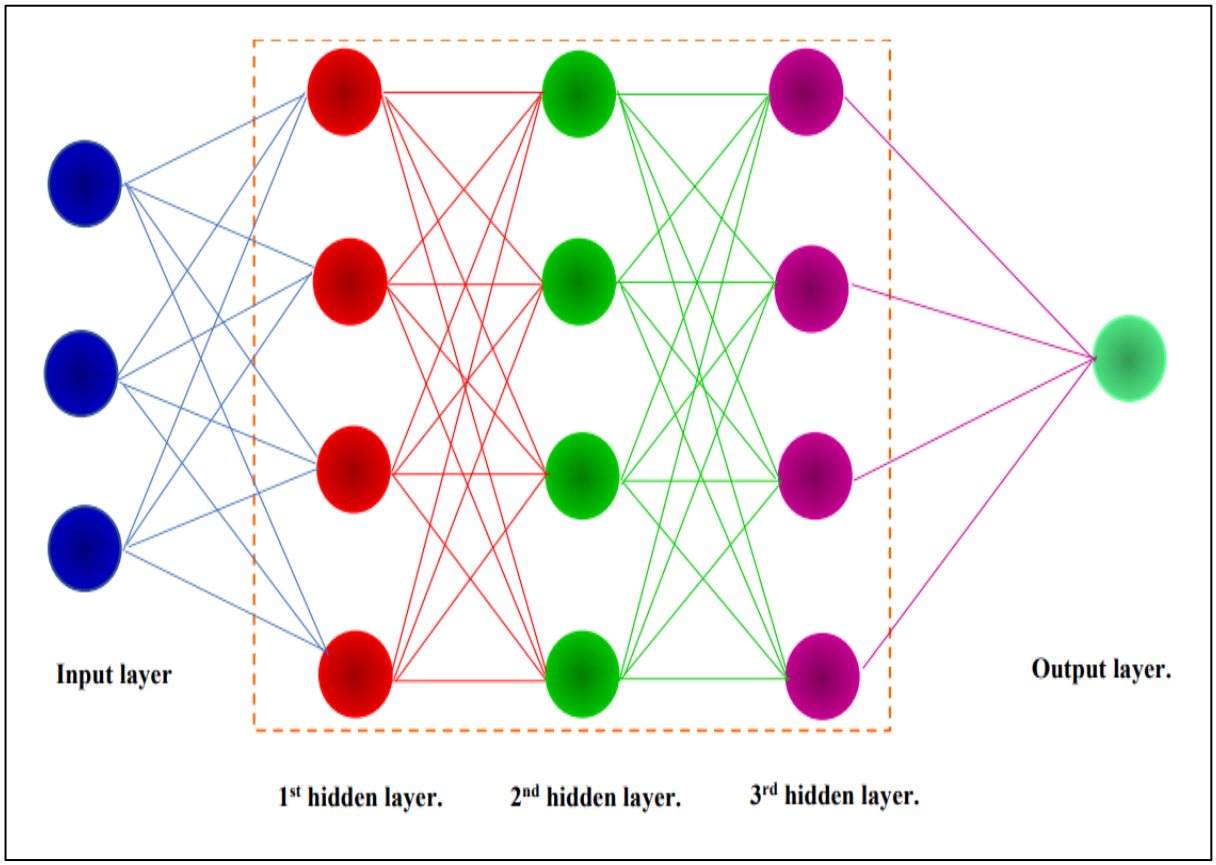


Figure 7: *Connection of different neural network connections from the input to the output layer* (Pani & Mohanta, 2011).

There are various types of neural networks, including feedforward neural networks (FNNs) for basic pattern recognition, convolutional neural networks (CNNs) for image and spatial data processing, recurrent neural networks (RNNs) for sequential data and time-series analysis, long short-term memory networks (LSTMs) designed to address RNN limitations for long-term dependencies, and transformer networks for handling complex attention-based tasks in natural language processing and beyond such as transformers, convolutional networks, multi-layer perceptron's, and recurrent neural networks. However, ANNs also come with notable disadvantages. Their training especially deep networks with many layers requires significant computational resources and can be time-consuming. Without proper regularization techniques they can overfit the training data, resulting in poor generalisation to unseen data.

The application of neural networks in industrial processes has been rapidly increasing. Figure 8 illustrates the percentage of neural network adoption from before 2005 up to 2016 and beyond. A significant increase is observed from 2016 onward, coinciding with the emergence of Industry 4.0, during which industries began adopting more advanced and technology-driven solutions.

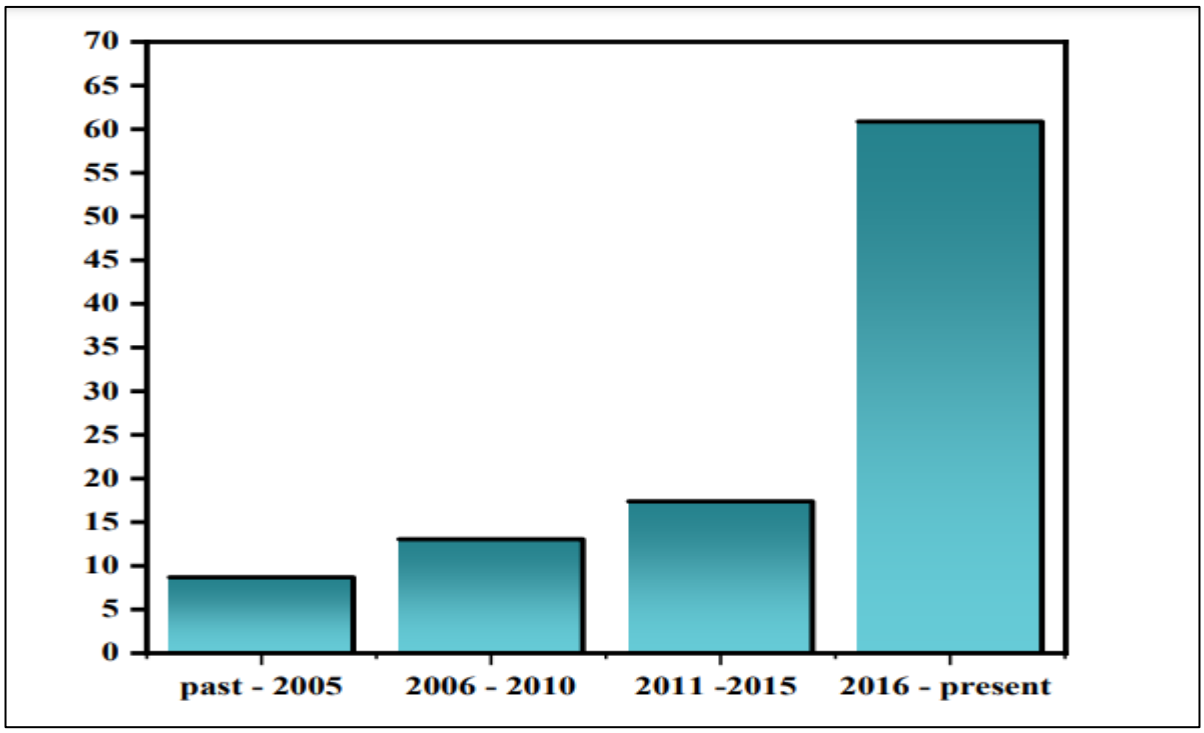


Figure 8: *ANN application in the industrial metallurgical field over the years (Zulu, Mvita and Thethwayo, no date)* .

Hyperparameter tuning is a crucial step in optimising the performance of artificial neural networks (Xiong *et al.*, 2024). It involves adjusting parameters that govern the learning process and architecture but are not directly learned from the data during training. Key hyperparameters include the learning rate which controls the speed at which the model updates weights. The number of hidden layers and neurons per layer which determine the network's capacity to learn complex representations. Finally, the batch size which influences the stability and efficiency of training (A Ilemobayo et al., 2024).

Proper tuning is essential because inappropriate hyperparameter values can lead to underfitting or overfitting which negatively impacts the model's generalisation to unseen data (Hinton, Srivastava and Swersky, 2012). Accurate tuning leads to faster convergence and enhanced accuracy by balancing the trade-offs between model complexity and overfitting. Techniques for hyperparameter optimisation include grid search, random search, and more advanced methods such as Bayesian optimisation, which aim to systematically explore the parameter space to find an optimal (Bergstra *et al.*, 2011). In summary the computational demands, risk of overfitting, and lack of interpretability are important considerations when applying ANNs in industrial settings.

2.3.4 k-Nearest Neighbours

k-Nearest Neighbours (KNN) is a versatile and widely used machine learning algorithm that can be applied to both classification and regression tasks (Yang, 2024). Unlike other algorithms, KNN stores the training dataset and uses it directly to make predictions, eliminating the need for a separate training step. This characteristic makes KNN a type of instance-based or lazy learning algorithm. It performs computations only when predictions are requested. The core idea behind KNN is to make predictions based on the similarity between the new data point and the k most similar instances in the training dataset.

The value of k which is a hyperparameter that specifies the number of neighbours to consider. In regression tasks the predicted value is typically the mean or median of the output values of the k nearest neighbours. For classification tasks, the predicted class is the mode of the classes of the k nearest neighbours. Similarity is measured using distance metrics, with Euclidean distance being the most popular for real-valued input variables (Prasetyo *et al.*, 2024). Other distance measures include the Hamming distance for binary vectors, Manhattan distance for real vectors, and Minkowski distance. These distance metrics are selected depending on the nature and characteristics of the data. They offer flexibility and enable the KNN model to better handle different data types and distributions, enhancing the model's performance for specific use cases (Cover and Hart, 1967).

The structure of this algorithm is relatively simple and is primarily built around three main components: the dataset, the distance metric, and the decision rule. Firstly, it stores the entire training dataset, consisting of input features and their associated labels or target values, without building an explicit model. During the prediction phase, a new data point is compared to every data point in the training set to find the k closest neighbours. The closeness or similarity is determined using a distance metric. These distances are calculated based on the feature values of the input variables. The selection of k is critical as a smaller value makes the model sensitive to noise, while a larger value smooths the predictions but may include less relevant neighbours (Prasetyo et al., 2024). After the nearest neighbours are identified, the algorithm applies a decision rule to be able to make a prediction.

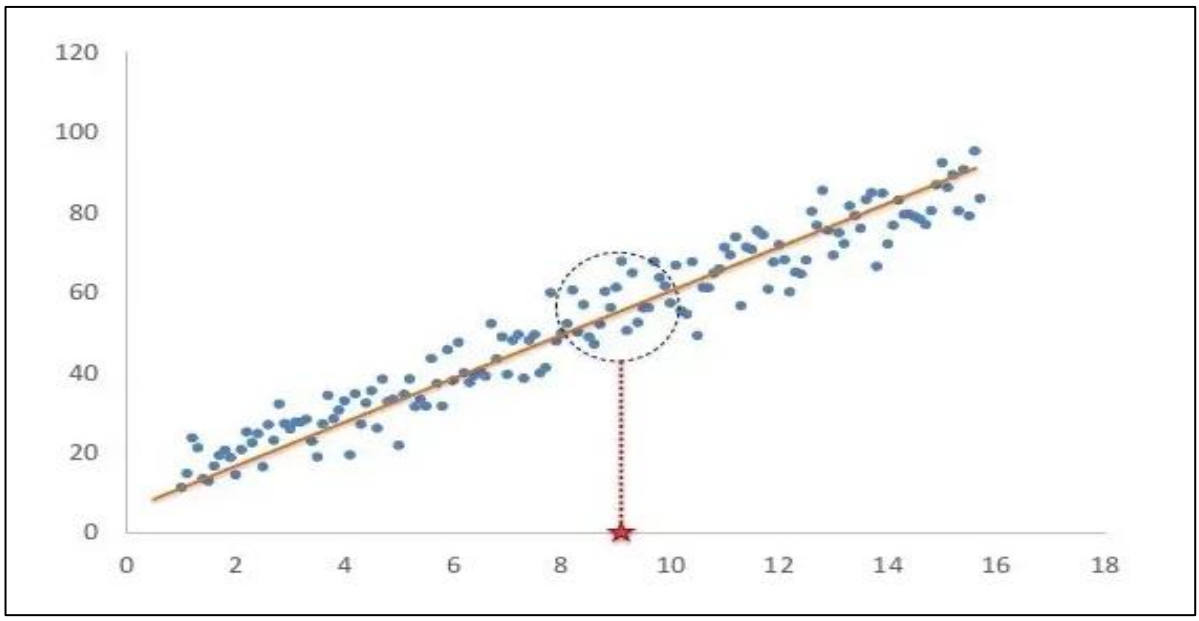


Figure 9: *The KNN models prediction from k nearest points (Singh, 2023)*

From the figure above, it can be observed that the KNN algorithm makes predictions based on the data points within the circle. The circle represents the set parameter value of K indicating the number of nearest neighbours used in the prediction process. One of the key advantages of KNN is its simplicity and ease of implementation. It requires minimal assumptions about the underlying data distribution and can adapt to complex, non-linear relationships in the data. However, the simplicity of KNN also comes with several limitations. The computational complexity of making predictions increases with the size of the training dataset, as the algorithm must calculate distances between the new data point and all training instances. Additionally, KNN can struggle with high-dimensional data due to the curse of dimensionality, where the volume of the input space grows exponentially with the number of dimensions, making distance measures less informative (Singh, 2023).

To mitigate some of these challenges, it is important to prepare the data appropriately before applying KNN. Rescaling the data so that all features have the same scale, such as normalizing values between 0 and 1 or standardizing to a Gaussian distribution, can improve performance. Handling missing data is also crucial, as missing values can disrupt distance calculations. Lowering dimensionality through feature selection or dimensionality reduction techniques can help KNN perform better on high-dimensional datasets. Despite its simplicity, KNN remains a powerful tool in the machine learning toolkit, especially for problems with low-dimensional data and clear distance-based relationships. Its interpretability and straightforward implementation make it an attractive choice for many applications, although careful consideration must be given to data preparation and parameter tuning to achieve optimal performance.

2.3. 5 Decision Trees and Ensemble Methods

Decision Trees, Random Forests, and XGBoost are essential machine learning algorithms widely used in regression tasks for industrial process control. Each algorithm offers unique advantages and employs different strategies to enhance predictive accuracy and robustness. Decision trees were the initial development in the family of tree-based algorithms, followed by the introduction of ensemble methods such as Random Forest and XGBoost, which offered enhanced robustness and improved performance over traditional decision trees. Decision Trees are simple, interpretable models that split data into subsets based on feature values, but they can overfit complex datasets. The limitations of decision tress led to the idea of ensemble algorithms that incorporates different strategies of bagging and boosting in order to ensure the decision trees overcome overfitting and to make them more robust.

Random forest is an extension of the bagging ensemble. It improves the decision trees performance by building multiple trees on different data subsets and aggregating their predictions, improving robustness and accuracy. XGBoost takes this further with advanced boosting techniques namely gradient boosting. In Gradient Boosting, models are built sequentially to minimize a loss function, with each new model correcting the residuals of previous ones. Other boosting techniques include Stochastic Boosting which introduces randomness by using random data subsets, enhancing generalisation and reducing overfitting. Adaptive Boosting (AdaBoost) improves model accuracy by adjusting the weights of misclassified data points in each iteration, focusing on difficult-to-predict instances. These boosting strategies collectively enhance the performance and reliability of regression models in industrial applications.

2.3.5.1 Decision Trees

These are simple yet powerful models that split the data into subsets based on feature values. Each node represents a feature, each branch represents a decision rule, and each leaf represents an outcome. Decision trees are highly interpretable, as the decision paths can be easily visualized and understood, making them valuable for industrial applications where model transparency is crucial. However, decision trees are prone to overfitting, especially with complex datasets, because they can become too closely tailored to the training data (Loh, 2014).

Decision Trees are a type of supervised learning model used for both classification and regression tasks. The basic structure of a decision tree resembles a flowchart where each internal node represents a "test" or decision based on a feature of the input data, each branch corresponds to the outcome of that test, and each leaf node represents a final decision or output (Quinlan, 1986). The model splits the dataset into subsets based on feature values, creating a hierarchy of

decisions that leads to a prediction. The process of building a tree involves selecting the optimal feature to split the data at each node to maximize the separation of classes or minimize prediction error as it can be seen in figure 10 the decision node splits the data into leaf nodes. Common splitting criteria include Gini impurity, Information Gain (using entropy), and Variance Reduction for regression tasks.

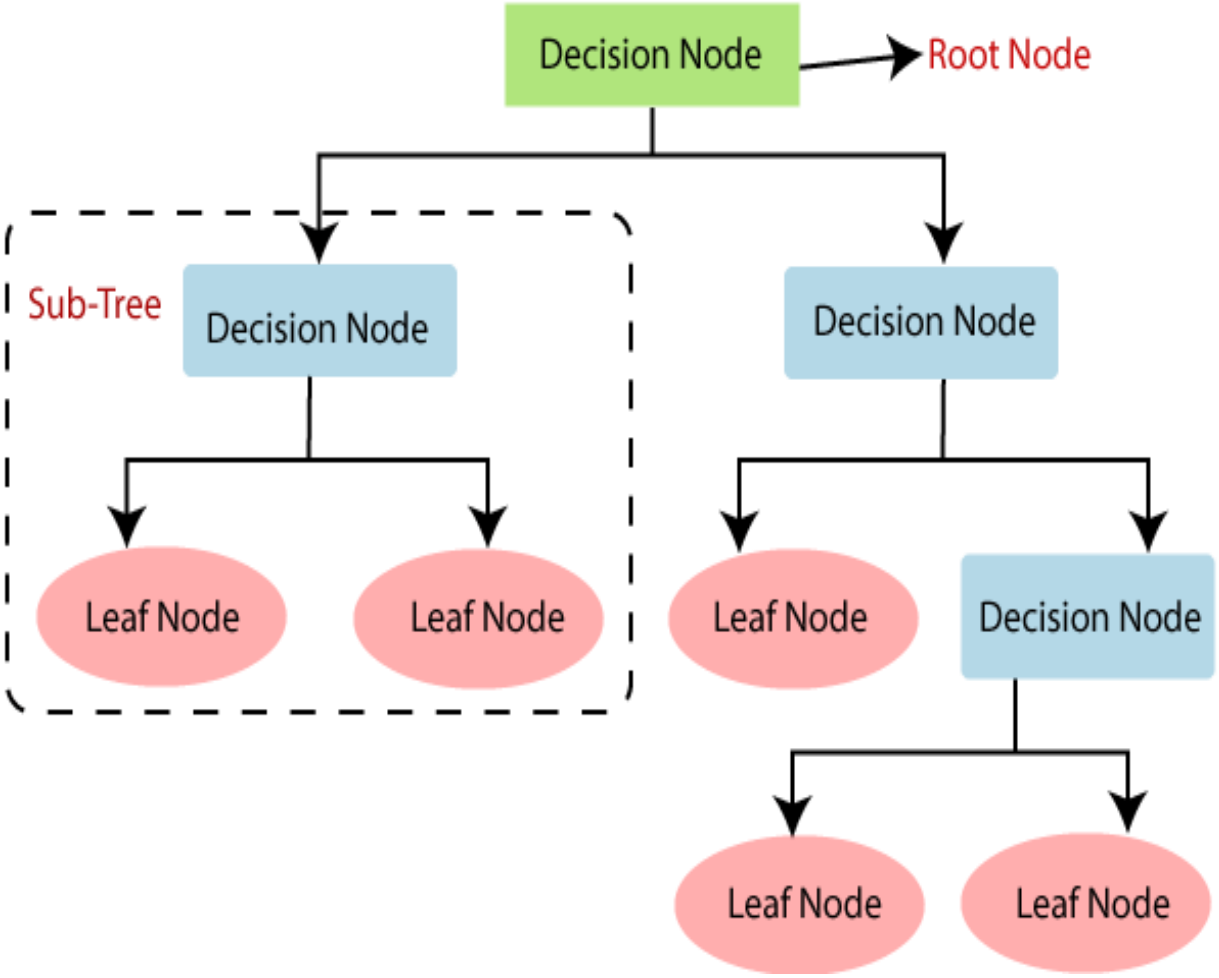


Figure 10: *Decision trees basic structure from root node to leaf node (Kumavat, 2024)*

The Decision Tree algorithm works by recursively partitioning the data to create a tree-like structure, where each node represents a decision point based on a feature, and the branches represent the outcomes of the decision. In predicting the class of a given dataset, the algorithm begins at the root node and compares the feature value at the node with the corresponding attribute value in the dataset. Depending on the result of this comparison it then follows the appropriate branch to the next node. This process repeats as the algorithm moves from node to node making decisions based on the attribute values until it reaches a leaf node which is responsible for the final prediction for that data point. The decision tree is built using the following steps:

Step 1: Start with the root node, which contains the entire dataset.

Step 2: Identify the best attribute to split the dataset using a measure like Information Gain, Gini Index, or Gain Ratio, known as Attribute Selection Measure (ASM).

Step 3: Divide the datasets into subsets based on the possible values of the best attribute.

Step 4: Create a decision node representing the best attribute identified and link its branches to the corresponding subsets.

Step 5: Recursively repeat steps 2 to 4 for each subset of the dataset, creating new decision nodes for each subset until all nodes can no longer be split further or a stopping criterion is met, such as all instances in a node having the same class. The nodes that cannot be split further are called leaf nodes, which provide the final output for predictions.

The decision tree algorithm continues this process until the tree is fully grown, and all branches end with a leaf node, representing the classification or value that the data point belongs to. The working principle of a decision tree is recursive partitioning, where the algorithm selects a feature that best divides the data into homogeneous subsets. It starts with the entire dataset at the root node, and based on a splitting criterion, partitions the data into branches and repeats this process for each subset. This continues until a stopping condition is met, such as reaching a maximum depth, a minimum number of samples in a node, or when further splitting does not improve the model's performance (Breiman, 2001). A well-constructed tree has leaves that are as pure as possible, meaning that the data in each leaf node belongs to the same class or has a low variance in case of regression. While decision trees are highly interpretable and easy to understand, they are prone to overfitting, especially if they grow deep and capture noise in the training data. Techniques like pruning, where sections of the tree are removed, or setting limits on tree depth are commonly used to prevent overfitting.

2.3.5.2 Random Forest

Random Forest (RF) is a versatile and powerful ensemble learning method that was first introduced by Breiman (2001b) . It emerged as an evolution of decision trees, designed to reduce their inherent weaknesses such as overfitting. By combining a multitude of decision trees and employing the technique of bootstrap aggregation or bagging, Random Forest builds a robust model capable of handling both classification and regression tasks. The method constructs each tree using a random subset of features and data samples, ensuring diversity among the individual trees, which leads to a more generalized model with reduced variance (Ho, 1995). Over time, enhancements in RF algorithms, including the integration of feature importance metrics and optimized decision-making strategies, have improved its performance, making it preferred for complex predictive tasks.

The core working principle of Random Forest regression lies in its ensemble structure. It builds multiple decision trees on different subsets of the data and aggregates their predictions to produce a final result. Each decision tree in the forest contributes to the final prediction by independently estimating the output for a given input. The algorithm aggregates these predictions by averaging them for regression tasks, resulting in a consensus output that tends to be more accurate and stable than any single tree. This process also known as bagging (Bootstrap Aggregating) helps to reduce the variance of the model which makes it more robust to noise and overfitting. The randomness introduced in the selection of training data subsets and features at each split creates an ensemble that is inherently more resistant to overfitting than individual decision trees (Liaw & Wiener, 2002).

This mechanism ensures that the RF model can handle high-dimensional spaces and interactions among variables effectively. A unique aspect of this model is the introduction of additional randomness. This arises from training each tree on a different subset of the data and considering a random subset of features at each node when making splits. This reduces the correlation between individual trees and enhances the model's generalisation capability. The ensemble approach of Random Forests enhances robustness and accuracy by averaging the predictions of multiple trees, which helps smooth out errors that might arise from relying on a single decision tree. This combined effect reduces overfitting while maintaining high model accuracy, as individual trees are free to capture different patterns within the data. According to Breiman (2001b), the capability of Random Forest models to handle extensive datasets with numerous input features allows for precise forecasting of mineral grades and recovery rates. This empowers mining engineers to optimize operational parameters, reduce costs, and improve overall output.

Several studies have demonstrated the successful application of Random Forest in the mineral industry for addressing challenges related to complex processes. For instance, in a study by Szmigiel et al., (2024) which focused on predicting flotation performance by analysing plant data that encompassed various operational and chemical variables. It was found that RF model can accurately capture complex relationships within the data contributing to improved predictive performance and process optimisation. Similarly, RF has been applied to real-time ore quality prediction in mining operations, which assists in decision-making related to ore blending and processing strategies. This results in higher throughput and more consistent product quality (Szmigiel *et al.*, 2024). In a separate application they were used for comminution circuit modelling to predict particle size distribution.

This application proved critical for downstream processing as it helped optimize milling operations and improve the overall efficiency of mineral processing plants (Le Roux, 2015). Identifying key factors that impact process efficiency such as reagent dosages, pH levels, and particle size distribution enables targeted process adjustments that lead to enhanced grade and recovery (Jung & Choi, 2021). The transparency of RF, when compared to more black box models like deep neural networks, supports its seamless integration into process control systems, where engineers and operators can make informed decisions with greater confidence (Rodrigues et al., 2020). Furthermore, its use to predictive maintenance and failure analysis within the industry contributes to operational cost savings and increased plant productivity. From analysing historical sensor data and operational logs they can detect patterns that precede machinery failures, allowing for proactive measures that minimize downtime and extend equipment lifespan (Singh & Vazirani, 2022).

Random Forests has several hyperparameters that play a critical role in influencing their performance and robustness by shaping the model's structure. One of the primary hyperparameters is n_estimators, which specifies the number of decision trees in the forest. Increasing this number typically improves performance as it enhances the ensemble's predictive strength, but it also increases computational cost (Breiman, 2001b). The max_depth parameter sets a limit on the depth of each tree. This helps balance model complexity, preventing overfitting while ensuring that essential data patterns are captured. The min_samples_split and min_samples_leaf parameters are essential for controlling the minimum number of samples needed to split a node and to define a leaf, respectively. These parameters significantly impact the model’s ability to generalize effectively. The max_features parameter specifies the number of features to consider when splitting at each node. This introduces randomness, which helps to reduce correlation between the trees and promotes greater model diversity. The bootstrap parameter controls whether sampling is performed with replacement. When enabled, it allows individual trees to be trained on different subsets of the data, increasing model variability and enhancing robustness. The criterion parameter for split quality e.g., gini, entropy, mse, or mae influences model performance by determining how the decision at each node split is made. Meanwhile, the max_samples parameter sets the sample size used when bootstrapping, balancing computational efficiency and performance. Additionally, the max_leaf_nodes parameter can limit the number of leaf nodes, further reducing the risk of overfitting by simplifying the model structure. Effectively tuning these hyperparameters strikes a balance between accuracy, computational efficiency, and model generalisation, ultimately leading to improved performance of Random Forests (Hastie et al., 2009).

2.3.5.3 Extreme Gradient Boosting

This family of algorithms is also known as the XGBoost which represents a more advanced form of boosting applied to decision trees. Unlike random forests, which build trees independently, XGBoost builds trees sequentially, with each new tree attempting to correct the errors of the previous ones. XGBoost incorporates several enhancements over traditional gradient boosting, including regularization to prevent overfitting, parallel processing to improve computational efficiency and sophisticated handling of sparse data. It supports various boosting strategies, such as gradient boosting, which focuses on minimizing a loss function through iterative improvement. XGBoost's ability to handle large-scale and high-dimensional data, coupled with its robustness to overfitting, makes it a preferred choice for many regression tasks in industrial processes (Chen & Guestrin, 2016).

Extreme Gradient Boosting (XGBoost) is a highly efficient and scalable implementation of gradient boosting algorithms, first introduced by Tianqi Chen in 2014 as part of his research at the University of Washington (Chen & Guestrin, 2016). The development of XGBoost was driven by the need for a faster, more resource-efficient version of traditional gradient boosting methods, particularly for handling large-scale data and complex predictive tasks. Over time, XGBoost has evolved to include advanced regularization techniques and parallel processing capabilities, making it a preferred choice in both academia and industry for its superior performance and flexibility.

The working principle of XGBoost builds on traditional boosting algorithms by training models sequentially, where each new model attempts to correct the errors of its predecessor. This is evidence in figure 11 where trees up to nth trees are created and their predictions are summed to give the final prediction. However, XGBoost incorporates several enhancements that set it apart, such as a highly optimized algorithm for tree pruning, built-in regularization (L1 and L2), and the ability to handle sparse data effectively. It also introduces techniques like weighted quantile sketch to handle large datasets efficiently. This structure enables XGBoost to construct robust models that minimize bias and variance, yielding predictions that are both accurate and generalizable (Chen & Guestrin, 2016). The diagram illustrates an ensemble learning approach where multiple decision trees are trained independently. Each tree generates a separate prediction based on input data. These individual predictions are then aggregated. The combined result produces a more accurate and robust final prediction.

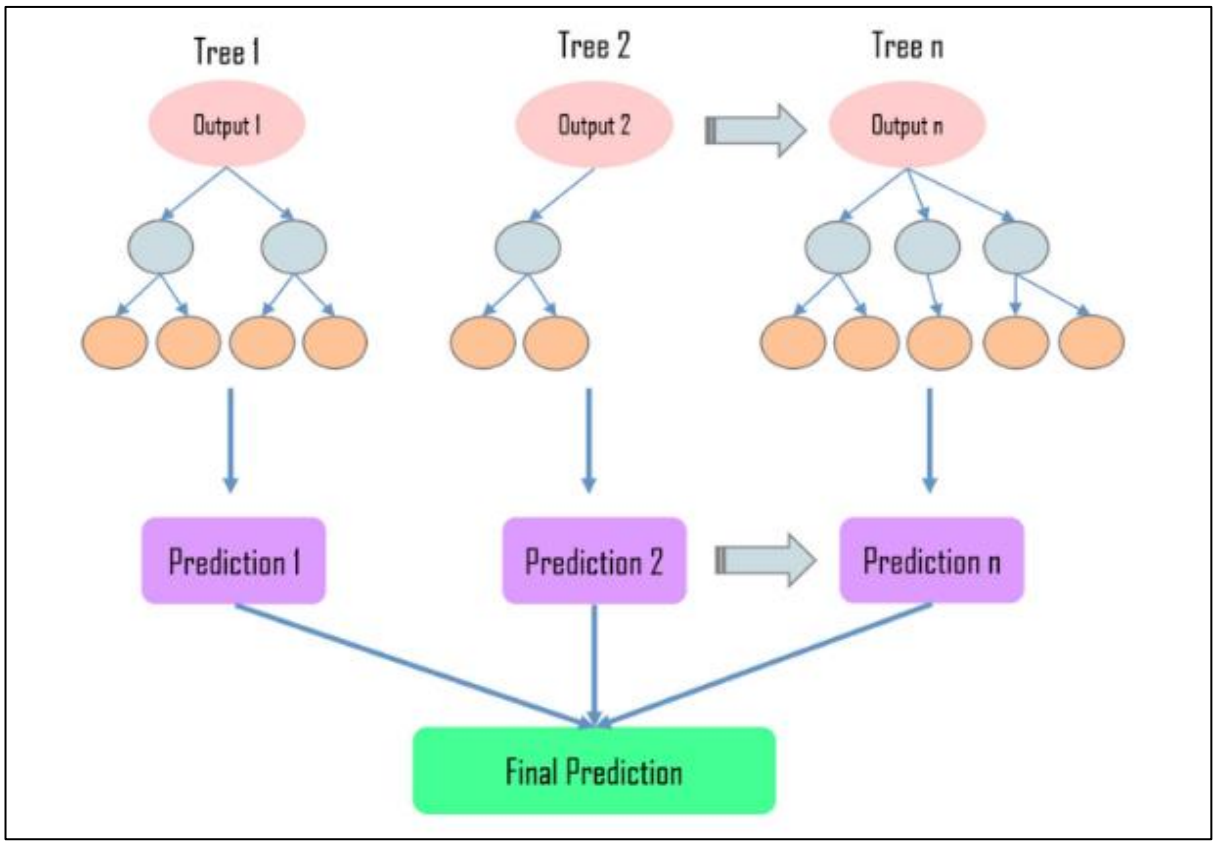


Figure 11: *Basic structure of boosted decision trees* (Jumin *et al.*, 2020).

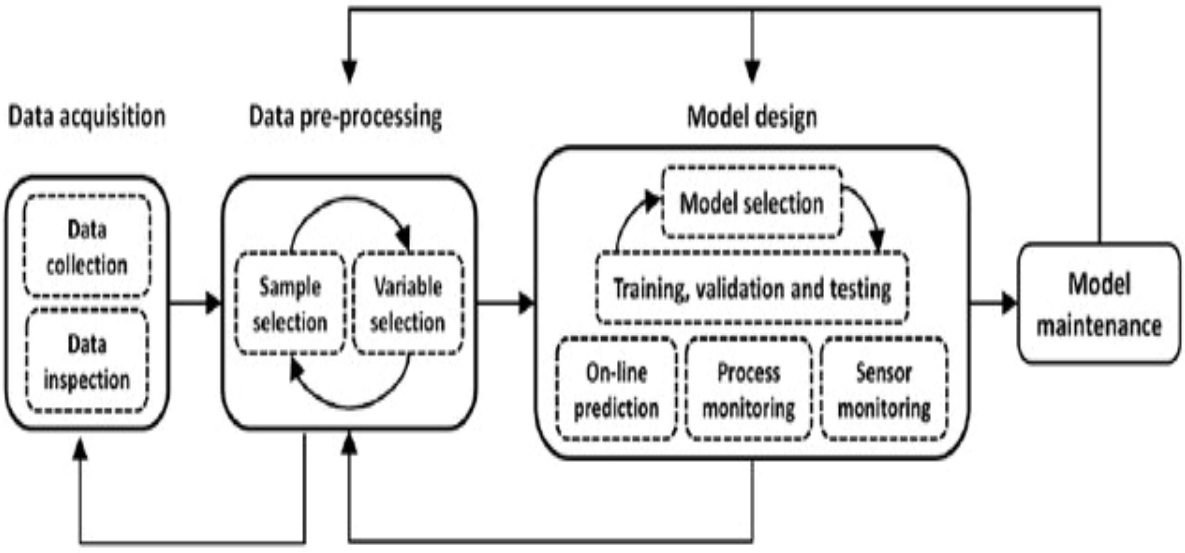
XGBoost has found valuable applications in the mineral industry, particularly in modelling and predicting flotation performance and mineral grade recovery. In processing plants where multiple variables interact in nonlinear and complex ways, XGBoost's capability to manage high-dimensional data makes it a reliable tool for forecasting outcomes and optimising operational parameters. For instance, its application in predicting the recovery rates and grades of minerals in flotation processes allows engineers to fine-tune process conditions and improve yield, thereby reducing waste and operational costs (Haque & Lu, 2017). The integration of XGBoost into mineral processing has also facilitated better decision-making for real-time adjustments, enhancing overall plant efficiency and productivity.

Despite its advantages, XGBoost does come with its own set of pros and cons. On the positive side, XGBoost is known for its high accuracy, efficient handling of large datasets, and robust regularization capabilities that prevent overfitting. Its flexibility to work with different loss functions and custom evaluation metrics adds to its appeal. However, the computational cost and memory usage can be significant, particularly for extremely large datasets. Furthermore, XGBoost requires careful hyperparameter tuning to achieve optimal performance. Fine-tuning these parameters helps achieve a balance between model accuracy, training speed, and generalisation (Chen & Guestrin, 2016).

Hyperparameter tuning in XGBoost is crucial for optimising model performance and ensuring robust results. The model's performance is significantly influenced by several key hyperparameters. The n_estimators determines the number of boosting rounds; increasing this value can enhance accuracy but may also increase training time and the risk of overfitting (Chen & Guestrin, 2016). Max_depth sets the maximum depth of each tree, balancing model complexity to capture essential data patterns without overfitting. The learning_rate (or eta) adjusts the contribution of each tree to the overall model. While lower values of learning_rate improve generalisation, they may require more boosting rounds to achieve optimal results.

The subsample parameter defines the fraction of training data used for each boosting iteration, adding randomness that helps prevent overfitting and enhances model robustness. colsample_bytree specifies the fraction of features sampled for building each tree, promoting diversity in feature selection and reducing dependency on specific features. The gamma parameter sets the minimum loss reduction required for a node to split, acting as a regularization measure to avoid unnecessary splits. Regularization parameters such as reg_alpha (L1 regularization) and reg_lambda (L2 regularization) help control model complexity and reduce overfitting. random_state ensures reproducibility by setting a seed for the random number generator, and n_jobs allows parallel processing, accelerating training on large datasets. Effective tuning of these hyperparameters helps practitioners strike a balance between model accuracy, computational efficiency, and generalisation, leading to enhanced XGBoost performance (Chen & Guestrin, 2016).

2.4 Soft Sensor Development



The figure 12 outlines the typical process for soft sensor development, integrating data-driven modelling with expert knowledge to ensure accuracy and robustness. Key stages include data collection where raw data is gathered from sensors or processes followed by data preprocessing which involves cleaning and transforming data for analysis. After the data is cleaned model selection and training then follows which is where suitable algorithms are applied to assess their predictive power. The model validation phase then ensures that the model's performance aligns with the desired standards before deployment as a soft sensor for real-time prediction. Continuous soft sensor maintenance ensures reliability by updating the model as system dynamics evolve. Expert knowledge is integrated throughout the process supporting preprocessing, model refinement, and maintenance.

This systematic workflow aligns with established methodologies in soft sensor development, emphasizing the iterative nature of data-driven and knowledge-based approaches (Fortuna, et al., 2007). By combining domain expertise with machine learning techniques, soft sensors enhance process monitoring and control, particularly in industries where physical sensors are infeasible or expensive (Kadlec et al., 2009). The feedback loop from the output to earlier stages further ensures continuous improvement and adaptability of the model.

2.4.1. Data collection

The success or failure of a soft sensor largely depends on the effective preprocessing of process data extracted from a plant's database. For this reason, it is vital that industries store as much historical production data as possible. This practice offers several advantages, such as enabling optimisation strategies and facilitating fault detection through data analysis. Technological advancements have made it easier for many industries to adopt comprehensive data storage solutions for historical data. For instance, Anglo Platinum’s operations in particular at Mogalakwena North’s rougher banks, exemplify an advanced approach to storing vast amounts of process data.

Before beginning data collection, it is essential to verify that the correct data tags are being used. Units and process expertise are typically used to ensure the accuracy and validity of the data collected. Software tools such as Python, RapidMiner, and MATLAB are commonly employed to assess the availability and quality of the data. These offer scatter plots, histograms, bar charts, and other basic visualizations which are vital in checking the reliability of the data.

The volume and accuracy of this data are critical, as sensors that capture real-time data during plant operations are often prone to failure or erroneous readings due to a variety of factors, including sensor degradation and environmental conditions (Ye *et al.*, 2020). Moreover, robust data storage and preprocessing methods can significantly enhance model accuracy and predictive capabilities in soft sensor applications, leading to improved process control and efficiency (Souza, et al., 2016). The storage of historical data offers numerous benefits, particularly in industrial settings where process optimisation, fault detection, and predictive maintenance are critical. Historical data allows industries to identify long-term trends, enabling more informed decision-making and more accurate predictions.

This data can be used to improve the accuracy of models by providing a wealth of information for training machine learning algorithms, thereby enhancing the performance of soft sensors and other predictive tools. For example, industries such as oil and gas, chemical manufacturing, and power generation commonly store vast amounts of historical data to improve operational efficiency and reduce downtime (Mohammadpoor & Torabi, 2020). Furthermore, historical data can significantly aid research by providing a valuable resource for developing new techniques and technologies, such as anomaly detection algorithms and optimisation strategies (Fisher *et al.*, 2022).

Industries now have access to a variety of data storage techniques that uses different technologies to handle the massive volumes of data generated by modern processes. Cloud storage platforms such as AWS and Microsoft Azure offer scalable solutions that enable real-time data access and long-term archiving. On-premises storage systems like distributed databases and data lakes are also widely used, particularly in industries where data security and control are paramount (Hashem *et al.*, 2015). Advanced data compression techniques such as lossless compression and efficient indexing methods further enhance storage efficiency making it easier to retrieve and analyse historical data quickly. These storage practices combined with rigorous data preprocessing ensure that industries can leverage their historical data to drive continuous improvement, innovation, and competitiveness in their respective fields.

2.4.2 Data Pre-processing

2.4.2.1 Sampling time

To address the issue commonly referred to as the "multiple rate phenomenon," which pertains to variations in sampling time when dealing with rapidly changing inputs and hence innovative strategies will be explored. All these methods aim to find a way to adjust the time on a graph so that the data from different sources line up properly. Imagine having data from various sources,

but the timestamps of the collected data do not align. These methods help create a function that rearranges the time points on the graph resulting in new synchronized data where each point on the graph corresponds to a specific time according to the function.

Sampling time is a crucial factor in data preprocessing, particularly for industrial processes where real-time data collection and monitoring are integral for process optimisation and control. The selection of an optimal sampling time is critical as it influences the accuracy of data collection and the reliability of predictive models. Short sampling times can lead to data overload and increased noise, while long sampling intervals risk missing critical transient events, reducing the ability to detect faults or anomalies (Shardt & Huang, 2011). Proper sampling ensures that the data accurately reflects the process dynamics and prevents computational inefficiency.

Research highlights the need to balance between capturing essential process variations and maintaining manageable data volumes. In continuous processes, particularly those with time delays between sensors, aligning sensor data based on sampling time is essential for accurate process monitoring. Shardt & Huang (2011) demonstrated that optimal sampling times are particularly important for systems with high time delays where inadequate sampling can result in inaccurate models of system dynamics. Additionally, in systems that require high-speed sequential sampling such as those studied by Hildebran (2007), appropriate sampling intervals ensure the integrity of the collected data by reducing latency between measurements. Techniques such as adaptive sampling are becoming increasingly popular that allows systems to dynamically adjust the sampling interval based on current process conditions which improves efficiency and data quality (Cattaldo et al. 2024). These innovations not only streamline data collection but also enhance the performance of machine learning algorithms used in industrial process control, anomaly detection, and optimisation.

2.4.1.2. Outliers

Outliers pose a significant challenge in data preprocessing, as they can distort model accuracy and lead to false-positive detections in algorithms. Several approaches to detecting and managing outliers have been proposed in the literature with each having its own strengths and weaknesses that normally depends on the context and data characteristics. The first step in mitigating the effect of outliers is to accurately identify them and followed by the appropriate method of correction or removal (François-Lavet *et al.*, 2018). Di Bella et al., (2007) compared various outlier detection methods and found that Principal Component Analysis (PCA) yielded the best results for a sulfur recovery unit which demonstrated its efficiency in dealing with complex industrial processes where outliers are not easily detected and identified.

Maximum likelihood estimation is another widely used method for handling outliers, particularly in cases where missing data are present. Walczak & Massart (2001) explained that this approach estimates model parameters by maximizing the probability of the observed data, which not only addresses outliers but also accounts for missing values. The Expectation-Maximization (EM) algorithm builds on this principle by iteratively refining estimates for missing data and model parameters, improving data integrity with each iteration until convergence (Souza et al. 2016). Bayesian multiple imputation offers an alternative by incorporating probabilistic assumptions about the missing data, thereby improving robustness and reducing bias in the data set.

Another effective method is hot-deck imputation, which replaces missing values with randomly selected values from similar records, ensuring the imputed data remains consistent with observed patterns (Andridge & Little, 2010). Manual inspection during data visualization plays an equally important role in identifying outlier masking (undetected outliers) and outlier swamping (correct values mislabelled as outliers). To minimize false negatives, methods such as the 3σ rule identifier (Davies & Gather, 1993), Mahalanobis distance, and PCA or partial least squares (PLS) in conjunction with Jolliffe parameters are commonly employed (Di Bella *et al.*, 2007). The selection of the most suitable approach depends on how adaptive and robust the method is when applied to new, unseen data. In this context, the outlier detection method will be validated using a 20% test data sample that has been separated from the original data. This ensures that the chosen method performs well on both the training and testing data, maintaining its effectiveness in practical applications.

2.4.1.3 Missing data

Missing data is a common challenge in data preprocessing, and various algorithms have been developed to address this issue effectively. Researchers and data processing experts typically apply two primary approaches to manage missing data: deletion-based methods and imputation techniques. The traditional method, known as listwise deletion, involves removing any samples or rows that contain missing data (NaN or zero values). While this approach is simple and ensures the integrity of the remaining data, it can lead to significant data loss, particularly when a large portion of the dataset is affected, making it less feasible in scenarios with small datasets (Little and Rubin, 2019).The second approach, imputation, is a more refined strategy that fills in the missing values using statistical methods, such as replacing them with the mean, median, or mode of the available data.

This method retains more data for analysis, which is particularly important in smaller datasets where deletion could result in losing valuable information. According to Schafer and Graham

(2002), imputation methods like mean or median substitution are effective for datasets with minimal missing data and relatively normal distributions. However, more advanced methods like multiple imputation or k-nearest neighbours (KNN) can be used to handle more complex missing data patterns and maintain data variability (Andridge & Little, 2010). In this study, the initial step will involve identifying and assessing the missing data through listwise deletion to detect any patterns or variables that consistently contain missing values. After this preliminary step, careful evaluation of the missing data will determine if any vital process parameters are at risk of deletion, thereby guiding the decision to apply imputation methods. Given the size of the dataset in this case, more emphasis will be placed on imputation techniques to preserve as much data as possible, ensuring that the final dataset retains its integrity for robust analysis.

2.4.2.4 Input variables selection and feature extraction

Fewer variables offer several advantages, such as faster model creation and training time. It also offers the potential for better understanding the insights into the process's physical interpretation that aids in enhanced model accuracy (Souza et al., 2016). Traditional way of selecting the input variables involved process engineers and experts to visualize and interpret the data to find the relevant input variables. This approach is not feasible when there is a high amount of data with many variables that contribute to the target output. Research underscores the value of integrating domain expertise with automated feature selection in industrial machine learning (ML). The principal component analysis (PCA) under unsupervised variable selection is one method used to automatically select the relevant variables. In executing this task, the PCA algorithm takes the input data and transform it to a latent space. Here the initial latent variable (known as the principal component) captures the most variance, and each subsequent component maintains the highest possible variance while being independent from the previous ones.

Supervised approaches that are used for variable selection are divided into three categories based on the criteria they use to select variables. This technique includes filter, wrapper and embedded variable selection method. Filter variable selection methods assess the quality of a feature subset using statistical metrics like correlation coefficient (CC) and mutual information (MI), quantifying their relevance. These methods operate without reliance on the specific model being employed allowing quick evaluation without relying on a model. However, this can overlook complex feature interactions. Wrapper criteria determine feature subset quality by evaluating model performance using metrics like mean square error (MSE), Akaike information criterion (AIC), or Cp statistics. They rely on the model's effectiveness rather than predefined statistical measures providing higher accuracy but with higher computational costs (Kohavi & John, 1997).

These strategies are typically divided into two types, namely forward selection and backward elimination (Guyon & Andre, 2003). In forward selection variables are added incrementally to build increasingly larger subsets whereas in backward elimination begins with the full set of variables the least relevant variables are removed step-by-step (Kohavi & John, 1997). Embedded methods establish their feature selection criteria based on inherent model characteristics or the process of model learning, examples being pruning techniques and regularization methods. Least Absolute Shrinkage and Selection Operator (LASSO) and Elastic Net are examples of the embedded approach as they incorporate feature selection into the model construction process itself striking a balance between computational speed and prediction accuracy (Zou & Hastie, 2005). Hybrid methods, which combine different feature selection strategies, are increasingly popular. For instance, blending filter and wrapper techniques has been shown to increase the stability of variable selection while maintaining system performance, making it an effective approach for industrial applications (Cateni & Colla, 2018).

Researchers frequently adopt the black-box approach in soft sensor design, as it is widely accepted for its practical effectiveness (Kadlec et al. 2009). Certain variable selection algorithms also use this black-box approach, where the specific method of variable selection remains unclear (Fortuna *et al.*, 2007). These algorithms require a predetermined threshold indicating the number of variables to be selected. This threshold is typically derived from prior research, process knowledge, and iterative trial-and-error adjustments. Numerous variable selection techniques are now available (Leiria *et al.*, 2023). This enables researchers and designers to explore and compare different methodologies to identify the most effective subset of variables for soft sensor development. These comparative analyses help optimize sensor performance by pinpointing the most relevant variables, which enhances prediction accuracy and reduces computational complexity (Fortuna et al., 2007).

A variety of methods have been widely employed for selecting a relevant subset of input variables in data-driven modelling. These include traditional statistical approaches such as correlation analysis, Mallow's Cp statistics, and Lipschitz quotients, as well as multivariate techniques like Partial Least Squares (PLS). More advanced and feature selection-specific methods include the Minimum Redundancy Maximum Relevance (mRMR) principle and the non-parametric k-Nearest Neighbour algorithm (Peng, Long, and Ding, 2005). Additionally, entropy-based techniques such as histogram-based estimators, and heuristic search algorithms like Genetic Algorithms (GA) have been utilized (Estévez et al., 2009). Other notable methods include the Normalized Mutual Information Feature Selection (NMIFS) (Estévez et al., 2009),

the Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996), and the SBS-MLP algorithm developed by Wang, Lu, and Qin (2020). In practice multiple models are tested in selecting the variables. The historical data is fed to each model so that the model's performance can be compared and the one that has high accuracy will be chosen.

2.4.3. Model choice and training

After completing data preprocessing, splitting the dataset into training and testing sets is an essential step for evaluating model performance. Commonly applied splits are 80% for training and 20% for testing, or alternatively 75% and 25%. This allocation ensures the model has sufficient data for learning from a diverse historical range while preserving an unseen portion for testing, which is critical for assessing the model's generalisation capabilities (Hastie, Tibshirani and Friedman, 2009). Sequential splitting, which maintains the temporal order of observations, is particularly beneficial for process data as it preserves the chronological sequence, crucial for time-dependent industrial processes (Brownlee, 2016).

The choice of programming environment can impact the efficiency of model development. MATLAB is frequently utilized due to its user-friendly interface and comprehensive library of built-in machine learning functions, allowing for rapid prototyping without extensive coding. This can save valuable time, especially when handling complex industrial data. On the other hand, Python has become a dominant platform for machine learning tasks, offering extensive libraries such as scikit-learn and TensorFlow, which are powerful for both machine learning and deep learning applications (Pedregosa et al., 2011).

These environments provide the flexibility to implement models that range from simpler algorithms like Principal Component Regression (PCR) and Partial Least Squares to more complex methods like Support Vector Machines (SVM) and Artificial Neural Networks (ANN). These are adept at capturing nonlinear relationships and managing high-dimensional data, essential for robust soft sensor development (Corrigan & Zhang, 2021). These models are chosen for their versatility in handling both linear and nonlinear systems, which is critical in soft sensor applications. For more complex and nonlinear data, advanced models like Support Vector Machines (SVM), Extreme Learning Machines (ELM), Artificial Neural Networks (ANN), and Neuro-Fuzzy Systems (NFS) will be tested. These methods are well-suited for capturing nonlinear relationships and high-dimensional feature spaces, particularly in noisy industrial environments (Corrigan & Zhang, 2021).

Model training is an iterative process that involves adjusting model parameters to minimize an objective function and achieve high predictive performance. Two critical metrics recorded during this process are training time and model accuracy. These metrics are vital for comparing models and selecting the most suitable one for deployment. Techniques like fine-tuning, parameter optimisation, and regularization are employed to balance bias and variance, ensuring that the model does not underfit or overfit (Goodfellow et al., 2016). Regularization methods, including L1 and L2 penalties which help constrain model complexity by penalizing large parameter values, while techniques like dropout add noise to the model during training, which improves robustness and reduces overfitting (Hinton et al., 2012).

Continuous monitoring during the training process is essential for detecting high bias (underfitting) or high variance (overfitting) which can compromise generalisation. Cross-validation techniques, such as k-fold cross-validation, provide robust estimates of model performance by ensuring that all data points contribute to both training and validation phases. This method reduces selection bias and offers a more reliable evaluation of how the model will perform on unseen data (Hastie et al., 2009). Monitoring tools and iterative optimisation strategies in environments like MATLAB and Python facilitate real-time adjustments, aiding in achieving a model that is not only accurate during training but also generalizes effectively when deployed (MATLAB Documentation, 2020).

2.4.4. Model validation

Model validation is a crucial step in developing reliable and effective data-driven soft sensors for predicting grade and recovery in flotation circuits. This process ensures that predictive models perform consistently well when exposed to new, unseen data, thereby confirming their robustness and applicability in real-world settings. Good generalisation in a model is crucial in ensuring that the model does not merely memorize the training data but can also predict outcomes accurately when presented with new information (Hastiez et al., 2009). According to Dietterich (1995) employing robust validation techniques is essential to prevent overfitting, which occurs when a model performs exceptionally on training data but fails to generalize to practical applications. Techniques such as k-fold cross-validation and hold-out validation are widely used to ensure comprehensive model evaluation. These strategies divide the dataset in a manner that ensures variability in training and testing, providing an accurate assessment of the model's performance under different operational conditions (Hastie et al., 2009).

The use of external validation datasets distinct from the training and testing data is another highly recommended practice. This step is essential for accurately evaluating the model's real-world performance, particularly in the context of flotation circuits where conditions such as ore mineralogy, reagent types, and equipment configurations vary continuously. According to Mitchell & Mitchell (1997) these external validation helps to confirm a model's adaptability and robustness in varying operational scenarios, providing a more realistic measure of its readiness for deployment. Ensuring that the model is tested against diverse datasets helps improve its reliability in complex industrial settings.

Selecting suitable performance metrics is another vital aspect of model validation. Mean Squared Error (MSE) is one of the most widely used and traditional metrics for evaluating model performance, particularly in regression tasks, as it emphasizes larger errors by squaring the differences between predicted and actual values (Brownlee, 2016). Currently the most utilised in industrial settings include Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared. Each metric highlights different facets of model performance. For instance, RMSE is especially valuable for penalizing larger errors which is crucial when significant deviations can impact operational efficiency (Goodfellow, Bengio and Courville, 2016).

Conversely, MAE provides a balanced view by treating all errors equally, making it useful for models intended for routine monitoring (Brownlee, 2016). It is particularly useful when the goal is to understand the typical size of the prediction errors making it an important metric for assessing the model's reliability (Goodfellow, Bengio and Courville, 2016). Root Mean Squared Error (RMSE) and Normalized Root Mean Squared Error (NRMSE) are preferred for their ease of interpretation, as they express the error in the same units as the data which makes it easy to understand and interpret. NRMSE is particularly valuable as it expresses the error as a percentage, allowing for a normalized comparison across different models or datasets(Brownlee, 2016). The choice of validation metrics should align with the specific objectives of the flotation circuit's operational framework to ensure that model outputs support effective process control and decision-making (Hastie, Tibshirani and Friedman, 2009).

Real-time validation and adaptive retraining are increasingly recognized as critical strategies for maintaining long-term model accuracy and reliability. Variations in process conditions, changes in feed characteristics, and equipment wear can lead to model drift, resulting in reduced predictive accuracy over time (Widodo & Yang, 2007). Continuous validation and retraining, as suggested by Pan & Yang (2010), are effective in addressing this issue. These strategies involve periodically re-validating and updating the model with new data to maintain its robustness and

predictive performance. Implementing adaptive validation and retraining strategies is essential for maintaining the reliability of soft sensors in dynamic industrial environments. Such approaches ensure that predictive models remain aligned with evolving process conditions, there by supporting continuous process optimisation and informed decision-making. For instance, Kubosawa et al. (2022) discussed the integration of artificial intelligence with dynamic simulation to enhance process control. They highlighted the importance of adaptive methodologies in accommodating unrecorded situations and maintaining model accuracy over time.

2.4.5. Model maintenance

Model maintenance is an integral part of deploying machine learning models in real-world applications, ensuring that they remain effective and relevant over time. This is important because these models are developed with historical data represents a limited time frame and a constrained range of operational states. This limitation can result in performance degradation as new process conditions and states emerge. Once a model is developed and validated, continuous monitoring and maintenance are essential to preserve its predictive performance in dynamic environments. According to Raschka (2018), models may degrade in accuracy over time due to factors such as data drift, changes in process conditions, or the introduction of new variables that were not accounted for during initial training.

Regular assessments of model performance using fresh data can help detect issues early and prevent potential failures in predictive tasks. To avoid obsolescence, the soft sensor must adapt to new data and changing conditions. Research has demonstrated that integrating expert knowledge of process dynamics to guide the selection of adaptive model parameters enhances the accuracy of adaptation. Key attributes such as moving window sizes and forgetting factors help the model respond to process changes while discarding outdated information (Kadlec et al., 2009). Moving window techniques are particularly effective for maintaining model performance in online settings, where real-time data updates are critical. These techniques use only the most recent data within a predefined window for model updates, ensuring that the soft sensor adapts to changing process conditions without relying on outdated information.

Block-wise moving windows allows models to update in manageable chunks while balancing computational efficiency with timely performance adaptation. This method proves advantageous for handling large datasets and high computational loads. Principal Component Analysis (PCA) is another valuable approach for maintaining adaptability. Introduced by Wang et al. (2005), this

method recalculates principal components in real-time, allowing the model to track dynamic changes effectively. This real-time recalibration ensures that the soft sensor can reflect current process trends and maintain its predictive accuracy as the operational environment evolves.

For more complex and non-linear systems, moving window kernel PCA can be employed to capture higher-order variable correlations. As discussed by Liu et al. (2009), this method enhances the model's capability to handle non-linear relationships in the data, ensuring robustness in diverse and evolving process conditions. Such adaptive approaches ensure that soft sensors remain resilient and computationally efficient, balancing sensitivity to new data with stability over time. By continuously fine-tuning attributes like window size and the forgetting factor, soft sensors can maintain their operational reliability, supporting sustained process optimisation and informed decision-making even as process dynamics change.

An essential aspect of model maintenance is the implementation of performance tracking systems. These systems monitor real-time KPIs, such as recent error trends or model confidence levels, to approximate and maintain prediction performance. By setting thresholds for acceptable performance, organizations can automate the process of triggering alerts or initiating retraining when model performance falls below the set standards. This proactive approach ensures that the deployed models continue to deliver reliable results and align with evolving business or operational needs (Hastie et al., 2009).

Finally, maintaining a clear documentation and version control of model updates is essential for reproducibility and transparency. This practice helps in understanding the changes made to the model over time and provides a basis for evaluating the impact of each update. Proper version control facilitates collaboration among teams and supports audits or compliance checks where a historical view of model performance and modifications is necessary (Goodfellow, Bengio and Courville, 2016). Ensuring robust model maintenance practices enables models to stay aligned with the objectives they were designed to achieve, ultimately supporting sustained process optimisation and effective decision-making.

Chapter 3 Process Description

3.1 Platinum Concentrator Process Description

The Platinum Concentrator Process is a sophisticated operation that plays a crucial role in the extraction and purification of platinum group metals (PGMs). This process involves several stages each critical to the overall efficiency and effectiveness of the mineral processing as it can be seen in figure 13 below. The diagram below provides an overview of the PGMs processing value chain from mining to final product fabrication. It begins with ore extraction followed by crushing, milling, and froth flotation to concentrate PGMs while generating tailings. The concentrate is then smelted in an electrical furnace, where sulphur is removed to produce converter matte. In the refining stage, base metals like nickel, copper, and cobalt are separated before PGMs and gold undergo final purification. The refined metals are then used in various applications, including auto catalysts, jewellery, and industrial products like electronics and fuel cells.

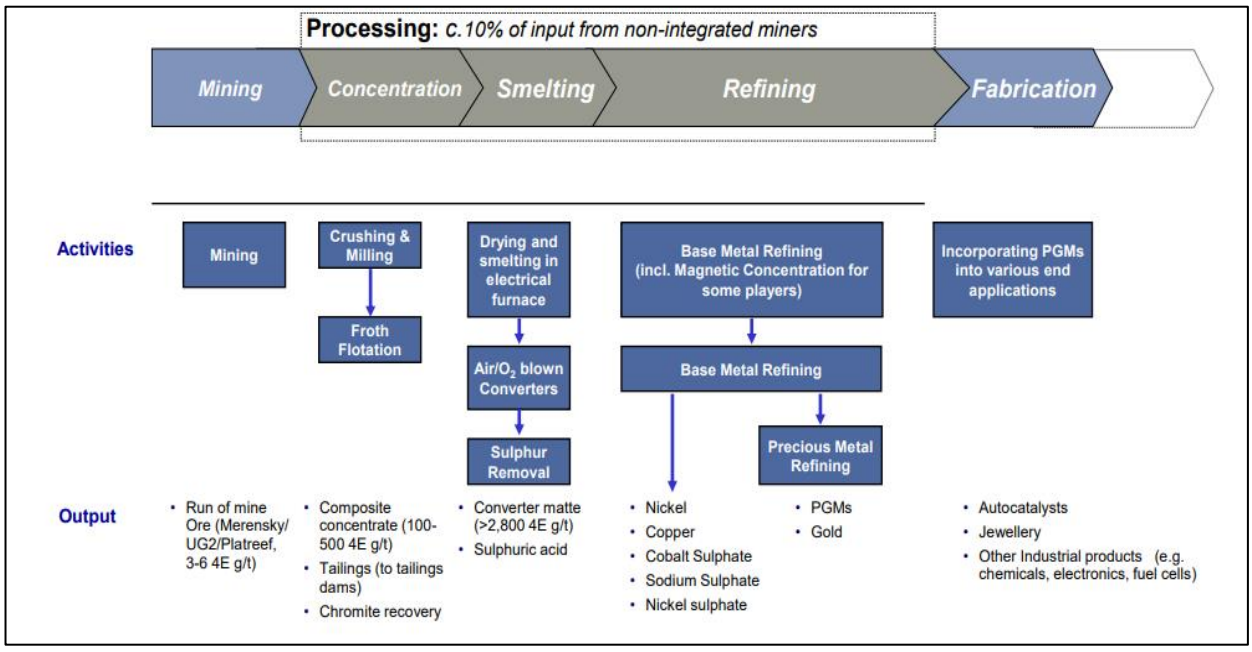


Figure 12: *Overview of the Platinum Group Metals (PGMs) processing value chain, from mining and concentration to smelting, refining, and fabrication (Ndlovu, 2021).*

The concentration primary stages include crushing and grinding followed by froth flotation which is responsible for the recovery of the valuable metals in the ore's fine granules. The design of a flotation circuit is a complex process that requires careful consideration of the properties of the ore being processed the desired grade and recovery of the concentrate and the economics of the operation. Flotation process control is a multi-tiered system with each level playing a specific role in maintaining the stability and efficiency of the process as a result Laurila et al. (2002) proposed a hierarchical sequence of four primary stages commencing with instrumentation followed by basic-level control. then advanced control. and concluding with optimisation. The

following picture shows the instrumentation and basic-level control of a Simplified process flow diagram (figure 1) and flotation circuit (figure 2) with both online measurement and assay samples for offline measurements.

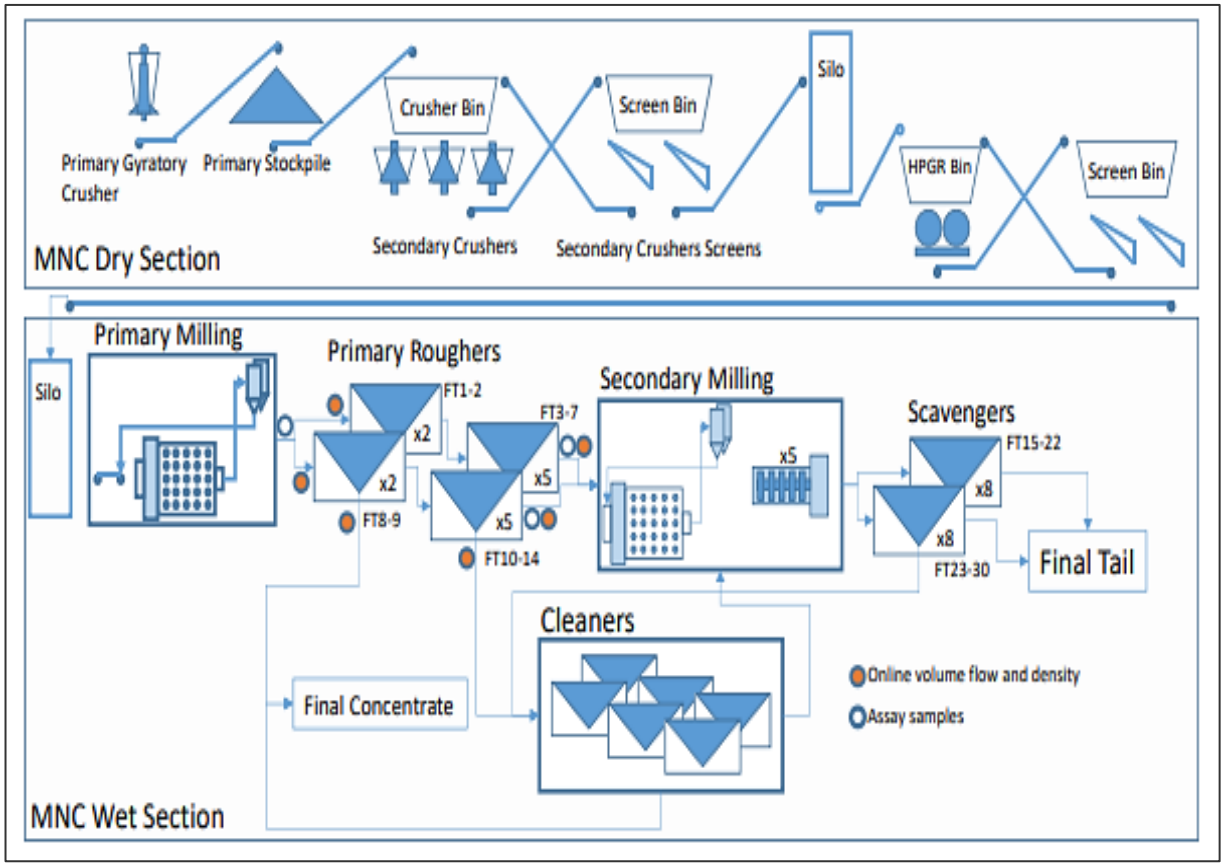


***Figure 13**: Overall process flow unit's diagram of the Mogalakwena North Concentrator (MNC) circuit* (Steyn & Sandrock, 2021).

The figures above show the primary and secondary platinum flotation cell in the Mogalakwena North Concentrator which is the largest platinum group metals concentrator in the world. Figure 14 shows the different units of the flotation circuit. The flotation circuit flow process begins with the delivery of raw ore to a grinding mill where it is crushed and ground into fine particles. This finely ground material is then mixed with water and various chemicals to create a slurry, which is fed into flotation cells. In these cells, air bubbles are introduced into the slurry. The chemicals added to the slurry make the target minerals hydrophobic, meaning they repel water and attach to the air bubbles.

As the bubbles rise to the surface, they carry the hydrophobic particles with them forming a froth layer that is skimmed off and collected. The remaining hydrophilic (water-attracting) particles which are typically waste sink to the bottom and are removed as tailings. The froth now enriched

with the desired minerals is further processed to remove any remaining impurities often through additional flotation stages or other separation methods. The final product is a concentrated form of the mineral which is then dewatered dried and prepared for further refining or direct use.

Each and every process or unit in the flotation circuit is vital to the overall success of mineral processing. Effective grinding liberates valuable minerals while conditioning sets the stage for their separation. Rougher cells determine the initial concentrate grade and cleaner cells further enhance the purity of the concentrate. Scavenger cells capture any valuable minerals that may have escaped the initial stages maximizing recovery. The primary objectives of milling circuits are to efficiently create a valuable product ensure high recovery rates in the subsequent stages and reduce operating expenses. Proper control and optimisation of each unit operation within the flotation process are crucial for achieving successful mineral recovery and producing high-quality concentrate.

3.1.1 Crushing and Grinding

Crushing and grinding are the initial stages in the flotation circuit where the ore is first crushed and then ground to liberate the Platinum Group Metals (PGMs) from the surrounding rock. This is a critical step as the efficiency of the subsequent flotation process heavily depends on the degree of liberation of the valuable minerals (Wills & Finch, 2015). Crushing typically reduces large chunks of ore to smaller. more manageable pieces. while grinding further reduces these pieces to a fine powder. The grinding process must be carefully controlled to achieve the optimal particle size. which is essential for effective separation during flotation. Too coarse a grind can result in incomplete liberation of the PGMs. while too fine a grind can lead to over-grinding. causing issues such as increased energy consumption and reduced recovery rates (Napier-Munn *et al.*, 1996).

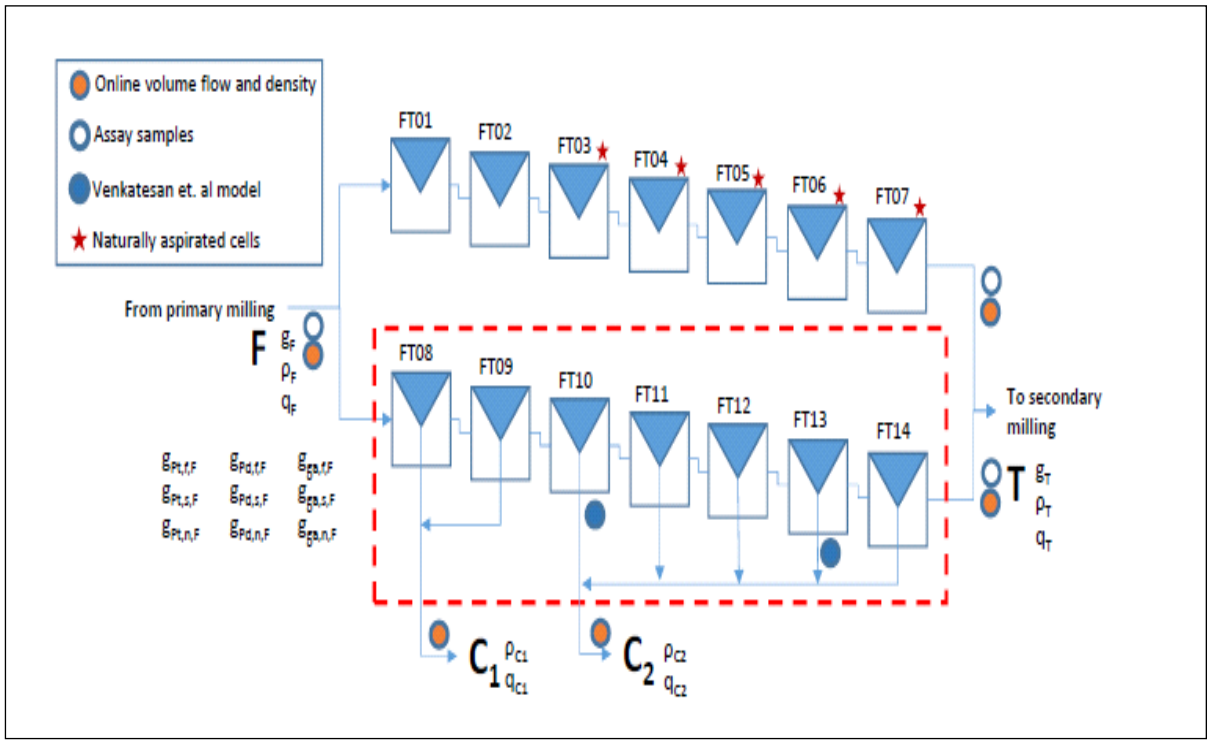
The particle size distribution resulting from the grinding process is closely monitored and controlled to ensure it meets the requirements for the flotation stage (Wills & Finch, 2015). Proper grinding not only liberates the PGMs but also ensures that the surface area of the particles is suitable for interaction with flotation reagents. This enhances the attachment of the PGMs to air bubbles during flotation. facilitating their separation from the gangue material (Gaudin et al., 1957).

In the grinding and crushing process. the type of equipment used is also crucial for achieving the desired outcomes. Primary crushers. such as jaw crushers and gyratory crushers. are used to

break down large chunks of ore into smaller. more manageable pieces. These are followed by secondary crushers. such as cone crushers or impact crushers. which further reduce the size of the ore. After crushing the material is then fed into grinding mills typically, ball mills or rod mills where it is ground to the fine particle size needed for effective flotation. The choice of crushing and grinding equipment and their configuration can significantly impact the efficiency of the process energy consumption. and the quality of the final product (Wills & Finch, 2015). Additionally, advances in grinding technology such as high-pressure grinding rolls (HPGR) and stirred mills have improved energy efficiency and reduced operational costs making the process more sustainable and economically viable (Schönert, 1988).

3.2.2 Flotation

The ground ore is then transferred to the flotation circuit. where various reagents are added to create conditions conducive to the attachment of PGM-containing particles to air bubbles. These bubbles rise to the surface and form a froth layer that is skimmed off. containing the valuable minerals.



The figures above show the primary and secondary rougher platinum flotation cells in the Mogalakwena North Concentrator. which is the largest platinum group metals concentrator in the world. In Figure 15. the red dotted lines highlight the second rougher bank. which is the primary area of interest for this research and the source of the historical data that will be used for training the models. The separation performance of flotation cells can be affected by several parameters. including the characteristics of the slurry the rate of slurry flow electrochemical

parameters and potentials the application and rate of chemical reagents levels of pulp within cells air flow rates into cells properties of the froth attributes of particles mineralogical composition of the ore. concentrations of minerals in the feed concentrate and tailings and rate of froth wash water (Laurila et al., 2002). These parameters are interconnected. and their individual and collective effects on the flotation process need to be considered for optimising the separation performance. The structure of a single flotation cell is presented in figure 17 showing some of the main parameters that influence the separation performance for grade and recovery. Proper management and optimisation of these factors are crucial for enhancing the efficiency and effectiveness of the flotation process leading to improved recovery rates and higher-grade concentrates. Understanding the interactions among these variables can help in fine-tuning the flotation conditions thereby maximizing the yield of valuable minerals and minimizing losses.

The mineral flotation procedure involves the collision and adherence of hydrophobic particles with bubbles followed by the release of these bubbles from the slurry carrying the hydrophobic particles with them. These stages are known as the attachment and detachment processes. This process separates minerals from complex ores based on variations in their hydrophobic properties at the interface of three phases: solid. liquid. and gas (Jovanović & Miljanović. 2015). Flotation cells are divided into two primary regions namely the pulp zone and the froth zone. The pulp zone is responsible for mixing reagents, collectors, depressants and facilitating the attachment of valuable minerals to the generated bubbles. In contrast the froth zone is characterized by bubbles forming froth which ensures the stability needed to transport the bubbles above the cell. Over the past decades fluidized flotation cells have demonstrated effective separation capabilities across a broad spectrum of particle sizes. To optimize PGM recovery via flotation, it is essential to focus on liberating minerals containing platinum group elements (PGEs) while suppressing gangue minerals. Comminution circuits in PGM processing typically involve multiple stages of milling followed by flotation in MF2 or MF3 circuits to improve recovery rates, although challenges such as overgrinding and the flotation of fine particles must be addressed.

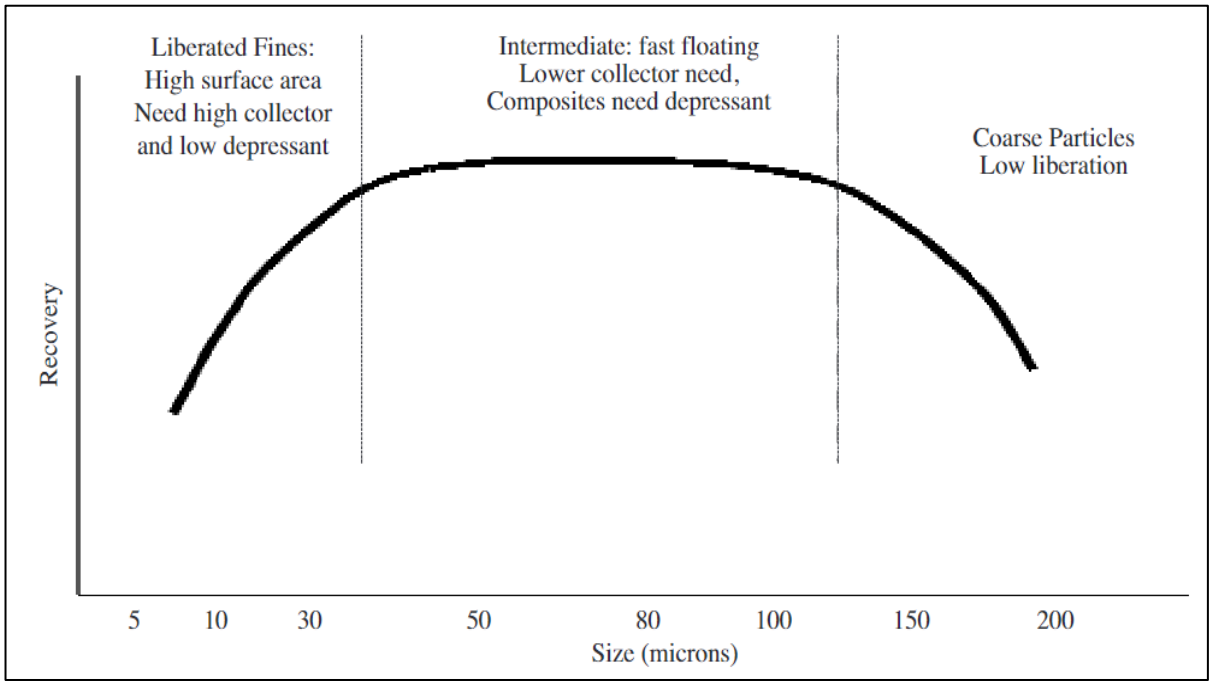


Figure 14: *Ore feed size relationship with recovery (Rule & Anyimadu. 2007)*

In the figure above. Rule & Anyimadu (2007) highlighted the importance of grinding ore into small particles to ensure more effective liberation of valuable minerals. The optimal particle size range for this process is approximately 37 to 100 microns. Beyond this range recovery begins to decline as the particles become coarser. This reduction in recovery is primarily due to the decrease in surface area available for interaction with flotation chemicals which limits the efficiency of the separation process. Gomez-Flores et al., (2022) further discovered that particle size also plays a critical role in machine learning models in predicting grade and recovery.

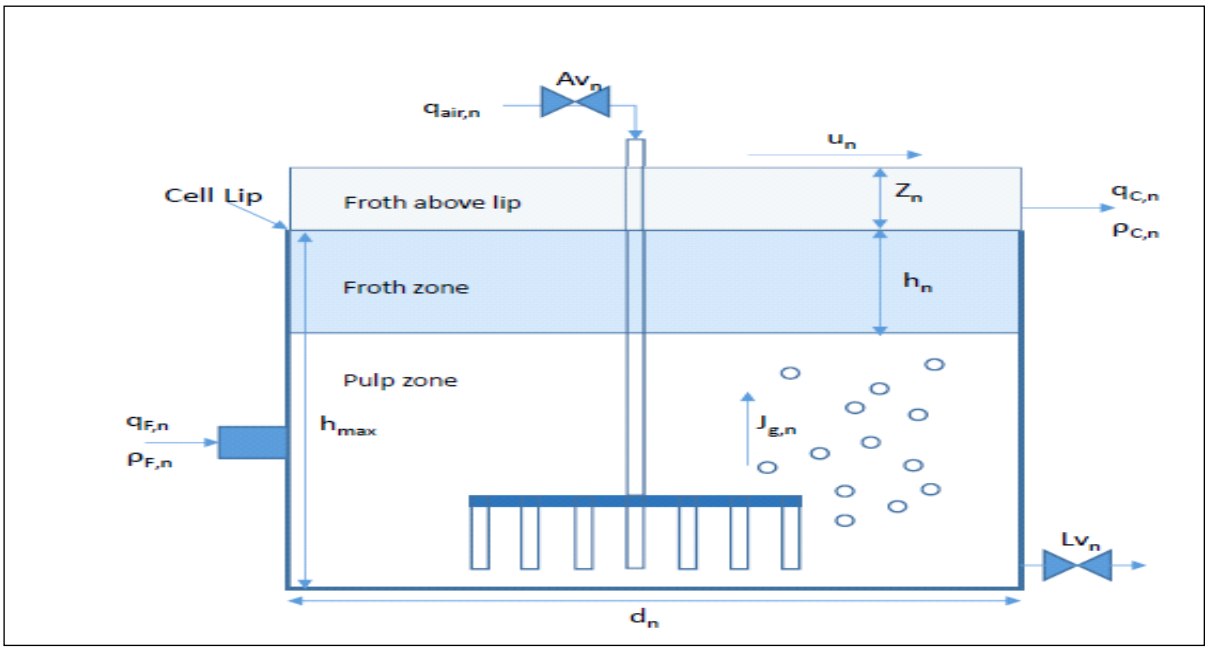


Figure 15: *Flotation Cell showing important process parameters (Steyn & Sandrock,2021)*

According to Han et al., (2023). mineral flotation is most effective within a particle size range of approximately 20–150μm. A common approach to establish the relationship between recovery and particle size involves splitting the flotation circuit feed into various size ranges. measuring the average size of each range conducting flotation on each size group and documenting the recovery for each range. The optimal flotation size range is influenced by multiple factors including flotation unit type, bubble size, particle size, froth stability, thickness, reagent composition, and turbulence intensity. Figure 17 showed the different factors of significance in a flotation cell for both the froth zone and the pulp zone.

In addition to particle size there are several other process variables within flotation cells significantly influence both grade and recovery. One of the most critical factors is the bubble size which is responsible for the attachment and transportation of the valuable minerals. Ren et al., (2019) elucidated that smaller bubbles improve recovery due to the increased probability of particle-bubble collisions particularly for fine particles. This is because smaller bubbles offer a larger surface area relative to their volume facilitating more effective particle attachment and gentler hydrodynamic conditions. Furthermore, bubble velocity commonly known as froth velocity in the froth zone is essential in improving recovery. Newcombe et al., (2013) observed that higher bubble velocities enhance particle attachment rates by increasing the frequency of particle-bubble collisions. which can improve flotation recovery. However, at excessively high velocities bubble coalescence becomes more likely reducing the available surface area for particle attachment and ultimately lowering flotation efficiency.

Another key variable is gas flow rate, which needs to be carefully controlled to maintain an optimal balance between recovery and grade. Excessively high gas velocities can lead to turbulence and reduced attachment efficiency (Varbanov et al., 1993). Air bubbles in flotation cells are formed when air pumped into the tank interacts with the slurry in the pulp zone aided by the agitator (Wang et al., 2020). Proper control of variables such as agitator speed. airflow. and chemical dosage is essential to ensure the formation of stable bubbles which are crucial for efficient flotation (Ren et al., 2019). The size of the bubbles is another critical factor. Very large bubbles are prone to bursting which disrupts the flotation process and reduces recovery efficiency (Newcombe et al., 2013). Therefore, maintaining optimal bubble size and stability is essential for effective particle attachment and flotation performance.

The following graph illustrates the relationship between bubble size and flotation recovery over time. showing that smaller bubbles (20 µm) result in higher recovery rates compared to larger bubbles (30 µm and 40 µm). This trend is consistent with Ren et al., (2019) who found that smaller bubbles increase the likelihood of particle-bubble collisions enhancing recovery especially for fine particles. The 20 µm bubble size shows the fastest recovery reaching around 80% within the first 6 minutes while larger bubbles exhibit slower recovery due to reduced collision efficiency and greater susceptibility to detachment and coalescence (Newcombe et al., 2013). The graph underscores the importance of optimising bubble size as smaller bubbles lead to faster and more efficient flotation performance, particularly in fine particle flotation.

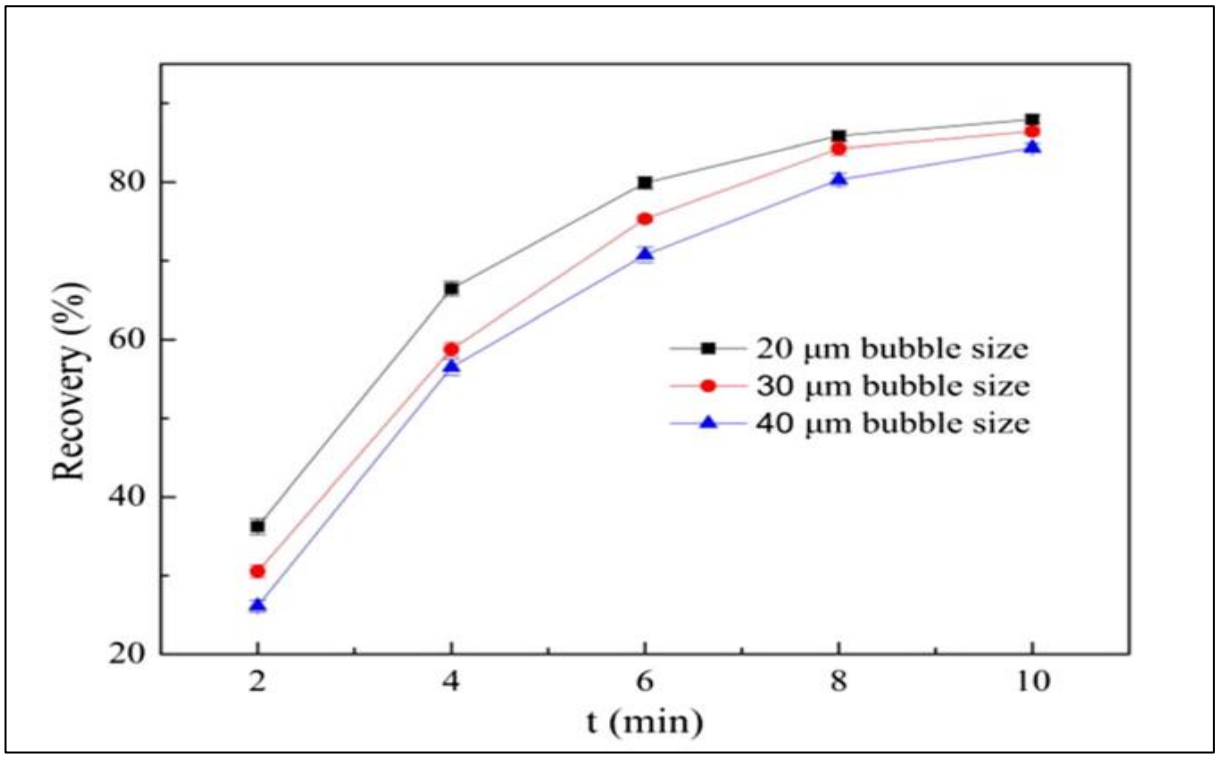


Figure 16: *Effect of bubble size on flotation recovery over time and the overall flotation performance.*

Turbulence in the cell also impacts flotation performance. as it influences bubble-particle contact time and collision frequency. Lower turbulence allows for higher recovery rates but can slow down the process (Wang et al., 2020). Additionally. pulp density and slurry viscosity are vital considerations; higher densities can reduce recovery by shortening residence time. while lower densities may improve recovery at the expense of grade (Reis et al., 2017). Finally, the hydrodynamics within the cell including the interaction between gas dispersion and liquid flow are crucial for ensuring efficient particle-bubble interactions and overall flotation performance (Wang et al., 2020). Managing these variables collectively is essential for optimising both grade and recovery in flotation systems.

Gas holdup is a critical variable for assessing bubble cluster size understanding the influence of gas on flotation operations and serving as a hydrodynamic indicator for the pulp phase (Kromah et al. 2022). Gas holdup is influenced by factors such as bubble size, particle concentration, froth concentration, superficial liquid rate, superficial gas rate (gas flux), and bed height (Han et al. 2023). Frother's facilitate the dispersion of air within the slurry forming fine and evenly sized bubbles while also improving their stability. Stable bubbles benefit recovery; however, excessively stable bubbles can transport valuable minerals along with unwanted gangue potentially negatively impacting overall recovery.

The flotation process involves treating the valuable mineral with a hydrophobic chemical reagent known as a collector which plays a critical role in influencing key outcomes such as grade and recovery (Popli et al. 2018). To achieve selective flotation of different minerals, additional agents such as modifiers or depressants are used to control the surface properties of other minerals. minimizing interference and improving the selectivity of the process. Frother's another essential reagent enhances flotation performance by improving the fluidization characteristics in the flotation cell. Frother's work through three primary mechanisms: reducing bubble size to increase bubble surface area flux minimizing surface tension gradients to mitigate the Marangoni effect and altering bubble surface charge to prevent bubble coalescence (Han et al., 2023). These chemical agents help maintain optimal flotation conditions minimizing the impact of unwanted minerals and ensuring more consistent grade and recovery outcomes. Monitoring these variables is critical for maintaining process efficiency.

Monitoring these variables is critical for maintaining process efficiency. Mehrabi et al., (2014) developed a machine vision system to monitor key flotation parameters such as bubble size distribution, bubble count, froth velocity and stability ensuring stable operating conditions and improving overall flotation performance. Similarly, Gomez-Flores et al., (2022) demonstrated the use of machine learning models to predict flotation performance by incorporating physicochemical variables such as particle diameter, bubble size, and collector concentration. significantly improving recovery predictions. In addition, Chunhua (2010) used a Support Vector Machine (SVM) model to monitor and predict particle size in grinding circuits which directly impacted grade and recovery in downstream flotation processes offering real-time control over particle size distribution and process efficiency.

3.2.3 Smelting and Refining

The recovery and purification of platinum group metals (PGMs) from primary and secondary sources involve a sophisticated sequence of metallurgical and chemical processes to achieve the high purity standards required for advanced industrial applications. The smelting of Platinum Group Metals (PGMs) represents a critical downstream stage in the mineral processing value chain, following initial concentration through flotation. This step is essential for the extraction and refinement of valuable metals such as platinum, palladium, and rhodium. This process follows flotation where PGM-bearing ores are concentrated through froth flotation techniques effectively separating them from gangue minerals (Crundwell et al., 2011). During smelting the PGM concentrate undergoes high-temperature treatment in a furnace where it is melted along with fluxes and reductants. This process serves multiple purposes but mainly it removes impurities such as sulfur and iron that are typically present in the concentrate (Jones, 2002). The choice of fluxes and reductants is crucial as they influence the chemical reactions and overall efficiency of the smelting process (Crundwell *et al.*, 2011). The following figure illustrates the concentrates undergoing high-temperature treatment in a furnace, along with the reactions involved in the process.

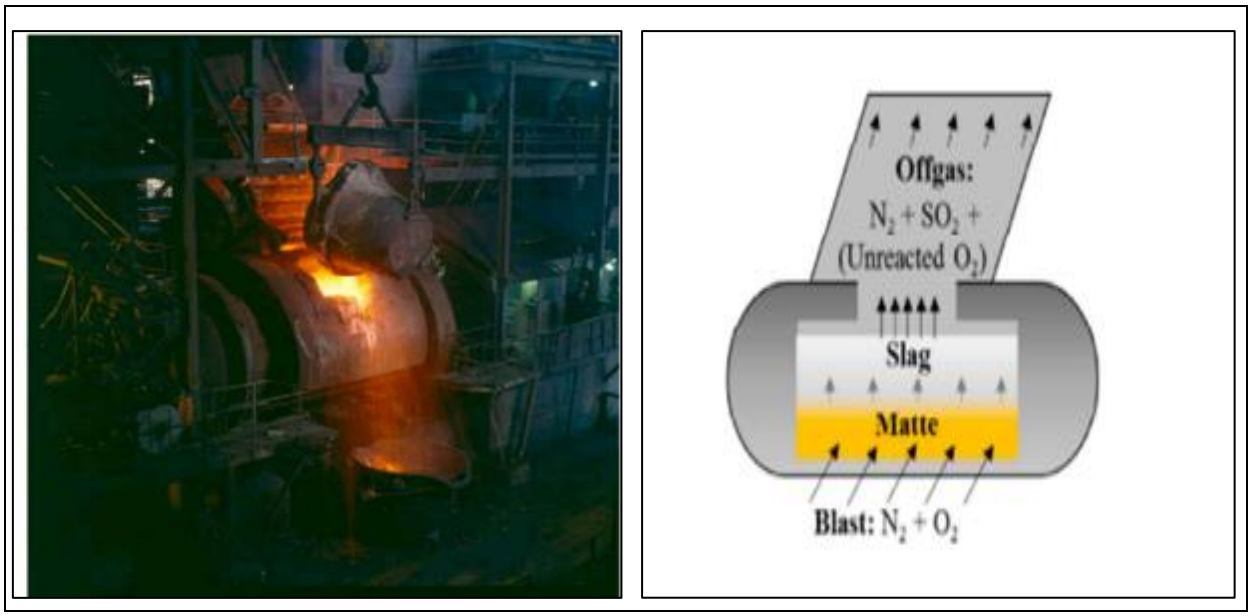


Figure 17: *PGMs concentrates undergoing heat treatment (Navarra* et al., *2020)*

After initial smelting, where PGMs are concentrated into a molten matte, the material undergoes targeted refining to isolate individual metals. A critical step involves leaching the matte with acids e.g., sulfuric or hydrochloric acid to dissolve base metals, leaving behind a PGM-enriched residue (Jones, 2002). This residue is further processed using methods such as solvent extraction, electrorefining, or selective precipitation to separate and purify platinum, palladium, rhodium, and other PGMs. For instance, hydro carbonyl technology is employed to selectively concentrate

specific metals, while ammonium chloroplatinate dissolution using sodium metabisulfite followed by hydrochloric acid and sodium chlorate enables the production of high-purity platinum. Additional purification steps, such as remelting platinum-rhodium alloys with Al_2O_3 powder in an inductive furnace, ensure the removal of residual impurities. The refining process is integral to meeting the stringent quality demands of applications like automotive catalytic converters, medical devices, and electronics. For example, spongy platinum a porous, high-surface-area form is produced through iterative cycles of precipitation, filtration, and thermal treatment, yielding materials compatible with industrial catalysis. These methods not only optimize metal recovery efficiency but also align with economic and environmental sustainability goals.

Secondary PGM recovery from spent catalysts, electronic waste, and industrial byproducts has gained prominence due to growing resource scarcity. Modern techniques such as membrane separation, supercritical fluid extraction, and advanced solvent extraction systems are being leveraged to reclaim PGMs from recycled materials.

These innovations minimize reliance on primary ore extraction, reduce environmental footprints, and support circular economy principles. For example, solvent extraction using selective ligands (e.g., alkyl sulfides or phosphine oxides) enables the separation of PGMs from complex matrices, while supercritical CO_2-based methods offer a greener alternative for metal recovery. Key refining protocols, such as pulpification, sodium metabisulfite addition, and controlled thermal treatments, remain foundational to achieving >99.9% purity levels. By integrating these processes often in tandem with spectroscopic quality control the industry ensures PGMs meet the exacting specifications of high-tech sectors.

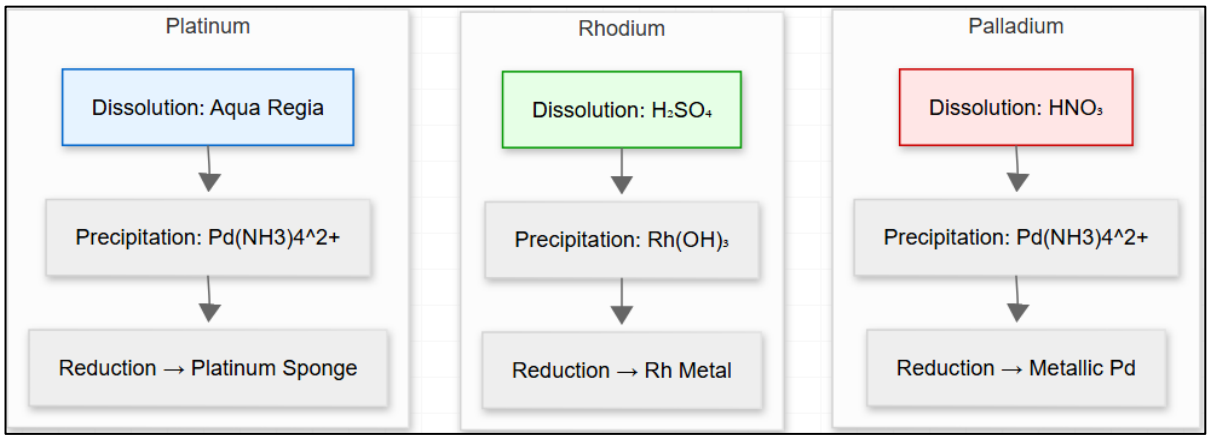


Figure 18: *A streamlined flow diagram depicting the refining of platinum, palladium, and rhodium through core hydrometallurgical processes.*

Chapter 4: Methodology

4.1 Research Design

The research design for this study is framed around developing a data-driven soft sensor that predicts both grade and recovery in a platinum secondary rougher bank. Soft sensors, or virtual sensors, are computational models that estimate hard-to-measure process variables from other easily measured ones (Kadlec et al., 2009d) . In the context of this study, the goal is to use historical process data to build a predictive model that can provide real-time estimates of grade and recovery, using key process parameters around the flotation circuit. These are critical performance metrics in mineral processing. The development of soft sensors involves a series of processes that need to be adopted to have a robust and reliable machine. The process of soft sensor development is illustrated in the accompanying schematic diagram, which details each step in a clear and systematic manner.

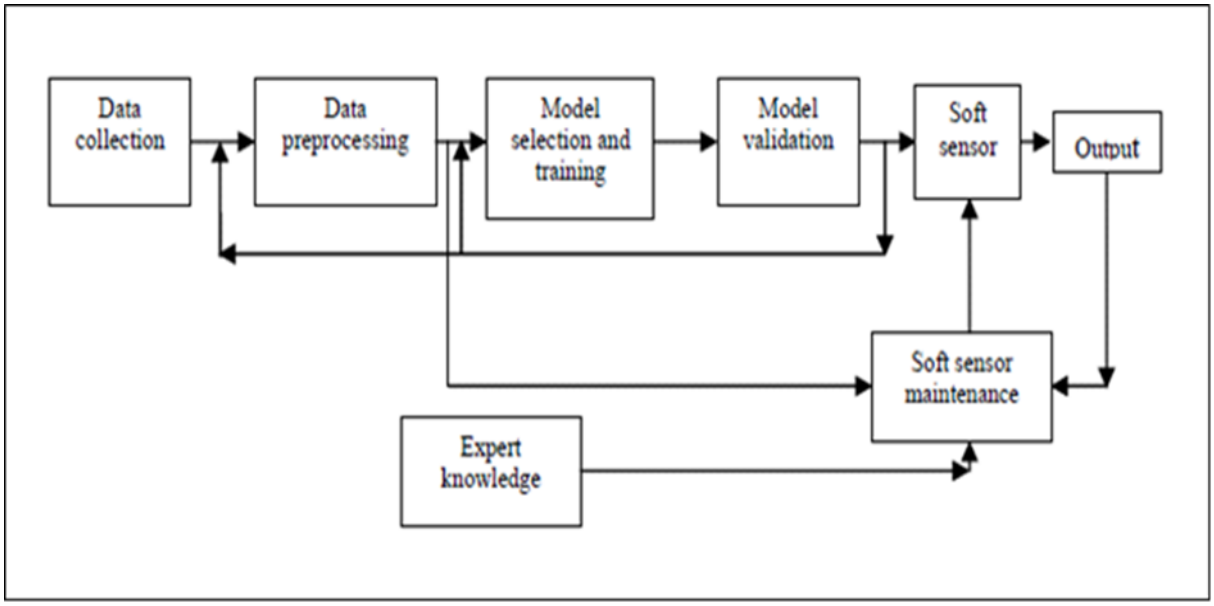


Figure 19: *General soft sensor design procedure (Pani & Mohanta, 2011).*

This study adopts a structured approach that begins with data collection and pre-processing, followed by feature selection, model development, and evaluation. The figure illustrates that after model validation; it is necessary to revisit the data processing stage. This iterative process is widely applied in soft sensor development because during model evaluation additional insights about the data come to light. For example, the identification of redundant variables that the feature selection technique may have selected. These redundant variables were manually reviewed and deselected, and the algorithm was then rerun to compare performance, ensuring a more refined model.

The model development incorporated a subset of expert knowledge, which was crucial for the study. This expertise was sourced from a collaborative team, including professional engineers from Anglo American and professors from Wits University, who provided valuable input on decision-making and process assumptions. Although soft sensor maintenance was beyond the primary scope of this study, it was partially addressed by training the algorithms iteratively using a factor of three and averaging the performance outcomes to produce the final output.

The data-driven modelling approach is supported by a thorough literature review that highlights the effectiveness of machine learning algorithms in similar industrial applications. For example, Bhardwaj et al. (2015) demonstrated the potential of machine learning models to predict key performance indicators in a cement plant, a process with similar complexity and variability as that of a platinum concentrator. Additionally, recent applications of machine learning in the flotation circuit, such as the work by Smith et al. (2022), have shown significant improvements in predicting flotation performance metrics, optimising reagent dosage, and enhancing overall recovery rates. These advancements underscore the growing role of machine learning in process optimisation within mineral processing industries, particularly in managing the complex, non-linear interactions inherent in flotation circuits.

The iterative nature of this research design allows for continuous refinement of the models based on performance evaluations. Multiple machine learning algorithms including linear regression, support vector machine, Multi-Layer Perceptron, Random Forest, and Extreme Gradient Boosting, are tested and validated using different data split strategies to ensure the robustness of the final model. This approach is consistent with best practices in machine learning, where model performance is enhanced through rigorous testing and validation (Goodfellow et al., 2016). The final selection of the model is based on a comprehensive set of evaluation metrics, which consider both the precision of the predictions and the model's ability to generalize across different operating conditions.

4.2 Data Collection

4.2.1 Description of Data Sources

The data used in this study was sourced from the Anglo-American Mogalakwena platinum concentrator plant situated in Limpopo, South Africa. It was extracted from their PI historian archive, where historical and real-time process data is stored. A comprehensive dataset encompassing various process variables recorded over approximately four and a half years, starting from January 2019 to the end of March 2024. The period and frequency of the data was chosen due to the nature of the grade data. The grade composite samples are collected in specific

time intervals in the process line which is then summed and stored as daily averages in the PI historian archive. Since the grade is available as daily averages, a decision was made to also obtain the process data which are input variables to our algorithm as daily averages to synchronize their frequency for the time period.

The assumption is that daily averages of the process variables will reflect the same process dynamics as our target variable (4T Concentrate grade). The synchronization of the data is crucial, as it ensures that the machine learning algorithm accurately learns from what was occurring in the process at that specific moment in time. The data collected duration was also extended to ensure that machine learning algorithms had sufficient data to learn effectively, enhance accuracy, and improve generalisation capabilities. Machine learning models, particularly those dealing with complex industrial processes, benefit from large datasets as they enable the models to capture underlying patterns more accurately and generalize better to new, unseen data (Goodfellow et al., 2016). Large datasets help mitigate overfitting, leading to more robust and reliable models. This dataset is representative of the complex and dynamic nature of mineral processing operations, where numerous factors interact to determine the final product quality.

The Mogalakwena platinum concentrator has primary and secondary rougher cells, as seen in Figure 14. The process data for this study was gathered from the secondary rougher banks within a platinum flotation circuit. This decision was guided by the significant role these banks play in the recovery of Platinum Group Metals (PGMs), as highlighted by Steyn & Sandrock (2021) in their causal model of an industrial platinum flotation circuit. Given that the secondary rougher bank consists of seven Wemco tanks, each with a capacity of 100 cubic meters, it was observed that certain process variables remain consistent across tanks 10 to 14.

As a result, these variables were deemed redundant for modelling process dynamics and were excluded from the analysis. Instead, the focus was placed on the process variables from tanks 8 and 9, which are crucial to understanding the system's behaviour. This decision is supported by the findings of Steyn and Sandrock (2021), who emphasized that tanks 8 and 9 are particularly significant contributing approximately 60.8% of the total recovery within the rougher bank achieving an overall recovery of 64.5%. This significant contribution underscores the importance of accurately modelling the soft sensor to estimate PGMs grades, which is essential for optimizing recovery rates throughout the flotation circuit.

The data collection process was meticulously designed to ensure a comprehensive representation of the Mogalakwena Concentrator's operations. This approach was executed in two distinct stages. Initially, efforts concentrated on collecting data from flotation cells 8 and 9, which are critical as they contribute approximately 65% to the overall grade and recovery metrics. In prioritizing these cells, the study aimed to capture the most influential variables affecting the concentrator's performance. Subsequently, an assessment was conducted to determine the sufficiency of the collected data. If the data from cells 8 and 9 were deemed insufficient to model the process dynamics accurately, additional data from cells 10 to 14 were incorporated. This supplementary data ensured a robust and comprehensive dataset. This two-stage data collection strategy ensured that the study captured a holistic view of the process variables, thereby enhancing the reliability and validity of the subsequent analysis.

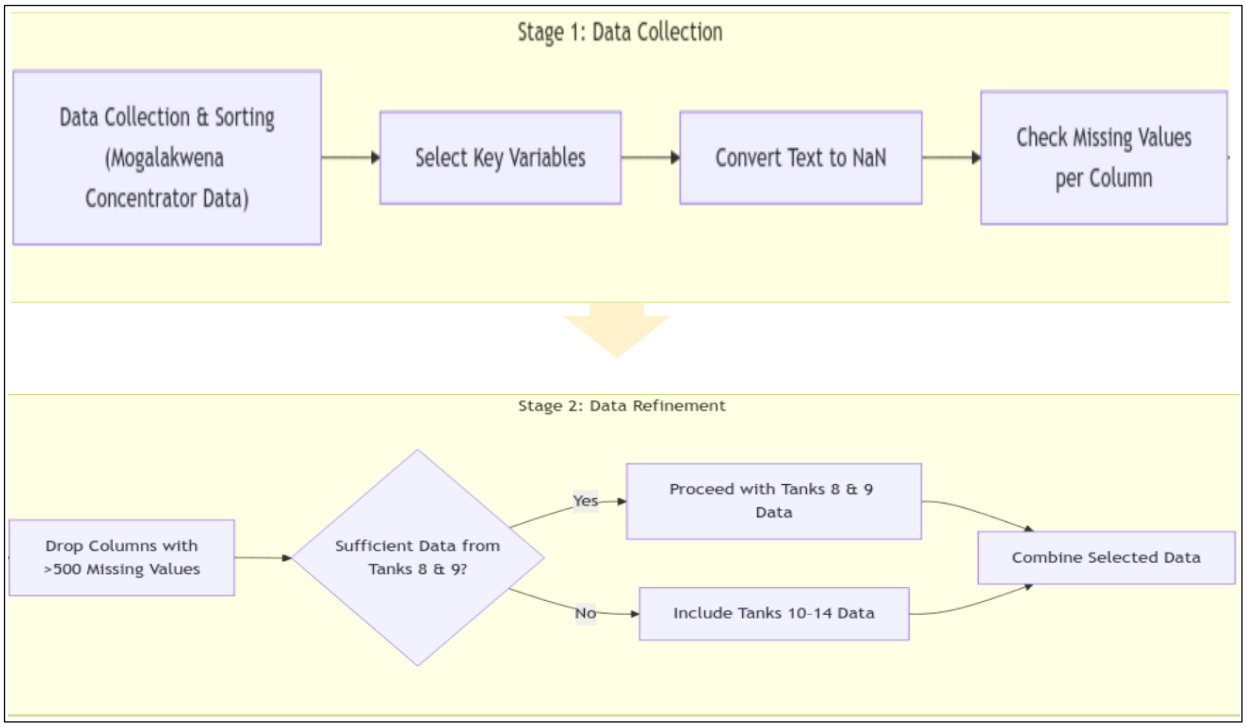


Figure 20: *Two stage process for data collection and Refinement.*

The resulting initial dataset contained 13,908 rows of entries across 92 variable columns, which included a mix of input process variables and output performance metrics. This was all collected from the second rougher cells starting from 8 up to 14 as it is highlighted by the red dotted line in figure 14 above. To obtain a sufficiently large dataset, daily averages of both input and output variables were collected from January 2019 to July 2023. This approach was necessitated by the limitation that grade data was only available as daily averages. Consequently, a four-year period was considered to ensure an adequate amount of data for effective model training.

These variables are critical for understanding the operational parameters of the concentrator, starting from ore feed to the rougher cells to airflow rates, froth velocities, mass pull fractions, and concentrate assays. The data collection process involved the use of advanced sensors and monitoring equipment installed throughout the concentrator plant. These sensors continuously record real-time process data, providing a detailed overview of the plant's operations. The data includes time-stamped records, essential for time-series analysis which helps capture the temporal dependencies inherent in the process.

Detailed data is essential for developing an accurate predictive model that reflects real-time concentrator operations (Fortuna et al., 2007). Visualizing the data before data cleaning using tools like Python and Rapid Miner was utilized to gain initial insights into its patterns, distributions, and relationships. This step provides an overview of the data structure which enables the identification of potential issues such as outliers or inconsistencies. Scatter plots, histograms, and bar charts are employed to facilitate these initial analyses and guide subsequent data pre-processing. Python was chosen for data pre-processing due to its exceptional capability to handle large datasets and its extensive libraries tailored for data science and machine learning. One of the key advantages of Python is its simplicity and readability, which allow for rapid development and iteration in data-intensive projects (Van Rossum & Drake, 2009). Python's extensive ecosystem includes libraries like Pandas and NumPy which is specifically designed to manipulate large datasets efficiently making it easier to clean, transform, and analyse data (McKinney, 2017; Oliphant, 2015).

To ensure the data processing tasks were carried out with maximum efficiency, it was essential to update the Python environment and all relevant libraries to their latest versions. Before initiating any data cleaning procedures, the Python version was checked and updated which was followed by upgrading all libraries to their most recent releases. This step was critical to leverage the latest improvements and optimisations in the libraries used for data analysis. After updating the necessary libraries for efficient data processing were imported. These libraries are foundational to the subsequent analysis and were carefully selected to ensure they provided the required functionality for handling, cleaning, and analysing the data. The initial set of imported libraries is shown in the figure below which illustrates the tools prepared for the upcoming data processing tasks.

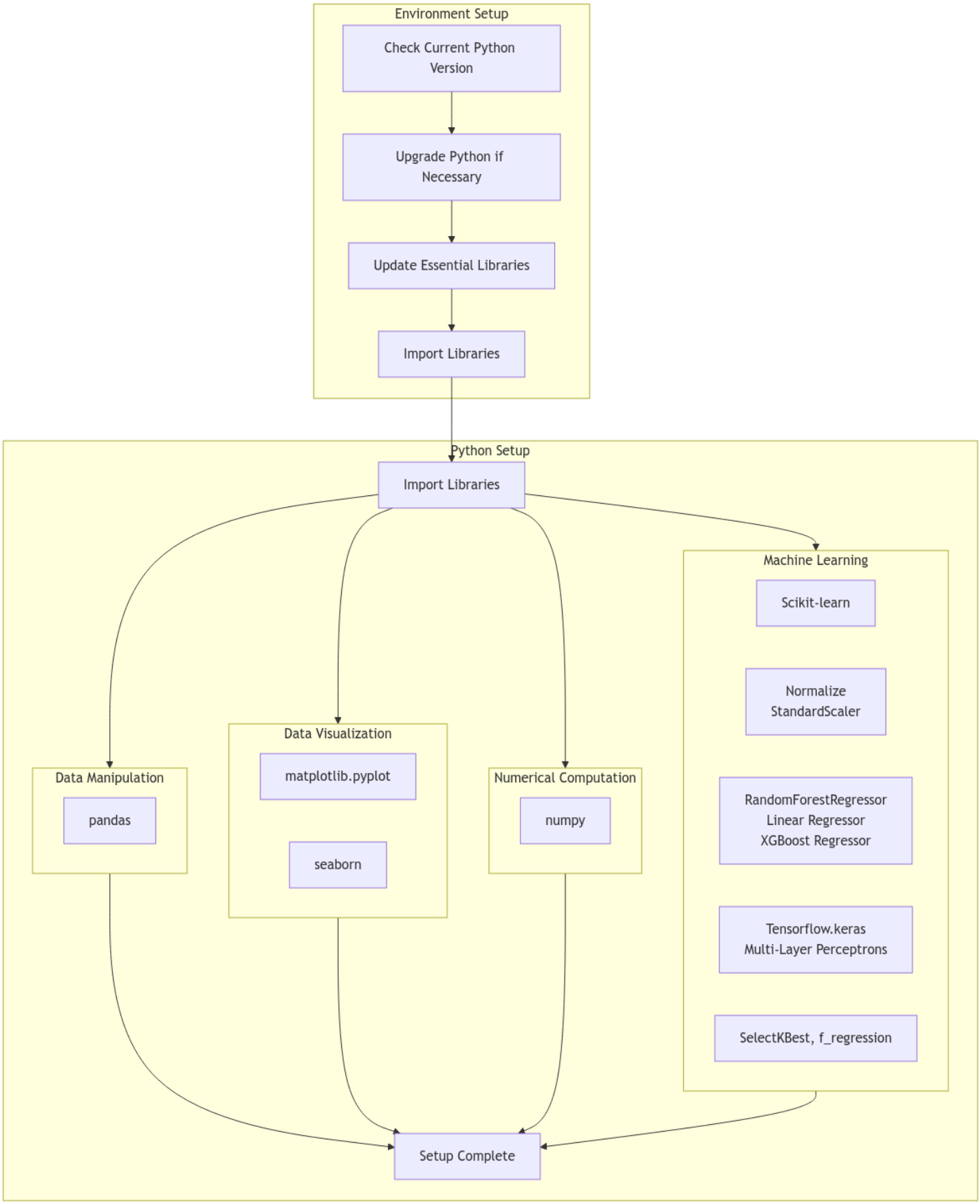


Figure 21: *Python Environment Setup*

The imported libraries in the above figure each serve a crucial role in the data cleaning and analysis process. Pandas are fundamental for data manipulation allowing efficient handling and transformation of datasets. Numpy provides essential numerical operations on arrays supporting the mathematical functions required for data processing. Matplotlib. pyplot and sea-born are used for creating visualizations while Matplotlib provides basic plotting capabilities, Sea-born enhances these with more complex statistical visualizations aiding in the exploration of data patterns. Scikit-learn modules are integral for both pre-processing and modelling: Standard Scaler normalizes data, ensuring consistent feature scaling, and PCA reduces dimensionality, preserving essential information while simplifying the dataset. SelectKBest with f_regression selects the most relevant features for modelling. Together these libraries establish a robust environment for performing efficient and accurate data cleaning and analysis.

Moreover, Python excels in the field of machine learning with libraries such as Scikit-learn, Tensor Flow, and Keras (Pedregosa et al., 2011). These libraries offer built-in algorithms that cover a wide range of machine learning models, from simple linear regression to complex deep learning architectures. Importantly, these algorithms are highly customizable, allowing researchers and developers to modify them to meet specific needs (Chollet, 2017). This flexibility is crucial when developing models that must adapt to the unique characteristics of a dataset or optimize performance for specific applications. Additionally, Python's large and active community continuously contributes to its growth, ensuring that the latest advancements in machine learning are quickly integrated into the available libraries (Müller & Guido, 2016). This makes Python not only a powerful tool for current project, but also future proof as new techniques and algorithms emerge. Compared to other languages, Python's balance of ease of use, robust libraries, and flexibility makes it an ideal choice for developing machine learning algorithms (Lutz, 2013).

4.2.2 Data Pre-processing

Data pre-processing is a crucial step in preparing the dataset for modelling, as it addresses the inherent issues found in raw industrial data, such as missing values, outliers, and non-numeric entries. Raw data from industrial processes often contain noise, inconsistencies, and gaps due to sensor malfunctions, communication errors, or manual data entry mistakes (Zhao et al., 2017). Therefore, pre-processing is essential to clean and transform the data into a suitable format for analysis. The initial phase of data pre-processing involved data cleaning, where non-numeric columns were removed, and missing values were handled using various imputation methods. Continuous variables were imputed using mean or median values, depending on their

distribution. This approach is consistent with standard practices in data science, where the choice of imputation technique is often guided by the nature of the data and the specific requirements of the analysis (Rubin, 1987).

Since this is a regression exercise, the first step was to ensure that all data was in numerical format, as non-numerical values can disrupt the modelling process. This was accomplished by detecting and removing any text data present in the dataset. For example, specific anomalies such as [−11059] no good data for calculation and [−10722] PINET were identified and eliminated. The data was then left with only numerical values, though this process did result in some NaN (Not a Number) values where the text data was removed. To handle missing data, a strategy was employed that centred on the ore feed rate, grade, and recovery as the key target variable. The primary goal was to retain as much valuable information as possible to ensure the accuracy and effectiveness of the machine learning models. First, the ore feed data was analysed by plotting it over time to observe its variability followed by PGM grade and tails grade. This is demonstrated in the following figures 24 to 26 below.

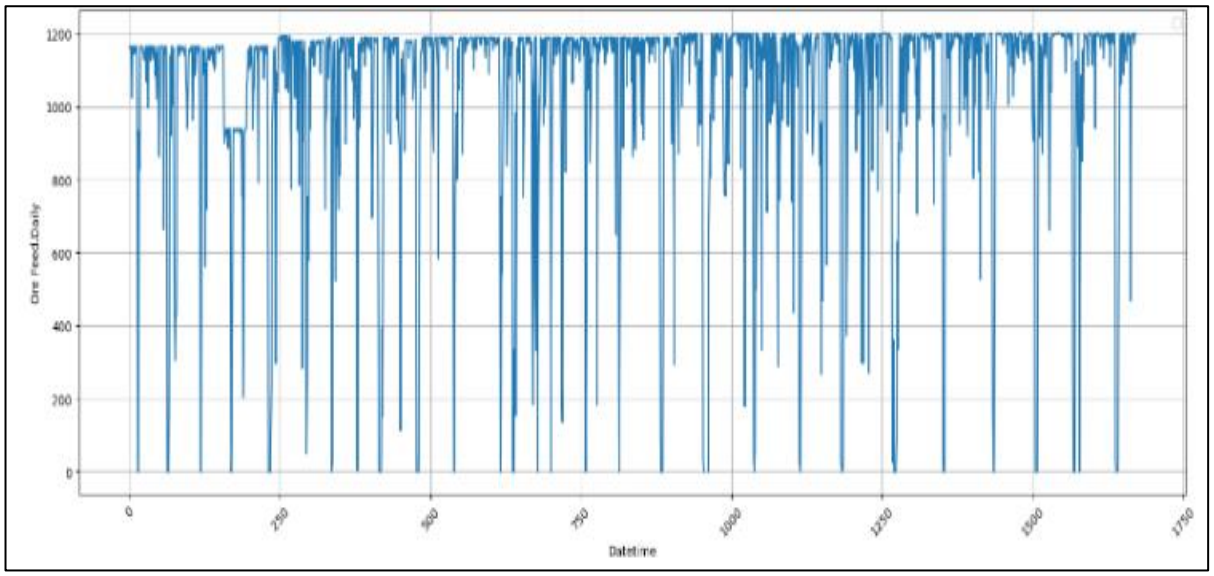


Figure 22: *Ore feed plot overtime before data cleaning obtained from python.*

From the above graph, it was noted that during normal operations the ore feed to the rougher bank maintained a set point of approximately 1200 t/h. This analysis was crucial as it helped identify periods of downtime represented by feed rates dropping to zero which can be seen from the above figure. This indicated that the plant was not operational due to various potential issues. To clean the data a low threshold of 500 t/h was established. This allowed us to remove the less reliable data from the set, meaning that any row with an ore feed below this threshold was deemed unreliable and was therefore removed from the dataset.

The filter ensured that only data representing normal operating conditions were retained. After addressing the ore feed data, we turned our attention to the PGM grade. Any row with missing grade data or with a grade value less than zero was removed. A negative grade is impossible in reality and likely indicates a system error or faulty data recording; hence, these entries were deemed untrustworthy and excluded.

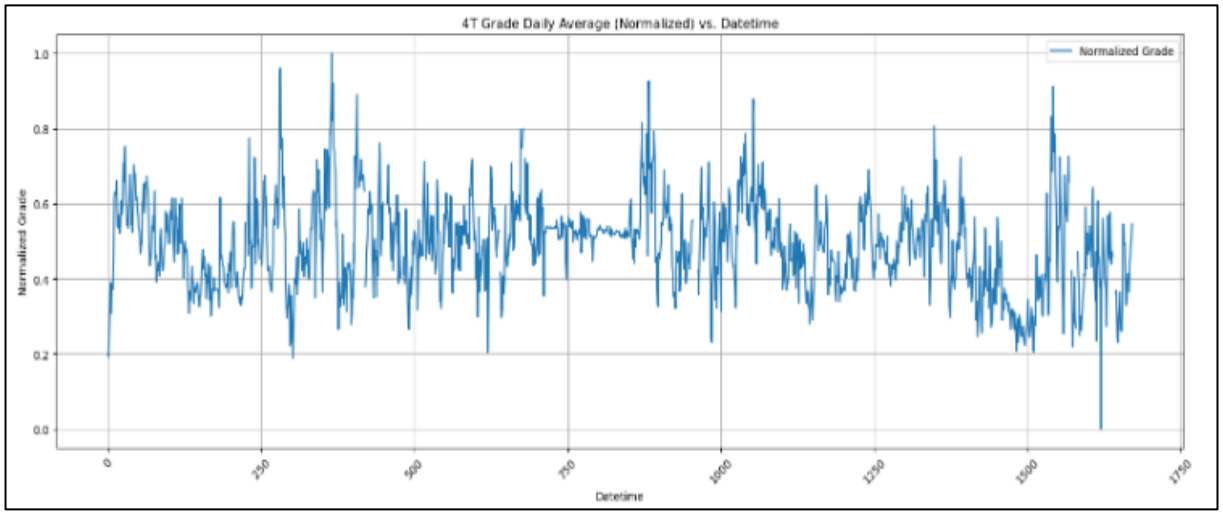


Figure 23: *PGM grade data over time normalized before data cleaning obtained from Python.*

The figure above illustrates how the PGM grade varies over time. It is noticeable that gaps are indicating missing data, particularly between the intervals of 250 to 500 and 1500 to 1750. To handle missing data efficiently, we utilized Python's Pandas library, specifically the function df. isna (). sum (), to assess the number of missing values in each column. This approach allowed us to quickly identify columns with more than 10% missing data, which were subsequently dropped from the dataset to maintain the integrity of the analysis.

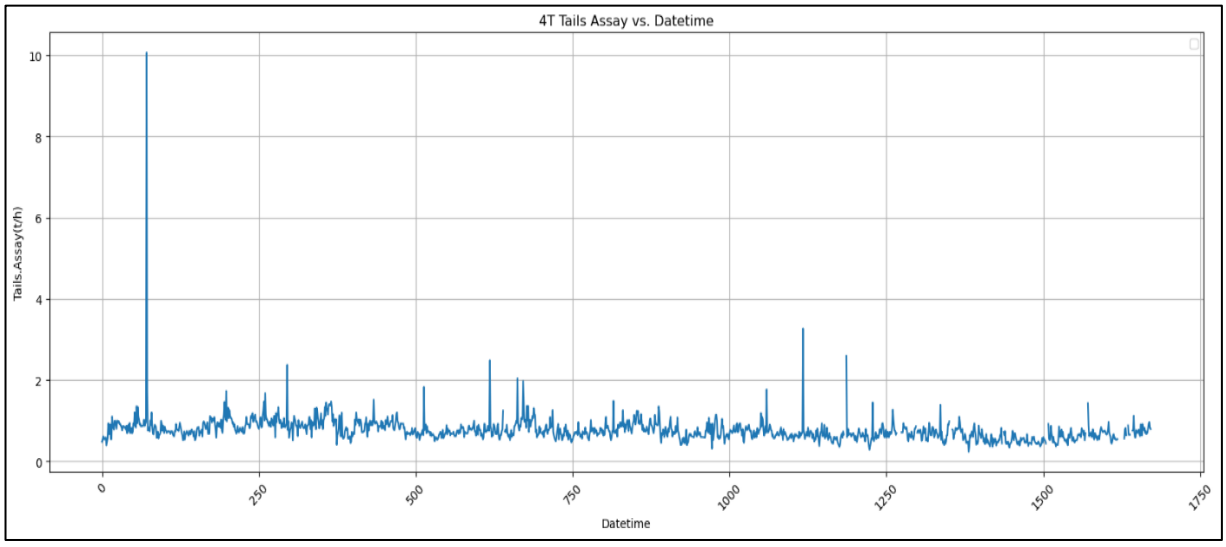


Figure 24: *Tails plot over time obtained from Python.*

The tails plot revealed the presence of an outlier, spiking to 10, while the majority of the data remained around 3 t/h. This tail data was used as a reference to filter out rows from the entire dataset where the tail value exceeded 3. Upon reviewing the graph only two rows were removed, which is a reasonable outcome in ensuring we do not lose valuable data. The tails data is a crucial variable as it provides insight into the recovery of PGM metals during the flotation process. At the Mogalakwena plant, tailings data is measured online using X-ray fluorescence (XRF) analysers and continuously transmitted to the SCADA system, enabling engineers to monitor the process in real time. Since grade data is derived from composite samples, the tail data serves as a key indicator for assessing the process. Similarly, the tail grade data is used here to identify periods of process issues by tracking times when data is either missing or displays abnormal values.

This method aligns with best practices as outlined in Brownlee's guidelines on data preparation, where managing missing values is critical to building robust machine learning models (Brownlee, 2020). Additionally, a variance check was performed to identify columns with constant values throughout the dataset, as such columns contribute little to the learning process and can be redundant for the algorithm. This step ensured that only informative features were retained, improving the model's efficiency and accuracy. These steps ensured that the dataset used for model training was both clean and representative of actual operational conditions.

Variable imputation techniques were thoroughly evaluated, with the KNN imputer using 5 neighbours selected as the most suitable method for handling missing data. This decision was informed by an analysis of the missing instances that remained after the initial data cleaning process. The analysis revealed variations in missing data across tanks, particularly for camera level readings. For example, Tank 11 had only 0.001% missing values, while Tank 14 had 0.02% missing instances. Additionally, many missing values occurred within overlapping time periods, highlighting the potential for reducing larger gaps by addressing smaller ones. Imputation was essential to retain valuable information and avoid losing significant portions of the dataset due to a single variable, which might not even hold substantial importance for the algorithm's generalisation. Given that the missing values constituted less than 1% of the overall dataset, this approach ensured that the dataset is complete, and its integrity is not compromised, supporting robust model performance during analysis and prediction. Close attention should be given to these variables during the feature selection phase of model development. This precaution ensures that the imputed data does not introduce bias, which could compromise the reliability and validity

of the model's results. By carefully evaluating the influence of these variables, we can mitigate the risk of skewed outcomes and ensure the model's robustness and predictive accuracy.

Table 2: *Missing values after the primary data cleaning*

	Column	Missing Count (%)
0	CAM-440-FT-008-FROTH-CAMERA.LEVEL. READING	0.03
1	MOGN.N-SEC.FT-CELL-440-FT-008.AIR-FLOW.PV	0.04
2	MOGN.N-SEC.SUMP-408-SU-002.DILUTION-WATER-FLOW.PV	0.02
3	MOGN.N-SEC.SUMP-408-SU-002.CONCENTRATION-01. PV	0.01
4	440SA031_P1/PV.V.1	0.01
5	CAM-440-FT-010-FROTH-CAMERA.LEVEL. READING	0.01
6	CAM-440-FT-011-FROTH-CAMERA.LEVEL. READING	0.001
7	CAM-440-FT-013-FROTH-CAMERA.LEVEL. READING	0.001
8	CAM-440-FT-014-FROTH-CAMERA.LEVEL. READING	0.02
9	CAM-440-FT-012-FROTH-CAMERA.LEVEL. READING	0.09

The dataset was split into input variables (X) and target variables (y) before performing any imputation or further processing. This approach is critical to avoid data leakage, a scenario where information from outside the training dataset inadvertently influences the model, leading to overly optimistic performance estimates. As suggested by Brownlee (2020), performing statistical calculations, such as mean imputation, on the entire dataset without separating X and y can result in the algorithm inadvertently gaining insight into the distribution of the target variable, which can compromise the model's ability to generalize to new data. Splitting the data beforehand ensured that all pre-processing steps including imputing missing values were performed independently on the training data, thus preserving the integrity of the model evaluation process.

Outlier detection was taken into account, but it was ultimately decided not to exclude outliers expect the extraordinary ones mainly since the data is presented as daily averages. The assumption is that averaging the grade data will mitigate the potential human error introduced during composite sample testing in the lab. This approach is also applied to process variables, under the belief that averaging helps to smooth out inconsistencies or anomalies that might arise from manual or procedural variances. In averaging the data, the influence of outliers or individual errors is reduced resulting in a more stable and representative dataset for modelling and analysis. Outliers in industrial data can often represent genuine process variations rather than errors, and their removal might lead to the loss of valuable information (Chandola et al., 2009). Therefore,

retaining these outliers allows the model to capture the full range of process behaviours which is crucial for developing a model that can generalize to various operating conditions.

Feature engineering was another critical component of the data pre-processing phase. This involved the creation of new variables that could potentially enhance model performance. For example, percentage recovery calculation, the mass flow rate was calculated using volumetric flow and density readings, which provided a more nuanced understanding of the process dynamics. Such engineered features can improve the model's ability to capture complex relationships between inputs and outputs (Domingos, 2012). These engineered features were included alongside the original variables to increase the predictive power of the models. Figure 27A and B illustrate how the percentage 4T recovery was calculated in python.

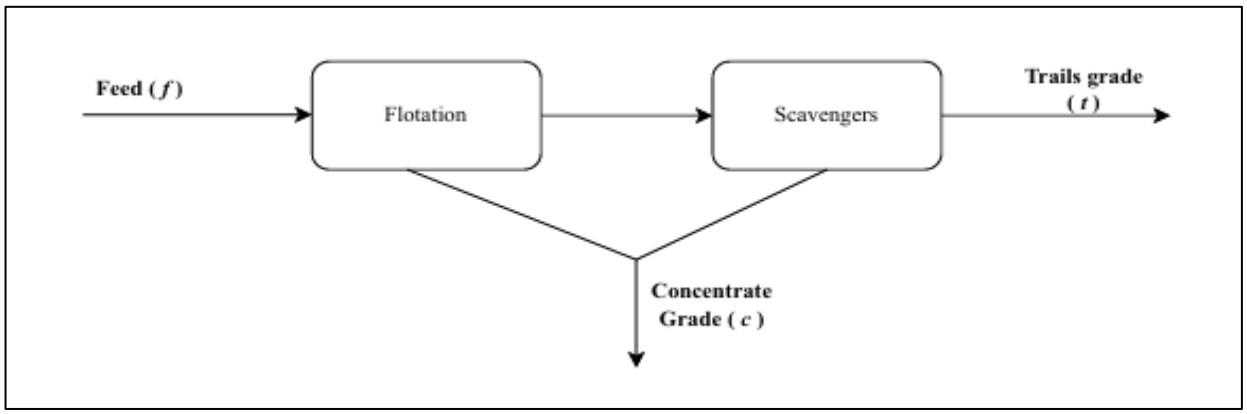


***Figure 25**: Simplified block flow diagram showing the feed, concentrate grade and the tails grade which are important parameters in recovery calculation.*

```
def calculate_recovery(c, f, t):
    """
    Calculate recovery percentage using the formula: R = 100 * c * (f - t) / (f * (c - t))

    """
    recovery = 100 * (c * (f - t) / (f * (c - t)))
    return recovery

c = Data_avg_final_cleaned1['L:MOGN.4T.Conc.Assay.Daily.Substitute']  # Concentrate assay
f = Data_avg_final_cleaned1['Estimate_PGMConc.Assay_calc_Samplehead']  # Feed assay
t = Data_avg_final_cleaned1['4T.Tails_Assay']  # Tailings assay

#  recovery percentage Calculation
recovery_4T_percentage = calculate_recovery(c, f, t)
print("Recovery Percentage:", recovery_4T_percentage)
Data_avg_final_cleaned1['recovery_4T_percentage']= recovery_4T_percentage
```

***Figure 26**: Recovery calculation Code*

where c, f, and t correspond to the concentrate, feed, and tailings assays respectively. This formula essentially compares the concentration in the concentrate against what is lost in the tailings, relative to the original concentration in the feed.

4.2.3 Feature Selection

Feature selection is a vital step in the modelling process as it reduces the dimensionality of the dataset and eliminates irrelevant or redundant variables that could introduce noise into the models. High-dimensional datasets often suffer from the curse of dimensionality, where the addition of more features increases the complexity of the model, often without a corresponding increase in performance (Bellman, 1961). Therefore, selecting the most relevant features is critical to building an efficient and accurate predictive model. Several feature selection techniques were rigorously evaluated to identify the most effective method for selecting relevant variables as inputs for training the machine learning algorithms.

Among the techniques explored were Manual Feature selection from process knowledge, Mutual Information (MI), Recursive Feature Elimination (RFE), and Minimum Redundancy Maximum Relevance (mRMR). Each of these methods offers distinct advantages firstly, MI measures the dependency between variables and the target, ensuring that the selected features have the highest predictive power, RFE recursively eliminates the least significant features to improve model accuracy and reduce overfitting, and mRMR balances relevance and redundancy by selecting features that are both highly relevant to the target and minimally redundant with each other. The comparative analysis of these techniques helped in pinpointing the most accurate and efficient approach for feature selection in this context.

While manual variable selection was initially guided by process knowledge, it was used primarily as a reference rather than the main feature selection strategy, as it resulted in lower model performance compared to recursive feature elimination. Moreover, manual selection has a higher risk of introducing bias, which can negatively affect the model’s ability to generalise to new data. To ensure a more objective and performance-driven approach, recursive feature elimination was adopted. Nonetheless, process knowledge remained important in interpreting results and validating feature relevance. The code for manual selection is provided in the appendix figure C9 to support the comparative assessment. RFE is a wrapper-based method that works by recursively removing the least important features and building a model on the remaining features (Guyon et al., 2002). This process continues until the optimal number of features is selected, aligning with a pre-defined threshold that indicates the point at which the features contribute positively to the model's generalisation capability. Iteratively removing less important features the algorithm refines its selection to include only those that enhance predictive accuracy while minimizing overfitting which leads to a more robust and reliable selection. This careful balance ensures that the final set of features improves the model's ability to generalize

well to unseen data, thereby boosting its overall performance. RFE was selected for its ability to improve model accuracy by focusing on the most influential variables, thereby enhancing the interpretability of the final model. Additionally, RFE is versatile and can be used with various base algorithms, making it a robust choice for feature selection.

The initial step in the Recursive Feature Elimination (RFE) process involved selecting the most suitable base model to identify the most relevant variables. Linear regression was initially employed as the base model due to its straightforwardness and ability to provide a clear understanding of variable importance. The RFE process was executed with linear regression to rank the features but due to its linear nature, it sometimes struggled to capture the complexities in the data. Subsequently, Random Forest was tested as a potential base model for RFE. Random Forest being an ensemble method is more adept at handling complex as well as non-linear relationships which makes it a strong candidate for feature selection in datasets with intricate interactions. The performance of the RFE using Random Forest was compared to that using linear regression and it was observed that Random Forest offered better results in identifying the most significant features. Therefore, Random Forest was chosen as the base model for the final RFE process, allowing for a more effective selection of the most influential variables for the predictive models. The python code created for the feature selection with random forest as the base model can be found in the appendix Figure C 1.

The feature selection process also involved a multicollinearity check using Variance Inflation Factor (VIF) analysis. Multicollinearity occurs when two or more predictor variables in a regression model are highly correlated, leading to unreliable estimates of regression coefficients (Mansfield & Helms, 1982). In this study, variables with a VIF greater than three were removed to ensure that the final model was free from multicollinearity issues. By addressing this issue, the study ensured that the selected features were independent of each other, thereby improving the stability and reliability of the model. The final dataset, after feature selection, included 18 variables identified as most relevant to predicting grade and recovery in the concentrator circuit. These variables were selected based on a combination of domain knowledge and statistical techniques, ensuring that the model captured the most critical aspects of the process.

The selected features were then used in the subsequent modelling phase, providing a strong foundation for building accurate and robust predictive models. The correlation matrix was generated to identify any variables that exhibit a correlation of 50% or higher with each other. If such highly correlated variables were found, the Variance Inflation Factor (VIF) threshold was lowered to reduce multicollinearity. Alternatively, if lowering the VIF threshold did not

sufficiently address the issue, the variable with the lowest correlation to the target variable was removed from the dataset. This step ensures that only the most relevant and independent features are retained, which enhances the model's performance and generalisation capabilities.

4.3 Model Development

4.3.1 Machine Learning Algorithms Investigated

A systematic approach was taken to model the relationship between the process variables and the target variables (grade and recovery). The modelling began with simpler and interpretable algorithms such as Linear Regression because they assist in establishing a baseline understanding of the data's linear characteristics. This was followed by more complex models, including Support Vector Machine (SVM), Multiple Layer Perceptron (MLP), Random Forest, and XGBoost, to explore and capture the potential non-linear and complex relationships within the data. The flow diagram below shows the improvement of the structure of the algorithms.

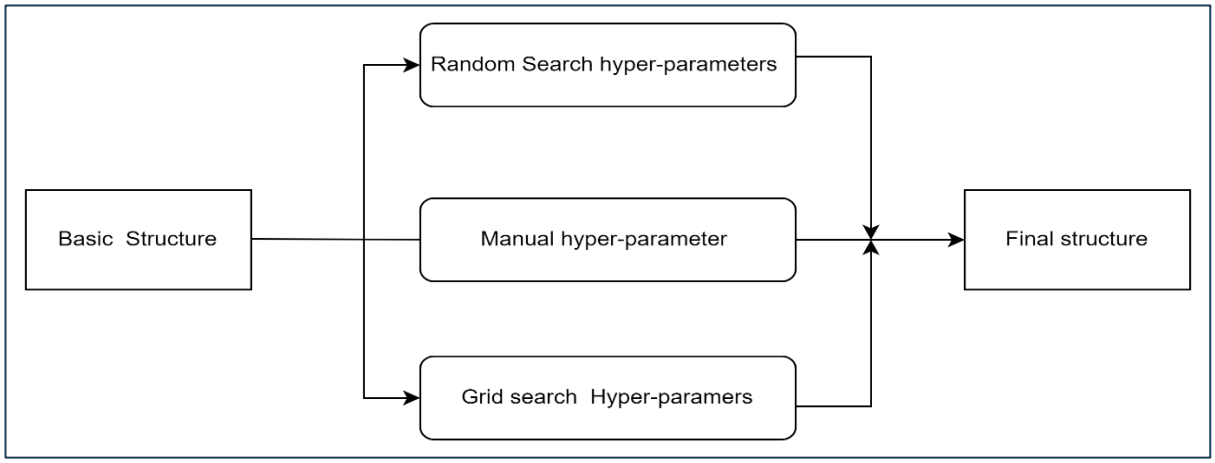


Figure 27: *Flow diagram illustrating the progressive improvement strategy of algorithm structure from basic to advanced hyperparameter tuning.*

Hyperparameter tuning in the algorithms was conducted using three methods. The first method involved manually setting the parameters, a trial-and-error process. In this approach, one parameter is adjusted at a time, and the algorithm's performance is evaluated. The process is repeated with different parameter settings to optimize the algorithm's output. The second method utilized was the random search technique also implemented in Python. This approach selects random combinations of hyperparameters from a predefined range and evaluates the algorithm's performance for each combination. Random search can be more efficient than manual tuning, as it explores a broader range of parameter values and can discover optimal configurations faster. The final method was grid search available as a built-in library in Python. Grid search exhaustively tests all possible combinations of hyperparameters from a predefined grid or matrix. Unlike random search, it systematically evaluates every combination, ensuring that the global

optimum is found, but it can be computationally expensive as it tests all parameter possibilities. All the above techniques were used in search of the best parameters.

The training process commenced with Linear Regression to determine if a linear model could adequately capture the relationship between the process variables and the target outcomes. Linear Regression is a fundamental technique that assumes a linear relationship between the input features and the target variable. Starting with the linear model, the study aimed to establish a baseline and assess whether the problem at hand exhibited significant linear characteristics. The simplicity of Linear Regression makes it highly interpretable, allowing for a clear understanding of the influence of each feature on the target variables. Additionally, the coefficients derived from this model provided valuable insights into which variables had the most linear impact on the grade and recovery metrics. In building the structure of the algorithms the first basic structure without hyperparameter tuning was the starting point. This was done to understand the general performance of each algorithm to aid easy interpretation and understanding of their performance.

Following the linear regression, the next step involved exploring more complex, non-linear relationships using a Support Vector Machine (SVM). SVM is particularly useful for high-dimensional data and can be employed to handle both linear and non-linear problems. In this study, the regression variant of SVMS known as Support Vector Regression (SVR), was used to model the continuous target variables. The SVM algorithm was chosen for its ability to find the optimal hyperplane that maximises the margin between different data points, effectively handling cases where the data is not linearly separable. The use of kernel functions allows the model to capture intricate patterns and interactions within the data that a simple linear model could not, thus providing a more nuanced understanding of the data's structure.

After establishing the baseline with Linear Regression and investigating the non-linear possibilities with SVM, the study then employed Multiple Layer Perceptron (MLP) to delve deeper into the complex, non-linear patterns that might exist within the data. MLP is a type of artificial neural network that is particularly suited for capturing complex, non-linear relationships between variables. The MLP model used in this study was designed with multiple hidden layers, each capable of learning intricate features from the input data. To prevent overfitting, which is a common issue in deep learning models, techniques such as batch normalisation, dropout layers, and early stopping were incorporated into the training process. MLP was selected for its flexibility and power in modelling the complex, multi-dimensional relationships that are characteristic of the data generated from the platinum concentrator process.

The Random Forest algorithm was then employed to further explore the non-linear interactions between variables. Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and aggregating their predictions to enhance model performance. This method is particularly effective in reducing variance and preventing overfitting, making it well-suited for handling the large and complex datasets characteristic of industrial processes. In this study, Randomised Search-CV was used to optimise hyperparameters such as the number of trees, maximum depth, and minimum samples for splitting. Random Forest was selected for its robustness and its ability to handle a large number of features while maintaining interpretability, providing insights into the importance of different variables in predicting grade and recovery.

Finally, XGBoost, an optimised gradient boosting algorithm, was employed for its efficiency and high accuracy in handling structured data. XGBoost builds upon the principles of boosting by sequentially constructing trees that correct the errors of the previous ones, thus creating a strong predictive model. The algorithm was fine-tuned using Grid-Search CV to identify the best hyperparameters, including learning rate, max depth, number of estimators, and subsampling rates.

XGBoost's ability to handle missing data, its regularisation techniques to prevent overfitting, and its capacity to model complex interactions between features made it a strong candidate for this study. In choosing XGBoost, the study aimed to capture any remaining non-linear patterns that other algorithms might struggle to capture which ensures that the final predictive model was both accurate and generalisable. This structured approach, starting with Linear Regression and progressing through increasingly complex models, allowed for a comprehensive understanding of the data and ensured that the most suitable model was selected for predicting grade and recovery in the platinum concentrator process.

4.3.2 Training and Validation

Training and validation are critical phases in the development of predictive models, ensuring that the models are not only accurate but also generalisable to new data. In this study, the data was split into training and test sets using different strategies to evaluate the robustness of the models under various conditions. The training set was used to build the model, while the test set was used to evaluate its performance on unseen data. The first strategy involved a temporal split, where data from January 2019 to mid-2023 was used for training, and data from mid-2023 to March 2024 was reserved for testing. This approach is particularly relevant for time-series data, where temporal dependencies are crucial (Hyndman & Athanasopoulos, 2018).

However, to ensure that the model could generalise across different operational scenarios, additional data splits were performed using random sampling and stratified sampling. In the random split, the data was randomly divided into training and test sets, which helps assess the model's ability to generalise across the entire dataset. Stratified sampling was used to ensure that both the training and test sets had similar distributions of key variables, which is important when dealing with imbalanced datasets (Kuhn & Johnson, 2013).

Cross-validation was employed to further validate the models and fine-tune their hyper-parameters. Specifically, k-fold cross-validation was used, where the data is divided into k subsets, and the model is trained and validated k times, each time using a different subset as the validation set and the remaining subsets as the training set (Stone, 1974). This approach provides a more reliable estimate of model performance compared to a single train-test split, as it ensures that every data point is used for both training and validation. In this study, 5-fold cross-validation was used for hyper-parameter tuning, with the aim of finding the optimal settings for each algorithm, such as the number of trees in Random Forest, the learning rate in XGBoost, and the number of hidden layers in MLP.

The research further goes to the integration of a Leuenberger Observer, which is a state estimation technique that improves the accuracy of predictions by accounting for process disturbances and measurement noise, which was first proposed by Leuenberger in 1971. The observer is incorporated into the machine learning models to enhance their predictive capability, especially under varying operating conditions. The integration of the Observer into the training process provides an additional layer of refinement, particularly in dealing with process disturbances and noise. It was used to update the state estimates continuously, providing the models with a dynamic correction mechanism that improved their predictive accuracy. This approach is particularly effective in industrial applications, where processes are subject to various disturbances that can affect sensor readings (Luenberger, 1971).

This technique further explored by Shardt et al. (2023) in their study on advanced soft-sensor systems, where they emphasised the importance of state estimation in enhancing the accuracy and reliability of process control, especially in complex industrial environments where process conditions are prone to rapid changes and disturbances. The application of the Luenberger Observer in this research is to ensure that the soft sensor adapts to real-time fluctuations and also resilient to the inherent variability of the concentrator circuit, thereby maintaining high prediction accuracy and operational stability.

4.3.3 Model Evaluation Metrics

The performance of the predictive models was evaluated using a comprehensive set of metrics to ensure a thorough assessment of their accuracy and reliability. The primary evaluation metrics used in this study were Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). These metrics were chosen because they provide different perspectives on model performance, each capturing a unique aspect of prediction accuracy and error magnitude.

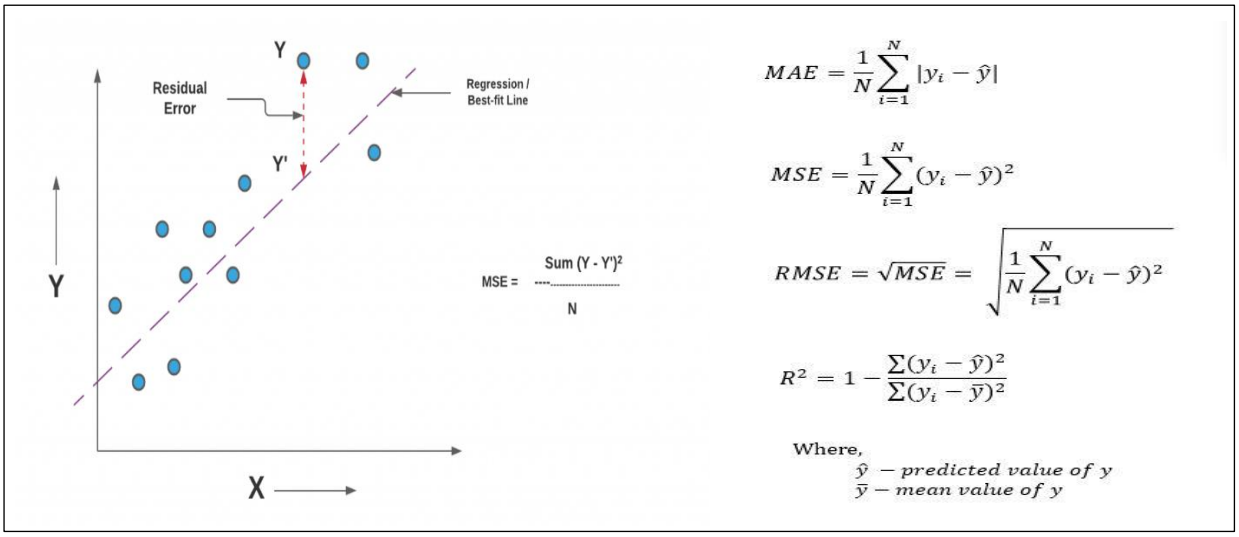


Figure 28: *Graph showing how the evaluation metrics is calculated from the model's predictions and actual target variable.*

MAE was also used as it provides a straightforward measure of the average error magnitude without considering the direction of the errors. Additionally, R-squared was employed to measure the proportion of variance in the target variable that the model explains. An R^2 value closer to 1 indicates that the model explains most of the variability in the target data, while a lower R^2 suggests that the model may not capture all relevant information. In addition to these standard metrics, the models were also evaluated based on their ability to predict recovery which were calculated using a specific formula that considers the predicted concentrate grade and tailings assay.

This custom metric provided additional insight into the model's practical utility in predicting key performance indicators in the platinum concentrator circuit. The combination of these evaluation metrics allowed for a thorough assessment of the models, ensuring that the final selection was based on a holistic view of their performance. Employing these multiple metrics the study was able to evaluate not only the accuracy but also the robustness and generalizability of the models which will ensure that the final model selection was based on comprehensive and rigorous criteria (Hyndman & Koehler, 2006).

Chapter 5: Data Analysis and Results

5.1 Introduction

This chapter presents a comprehensive analysis of the results obtained from the dataset and machine learning models described in Chapter 4. The analysis begins with an in-depth exploration of the dataset highlighting key trends and insights derived from preprocessing and feature selection. This step is crucial as effective data cleaning and selection directly impact model performance (Kuhn & Johnson, 2013). The study then evaluates and compares machine learning models using established performance metrics, assessing their predictive accuracy and generalisation capabilities. The results are critically discussed in the context of their implications for industrial flotation process optimisation.

A key focus is understanding how individual process parameters influence flotation performance. Since flotation is a highly dynamic and nonlinear process the identification of important variables is essential for accurate modelling and process control (Gupta and Yan, 2006) Integrating advanced data analysis techniques aids in uncovering patterns and trends within the flotation process. This study examines the effectiveness of different machine learning models in predicting the targeted process variables and thereby determining their suitability for soft sensor implementation. A soft sensor that is effectively deployed enhances the real-time monitoring and control of the flotation circuit. This helps in reducing reliance on physical instrumentation while improving process stability (Kadlec et al., 2009).

To enhance model robustness and reliability, various optimisation techniques are explored, including feature selection, hyperparameter tuning, and observer integration. Feature selection helps in reducing dimensionality while preserving critical information leading to improved computational efficiency and model interpretability (Guyon & De, 2003). Hyperparameter tuning is particularly vital for complex models like neural networks and tree-based algorithms as it directly influences predictive accuracy and generalisation (A Ilemobayo *et al.*, 2024b). The Leuenberger observer is known for its effectiveness in linear systems and will also be assessed for its potential to refine models' predictions particularly for linear and nonlinear algorithms. Integrating advanced data analysis techniques aids in uncovering patterns and trends within the flotation process which aids better decision-making and optimisation.

Comparing different machine learning models provides insights into their ability to handle nonlinear process dynamics which is regarded as a critical aspect given the complexity of mineral processing circuits (McCoy et al., 2019). The study also addresses common challenges in

industrial datasets, such as missing values and sensor inconsistencies which significantly impact model reliability. Implementing robust preprocessing and model adaptation techniques the research aims to develop a more stable and generalizable predictive framework. The insights gained from this study will be a guide for future research in developing more sophisticated soft sensors and adaptive modelling techniques which will assist in further bridging the gap between data science and metallurgical engineering.

5.2 Results of Data Analysis

Ore feed to the flotation circuit and the tails grade after flotation are critical process parameters used to calculate the mass balance around the flotation cell which makes them essential for analysing process performance. The ore feed is a critical factor in plant operations, as it provides essential insights into expected performance based on the material supplied to the circuit. Additionally, monitoring and analysing the ore feed enables operators to trace operational disruptions such as unplanned shutdowns back to potential root causes. The flow of the ore feed into the flotation circuit after data cleaning is presented in figure 31.

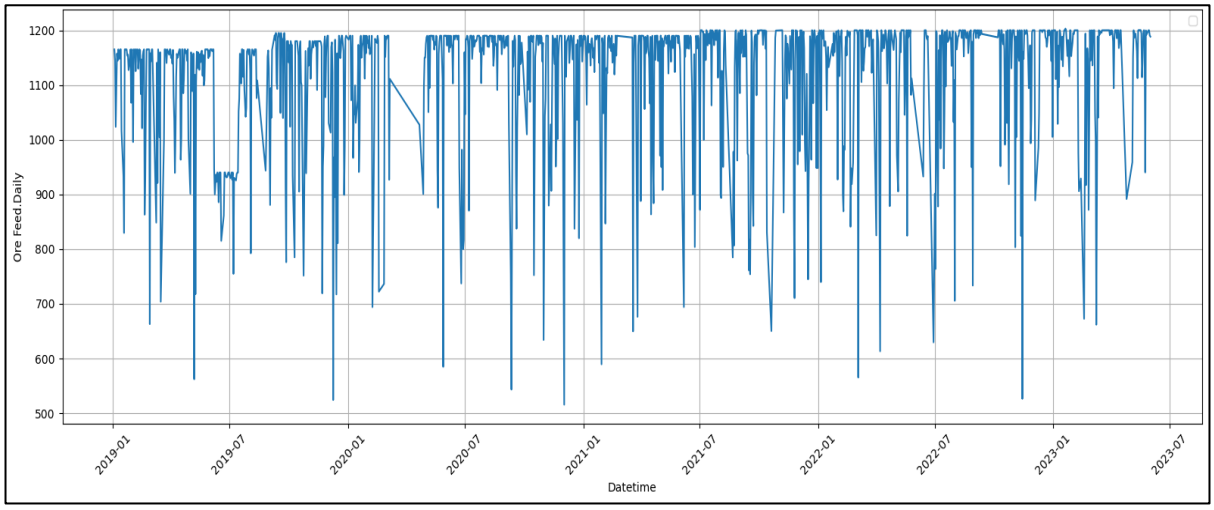


Figure 29: *Ore feed to the flotation circuit over the period after data cleaning.*

The raw ore feed dataset contained numerous zero values prior to preprocessing, as illustrated in Figure 24. These zeros correspond to periods of operational downtime or system errors, indicating intervals when the process was inactive or malfunctioning. Following data cleaning, the dataset was filtered to include only values within the operational range of 500 t/h to the setpoint of 1200 t/h. This refinement ensures the use of reliable process data, confirming the plant was actively running during these periods. In excluding downtime intervals, the cleaned dataset mitigates potential distortions in mass balance calculations and enhances the accuracy of flotation circuit performance evaluations. Mass balance calculations rely heavily on accurately

tracking material inputs and outputs within the system. To ensure consistency in these calculations the tails grade a key output metric is presented in Figure 32 after rigorous data cleaning.

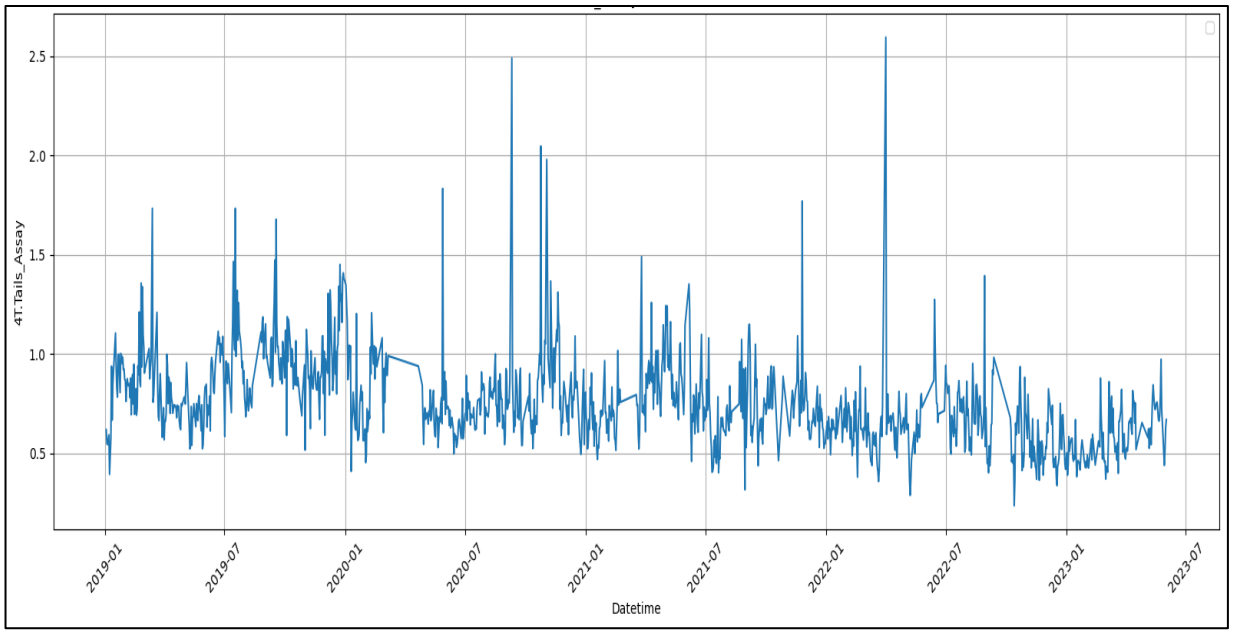


Figure 30: *Tails grade after data cleaning overtime for the flotation circuit*

Conversely, figure 25 depicts the tails grade before preprocessing which showed notable spike between the 0 to 250 time period. This anomality could result from sensor malfunctions, incorrect data storage, or process disturbances leading to suboptimal yield. Such spikes introduce uncertainty in the data and can hinder the reliability of analyses unless appropriately addressed. The lack of additional information from engineers or process operators present at the plant during the occurrence further complicates identifying the root cause of these anomalies. These observations underscore the importance of robust data validation, error handling, and collaboration with operational teams to ensure accurate and actionable process analyses.

Furthermore, analysing the target variables which is grade and recovery is crucial for ensuring alignment with the feed and tails datasets. Synchronizing these variables help validate that all plotted periods reflect consistent operational conditions free from downtime or anomalies. This alignment not only strengthens the integrity of the analysis but also highlights regions most impacted by data cleaning such as intervals where extreme values or disruptions were removed. Figures 33 and 34 illustrate the cleaned 4T concentrate grade and recovery plots respectively which are now constrained to the active production phases of 500 –1200 t/h.

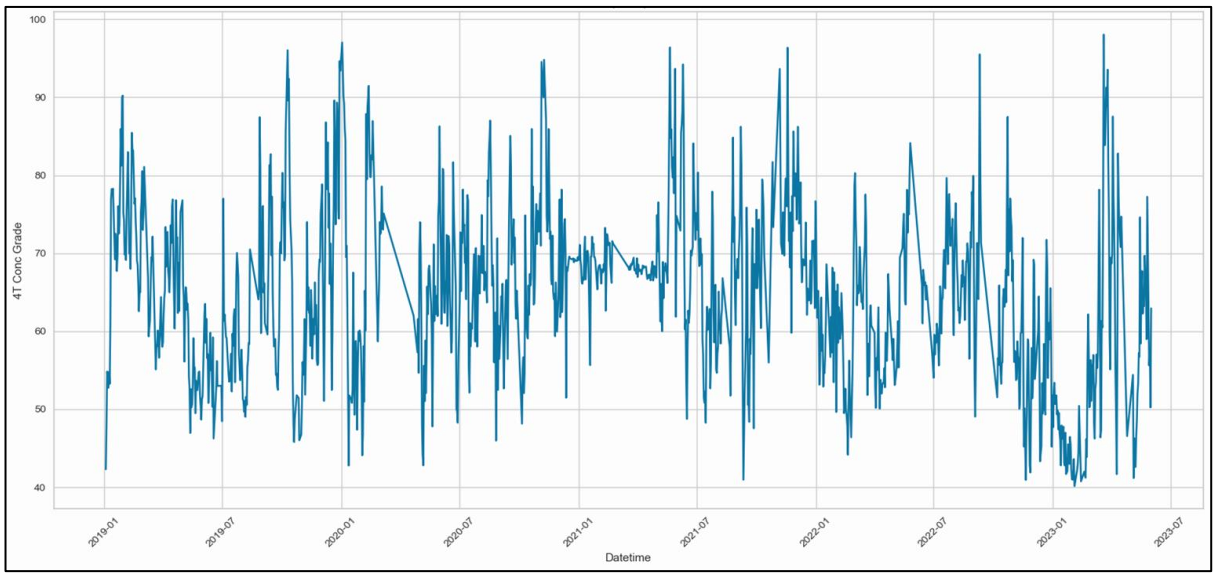


Figure 31: *4T PGMs Concentrate grade over time after data cleaning over time.*

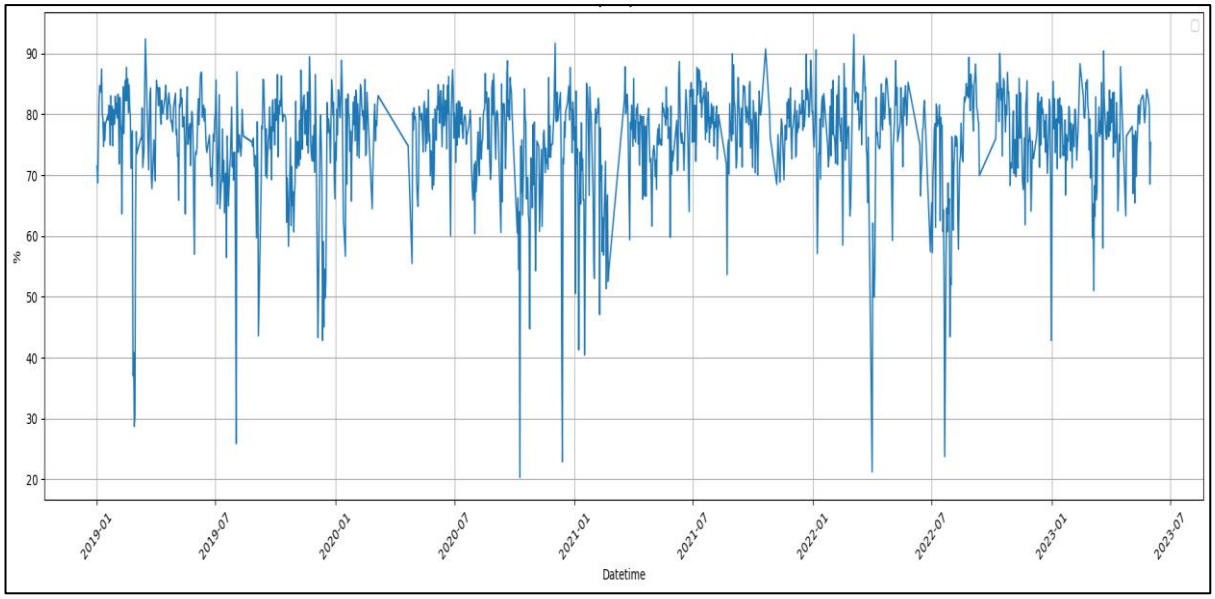


Figure 32: *4T recovery plot after data cleaning overtime.*

The cleaned grade and recovery plots offer a more coherent depiction of the second rougher banks behaviour over time compared to the raw grade dataset in Figures 25. These refined visuals highlight persistent challenges in industrial data, including outliers and temporal inconsistencies. For instance, the cleaned grade plot reveals fluctuations between 40 and 100 g/t, reflecting natural variability driven by factors such as ore composition and process adjustments. In contrast, the raw grade plot (Figure 25) contains implausible values exceeding 100 g/t which lack corresponding spikes in recovery. Such anomalies likely originate from sensor malfunctions, data entry errors, or calibration issues rather than actual process behaviour which underscores the need for rigorous preprocessing (Ge et al., 2017).

Similarly, the recovery plot exhibits abrupt drops to 0% and transient spikes to nearly 100%, indicative of process downtime, sensor failures, or measurement noise common issues in industrial time-series data (Bergmeir & Benitez, 2012). Notably, both cleaned plots feature a gap between January–July 2020 which is a result from the removal of non-operational or erroneous periods during preprocessing. While grade and recovery have an inversely proportional relationship, their temporal trends diverge slightly due to their distinct sensitivities to process variables and differences in their calculation methods. The graphs show that when the grade trend declines, the recovery trend moves in the opposite direction, though at a different rate of change. This relationship is evident in Figures 33 and 34, where the grade starts at a low value of approximately 40 g/t, while the recovery begins at a relatively high value of around 70%. After July 2022, the grade increases for a while before declining again, whereas recovery follows the opposite trend, initially declining and then increasing.

The raw data's unrealistic values risk distorting mass balance calculations and generalisation of the predictive models which could potentially lead to flawed conclusions about circuit performance (Srinivasan et al., 2008). Such noise and gaps obscure true process dynamics, reducing the reliability of machine learning models trained on uncleaned data (Qin & Chiang, 2019). These challenges emphasize the necessity of robust preprocessing techniques, including outlier detection, noise reduction, and gap imputation, to enhance data quality (Hyndman & Athanasopoulos, 2018). Addressing these issues enables a more accurate analyses that that leads to insights that reflect actual process behaviour rather than measurement artifacts. This approach aligns with best practices in industrial data analytics where preprocessing is important for deriving actionable insights that support process optimisation and operational decision-making.

5.2.1 Descriptive Statistics and Data Insights

his section summarizes the outcomes of the exploratory data analysis (EDA) performed during the data preprocessing phase. Key statistical metrics, including mean, median, standard deviation, and variable distribution patterns, are evaluated to provide a comprehensive understanding of the dataset's structure. Visual tools such as histograms, box plots, and correlation heatmaps are utilized to reveal relationships between variables, identify underlying patterns, and pinpoint potential anomalies. Detailed insights are drawn from the analysis, focusing on trends, outliers, and the presence of missing data, enabling a deeper understanding of the dataset's characteristics and guiding subsequent modelling decisions. Figure 34 illustrates the dynamic variations of key process variables over the selected time period. The different tag descriptions can be found in table A2 in the appendix.

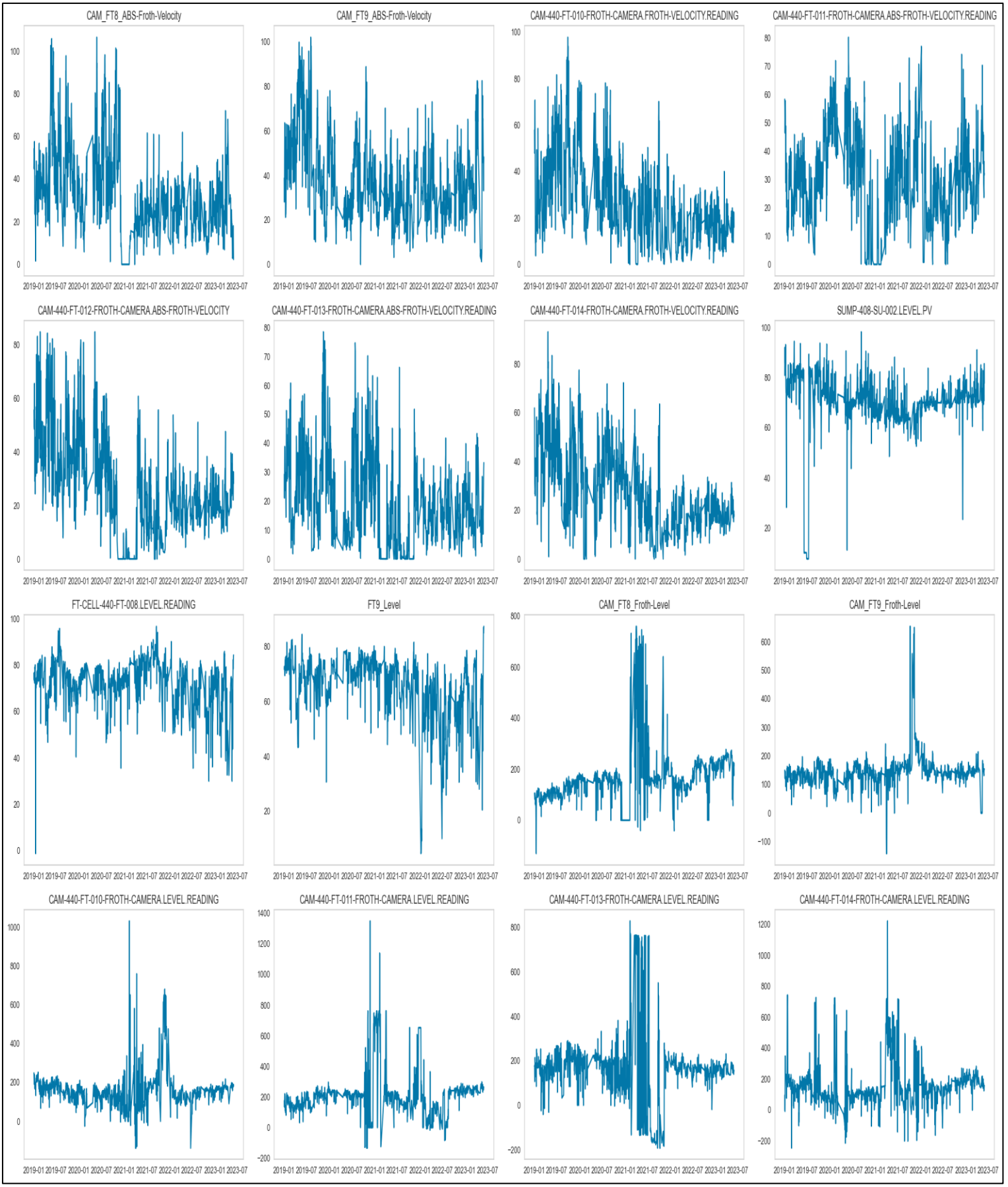


Figure 33: *Trends of froth velocity, froth level, and cell level within a flotation circuit over time.*

Visualizing these input variables prior to algorithm training is crucial to identify and exclude any redundant variables such as those exhibiting constant values. The first two rows in the figure display the absolute froth velocity across cells 8 to 14, recorded by a wall-mounted camera, while the bottom last plot illustrates the sump tank level. The subsequent rows 3 and 4 represent the levels of cells 8 and 9, followed by the froth levels across cells 8 to 14. The froth level data for tank 12 has been omitted from the analysis due to excessive missing values.

Notably, significant variability is observed between cells 8 and 9 in the secondary line aligning with the findings of Steyn & Sandroc (2021), who reported that maximum recovery and grade are achieved within these cells before declining in downstream cells. Spikes are observed in the froth level across the cells at consistent intervals indicating the presence of data patterns that are not apparent in the froth velocity or cell level graphs. Such anomalies should be thoroughly verified before analysis as they may introduce bias which could be potentially stemming from faults in the level-measuring instrument that may lead to the recording of unreliable data or outliers.

The presented plots facilitate informed decision-making during variable selection to optimize computational efficiency. For instance, it may be reasonable to select the froth level of cell 8 as a representative variable for the froth level process dynamics across all cells during the analysed period. This choice is supported by the observation that froth levels in Cell 8 exhibit a distinct increasing trend, while froth levels in the other cells fluctuate around their respective setpoints. This may indicate significant variability in those cells or suggest abnormal operation, possibly due to equipment failure. Such observations highlight one of the key challenges of working with historical data, it often provides an incomplete picture of the actual operating conditions.

The froth velocity varies significantly around cell 8 and 9 while cell 11 to 13 share the same variation the 10 and 14 having different unique differences. This also highlight the possibility to choose only three variables to be a representation of froth velocity instead of the whole set. It is also important to visualize the data split between the training and testing sets. This ensures that the unseen testing data closely represents the training set which is crucial for achieving accurate and reliable results.

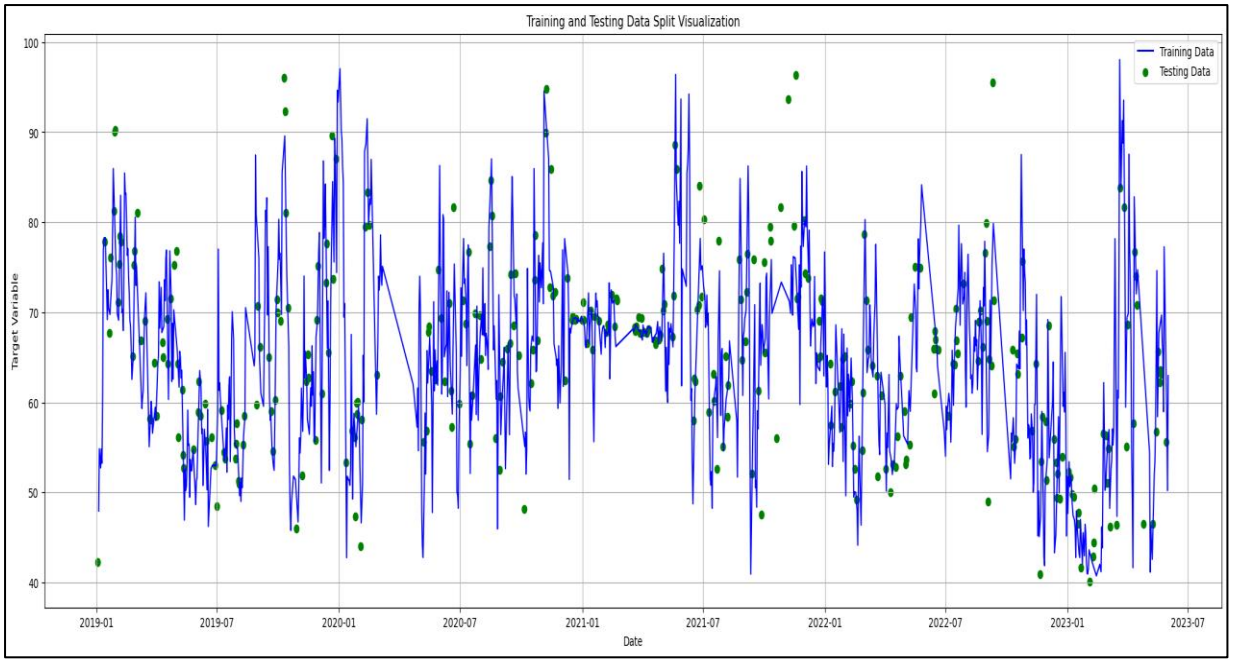


Figure 34:*Training and Testing data split plot obtained from sequential data split.*

The blue plot in figure 36 represents the training data while the green scatter points indicate the testing set over the selected period. The testing set was created by selecting data from seven random days each month. This approach ensures that the testing set provides a good representation of the target material and captures its key characteristics and variability. The scatter plot points are distributed evenly around the blue line which indicates that the testing data closely aligns with the training data. This alignment builds confidence that strong algorithm performance on the testing set would confirm the models' ability to learn meaningful insights from the data which would assist it in making reliable predictions of the PGMs grade and recovery.

The green points equal to or above 90 which are scattered far from the blue plot pose a significant challenge for the algorithm. Successfully predicting these outlier points would indicate that the training set's input variables contain valuable information that allows the algorithm to generalize beyond the seen data. While this represents a rigorous test requiring the algorithm to make predictions for scenarios it has not encountered. If this is possible it will align with the principles of traditional methods where spikes were analysed through statistical calculations to derive equations of a particular trend. The hope is that the algorithm can infer meaningful patterns from training points within a similar range which validates its ability to capture the underlying dynamics of the data.

5.2.2 Impact of Data Preprocessing

Data preprocessing plays a pivotal role in ensuring the accuracy and reliability of machine learning models particularly in complex systems like flotation circuits. The quality of the input data directly impacts the model's ability to learn meaningful relationships and generalize effectively. In flotation circuits data variability and anomalies such as noise, outliers, and missing values are common. Preprocessing is critical for standardizing inputs, reducing redundancy, and enhancing the interpretability of results. This section evaluates how preprocessing techniques such as normalization, outlier detection, imputation of missing values, and variable selection influence the performance of predictive models.

The analysis also examines the downstream effects on model accuracy, robustness, and computational efficiency which provides insights into the extent to which preprocessing contributes to improving predictions of grade and recovery. In quantifying the impact of these techniques proves the necessity of a well-designed preprocessing pipeline to optimize the predictive performance and operational applicability of machine learning models in metallurgical engineering. The figure below represents the dataset obtained after a rigorous data cleaning process. This is a critical step in ensuring that the data is of good quality and reliability to facilitate powerful predictive models. Initially, the dataset consisted of 70 columns and 1980 rows. During preprocessing 20 columns and 688 rows were eliminated from the dataset. Process variables such as froth mean bubble size contained significant number of missing values and were removed from the dataset to preserve the data integrity.

All the pulp process data obtained from the existing sensors was also removed from the dataset due to excessive missing values. This suggests potential issues with pulp sensor reliability, either due to sensor malfunctions or poor maintenance. The high rate of missing data per process variable limited its usability in model training, reinforcing the need for better data acquisition and sensor upkeep. Additional details about the variables that were removed, and their respective amounts of missing values can be found in Table A1 in the appendix. This practice is in line with what is emphasized in literature that addressing missing data is crucial, as unprocessed gaps can introduce bias and impair model performance (Little & Rubin, 2019). Additionally, variables showing no variance were excluded from the data as such features fail to capture any system dynamics or contribute meaningful information to model learning. In research, removing low-variance variables is a widely adopted technique to reduce model complexity and enhance interpretability (Guyon & Elisseefl, 2006).

In addition to missing values, certain variables exhibited no variation over the observed timeframe maintaining constant values throughout. Such variables were removed from the dataset as they contribute no meaningful information to the learning process and only increase computational complexity (Guyon & Elisseeff, 2003). For instance, the PGM concentrate mass daily estimate as illustrated in Figure 37 remained unchanged over time and was therefore excluded.

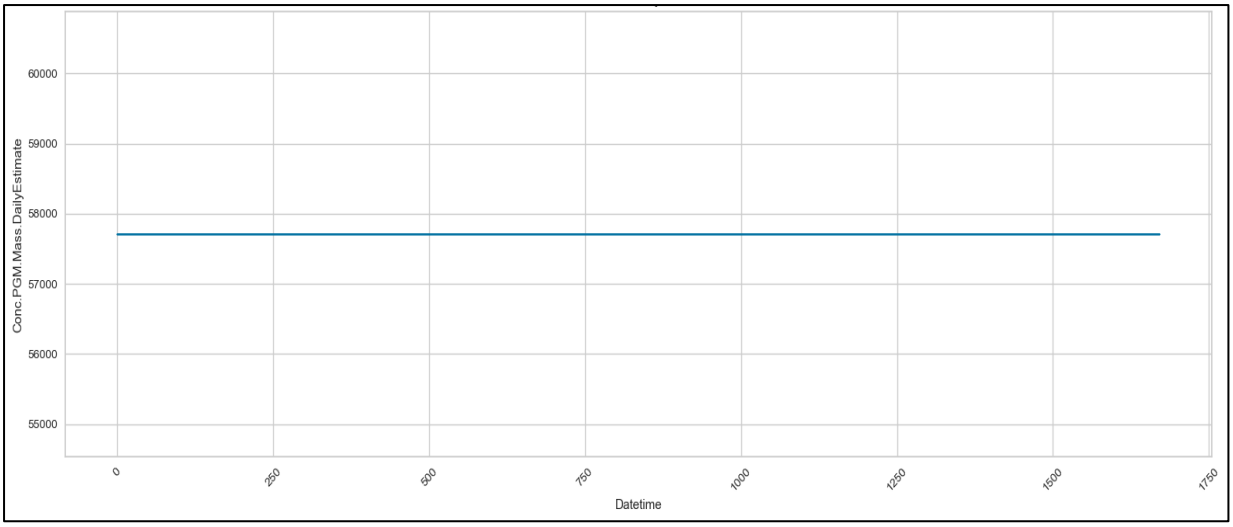


Figure 35: *Concentrate PGMs mass daily estimate over time.*

In a flotation cell, setpoints for specific parameters play a crucial role in process control strategies. These setpoints help monitor deviations from normal operating conditions and ensure process stability. Process engineers analyse these deviations between setpoints and actual values to assess the process performance and detect anomalies in order to implement corrective actions to optimize recovery and grade. Attention should be given to these types of variables as they may contain valuable information about specific process states and deviations from optimal conditions. However, if a setpoint remains constant over time without meaningful variation, it contributes little to model learning and should be removed.

As a result, the cleaned dataset comprises of 50 key process variables over 1292 time-stamped entries. This were measured between the 3rd of January 2019 up to the 1st of June 2023 as presented in figure 38 below. It includes critical metrics such as mass pull, froth velocity, reagent flows, and airflows which are essential for monitoring and optimising flotation efficiency. Froth data, including froth velocity and volume enhance the dataset by providing detailed insights into froth dynamics which strongly influence grade and recovery. The time-series structure of the data enables in-depth analysis which offers opportunities to model process dynamics, identify inefficiencies, and areas of optimisation of the flotation parameters in real time.

Applying and ensuring accurate data preprocessing methods is vital in flotation circuit due to the complex nature of the process hence applying data-driven decisions will significantly impact process optimisation and operational outcomes (Jadhav et al., 2019). This refined dataset provides a more representative foundation for subsequent modelling supporting the development of robust and generalizable insights. All process variables are represented in the float64 format which is Python's standard for numerical data. This ensures high precision in computations and maintains consistency across all variables for efficient processing in machine learning models.

```
DatetimeIndex: 1292 entries, 2019-01-03 06:00:01 to 2023-06-01 06:00:01
Data columns (total 50 columns):
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 0   Ore Feed                                              1292 non-null   float64
 1   Estimate_PGMConc.Assay_calc_Samplehead                1292 non-null   float64
 2   Rougher_Surge_Deliveryflow_cntrl(m3/h)                1292 non-null   float64
 3   FT8_Mass-pull_Reading                                 1292 non-null   float64
 4   FT9_Mass-pull_Reading                                 1292 non-null   float64
 5   CAM_FT8_ABS-Froth-Velocity                            1292 non-null   float64
 6   CAM_FT9_ABS-Froth-Velocity                            1292 non-null   float64
 7   FT8_Flow_Frother_Reading                              1292 non-null   float64
 8   SUMP-408-FLOW-COLLECTOR.PV                            1292 non-null   float64
 9   SUMP-408-FLOW-Promoter.PV                             1292 non-null   float64
 10  PUMP-404PP202-03-FLOW.PV                              1292 non-null   float64
 11  CAM_FT8_Froth-Level                                   1292 non-null   float64
 12  CAM_FT9_Froth-Level                                   1292 non-null   float64
 13  FT9_Airfolw                                           1292 non-null   float64
 14  FT-CELL-440-FT-008.FLOW-COLLECTOR.READING             1292 non-null   float64
 15  FT-CELL-440-FT-008.FLOW-DEPRESSANT.READING            1292 non-null   float64
 16  FT-CELL-440-FT-008.FLOW-MP-PV.RAW                     1292 non-null   float64
 17  FT-CELL-440-FT-008.FLOW-PROMOTER.READING              1292 non-null   float64
 18  FT-CELL-440-FT-008.LEVEL.READING                      1292 non-null   float64
 19  FT-CELL-440-FT-008.RESIDENCE-TIME.RAW                 1292 non-null   float64
 20  MOGN.N-SEC.FT-CELL-440-FT-008.AIR-FLOW.PV             1292 non-null   float64
 21  FT9_Level                                             1292 non-null   float64
 22  FT-CELL-440-FT-009.RESIDENCE-TIME.RAW                 1292 non-null   float64
 23  MOGN.N-SEC.SUMP-408-SU-002.LEVEL.PV                   1292 non-null   float64
 24  MOGN.N-SEC.SUMP-408-SU-002.DILUTION-WATER-FLOW.PV     1292 non-null   float64
 25  MOGN.N-SEC.SUMP-408-SU-002.CONCENTRATION-01.PV        1292 non-null   float64
 26  440SA031_P1/PV.V.1                                    1292 non-null   float64
 27  440FT008_1/DOL.STA_IPER                               1292 non-null   float64
 28  440FT009_1/DOL.STA_IPER                               1292 non-null   float64
 29  FT-CELL-440-FT-010.AIR-FLOW.READING                   1292 non-null   float64
 30  FT-CELL-440-FT-011.AIR-FLOW.READING                   1292 non-null   float64
 31  FT-CELL-440-FT-012.AIR-FLOW.READING                   1292 non-null   float64
 32  FT-CELL-440-FT-013.AIR-FLOW.READING                   1292 non-null   float64
 33  CAM-440-FT-010-FROTH-CAMERA.FROTH-VELOCITY.READING    1292 non-null   float64
 34  CAM-440-FT-010-FROTH-CAMERA.LEVEL.READING             1292 non-null   float64
 35  CAM-440-FT-011-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING 1292 non-null  float64
 36  CAM-440-FT-011-FROTH-CAMERA.FROTH-VELOCITY.READING    1292 non-null   float64
 37  CAM-440-FT-011-FROTH-CAMERA.LEVEL.READING             1292 non-null   float64
 38  CAM-440-FT-010-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING 1292 non-null float64
 39  CAM-440-FT-013-FROTH-CAMERA.FROTH-VELOCITY.READING    1292 non-null   float64
 40  CAM-440-FT-013-FROTH-CAMERA.LEVEL.READING             1292 non-null   float64
 41  CAM-440-FT-014-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING 1292 non-null  float64
 42  CAM-440-FT-014-FROTH-CAMERA.FROTH-VELOCITY.READING    1292 non-null   float64
 43  CAM-440-FT-014-FROTH-CAMERA.LEVEL.READING             1292 non-null   float64
 44  CAM-440-FT-012-FROTH-CAMERA.LEVEL.READING             1292 non-null   float64
 45  CAM-440-FT-012-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING 1292 non-null float64
 46  CAM-440-FT-013-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING 1292 non-null float64
 47  CAM-440-FT-014-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING 1292 non-null float64
 48  CAM-440-FT-012-FROTH-CAMERA.FROTH-VELOCITY.READING    1292 non-null   float64
 49  CAM-440-FT-012-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING 1292 non-null  float64
dtypes: float64(50)
memory usage: 514.8 KB
```

Figure 36: *Dataset after data pre-processing*

The figure below presents the correlation matrix illustrating the relationships between variables before and after preprocessing. This visualization is essential in understanding how different process variables interact, identifying strong correlations, and detecting potential multicollinearity, which can affect model performance (Dormann et al., 2013). This analysis makes it easier to select the most relevant features which assist in reducing redundancy and improving computational efficiency in machine learning applications.

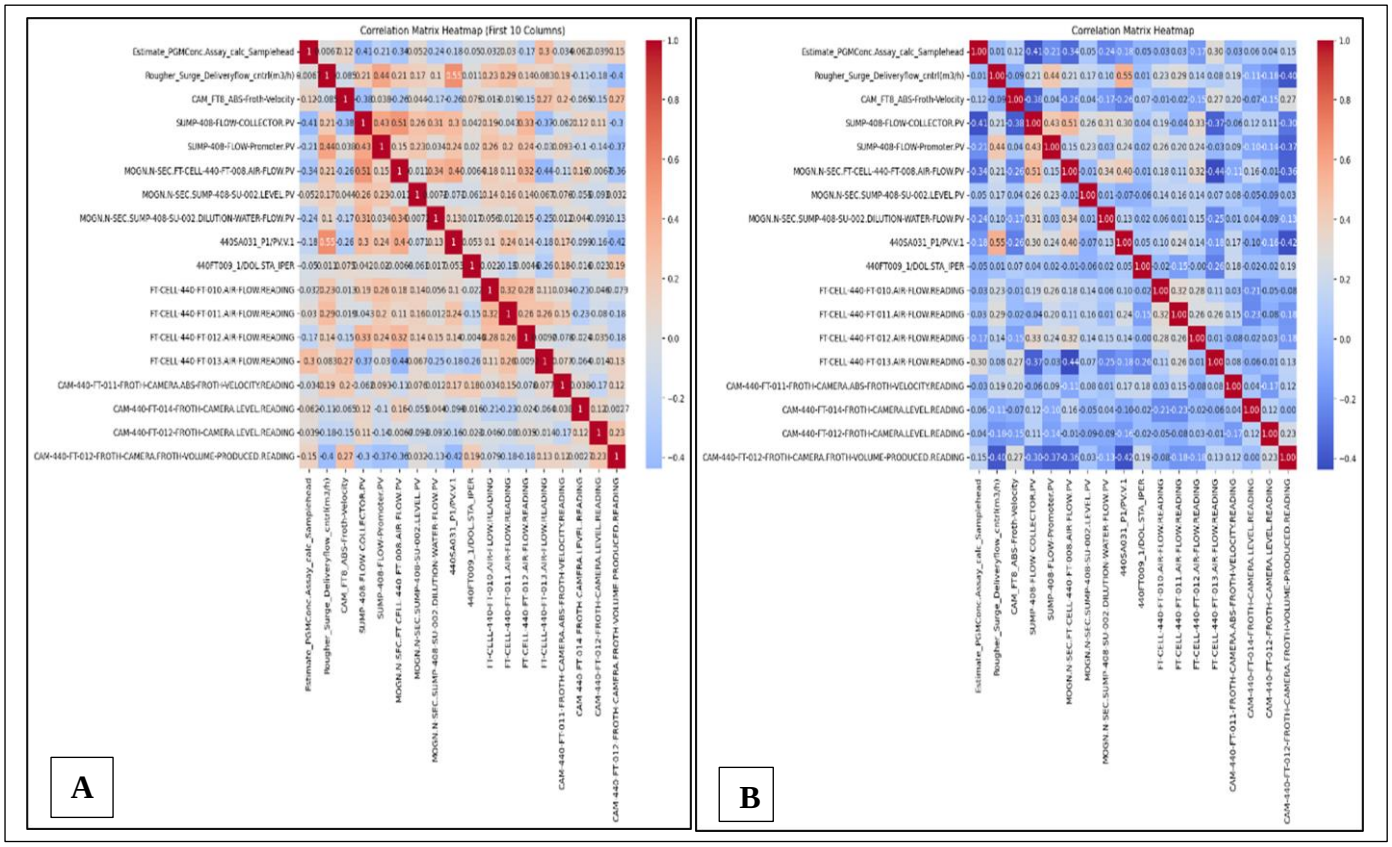


Figure 37: *Correlation matrix A and B representing variables relationship before and after data processing respectively.*

The correlation matrices highlight the impact of preprocessing steps, such as scaling, normalization, and outlier removal as well as multicollinearity on variables. The matrix on the left (A) represents the original data before feature selection and Variance Inflation Factor analysis which are used to address multicollinearity. This is evident in the extensive brown regions in the left matrix indicating high correlations. In contrast to the matrix on the right which shows the data after preprocessing. The right matrix (B) has significantly fewer brown regions and more blue areas, reflecting reduced correlations. The scale located on the far side of the figure indicates that blue represents values between -0.4 and 0.3 while brown ranges from 0.3 to 1. The diagonal red line across both matrices represents perfect correlation as it compares the same variable on both axes.

After preprocessing the right side has a few of the light brown regions scatted both above and below the diagonal which indicates minimal residual correlation. The remaining highest correlation value between the variables was 0.45 which is below the 50% threshold, allowing the data to proceed to model training. This reduction in multicollinearity and clarification of significant relationships improves the presentation of individual variable in the learning process of the model.

5.2.3 Feature Selection Results

It is essential to assess the importance of each variable to ensure that the selected features are relevant and non-redundant for the algorithm. Feature importance serves as a guiding metric to validate the choice of variables enhancing interpretability and understanding of the model performance. The following figure illustrates the feature importance analysis obtained through using the F-score evaluation criterion. F-score indicates the frequency with which a feature is utilized for splitting during the training process mainly reflecting its relative contribution to the model's decision-making. In prioritizing variables with higher F-scores ensures that critical predictors are retained while reducing noise and redundancy in the dataset.

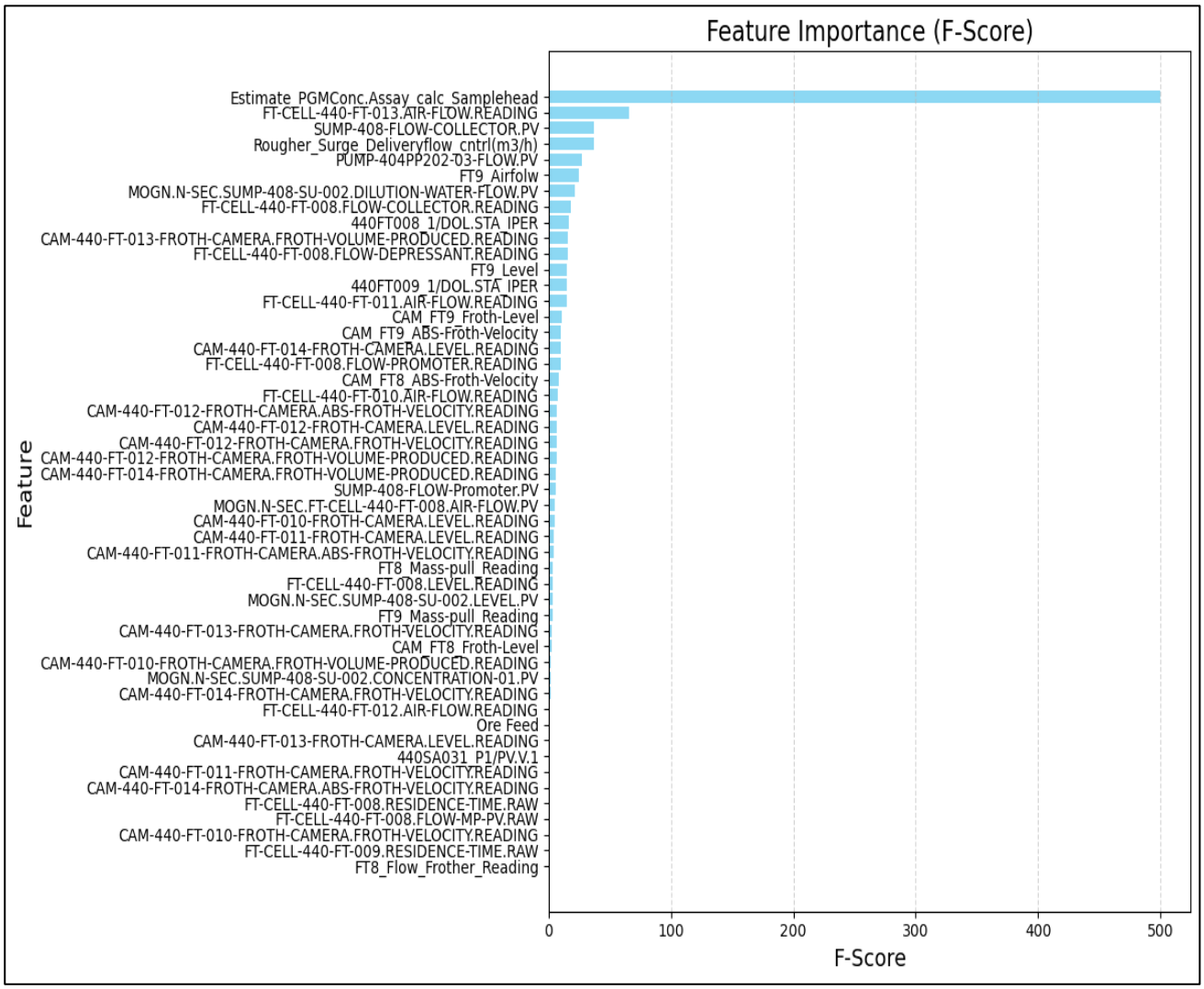


Figure 38: *F-score of variables before feature selection*

The features are ranked in descending order of their relative importance scores as depicted in Figure 40. According to the findings of Ding (2009) the higher F-score values denote greater importance, indicating a feature's significant contribution to the model's decision-making process. In this analysis, the Estimated Ore Sample Head Assay is identified as the most critical feature while the Frother Flow Reading for Cell 8 holds the lowest importance. Notably, the range between ore sample head and the cell 8 flow promoter highlights the subset of 18 variables that might be selected as the threshold for feature selection is set at 18.

This range serves as a critical reference point, ensuring that only the most influential variables are retained. These selected features are deemed valuable due to their significant impact on model predictions, reinforcing the importance of prioritising key features. This ranking approach provides a robust foundation for effective model training while minimizing the inclusion of redundant or less impactful variables. The most important features identified during the selection process are presented in the following table 3.

Table 3: *Selected features by RFE from high to low and the VIF score*

No	Variable Name	VIF Values
1	estimate_pgmconc.assay_calc_samplehead	1.41
2	rougher_surge_deliveryflow_cntrl(m3/h)	2.03
3	cam_ft8_abs-froth-velocity	1.50
4	sump-408-flow-collector.pv	2.77
5	sump-408-flow-promoter.pv	1.89
6	mogn.n-sec.ft-cell-440-ft-008.air-flow.pv	2.19
7	mogn.n-sec.sump-408-su-002.level.pv	1.38
8	mogn.n-sec.sump-408-su-002.dilution-water-flow.	1.25
9	440sa031_p1/pv.v.1	2.06
10	440ft009_1/dol.sta_iper	1.25
11	ft-cell-440-ft-010.air-flow.reading	1.33
12	ft-cell-440-ft-011.air-flow.reading	1.57
13	ft-cell-440-ft-012.air-flow.reading	1.35
14	ft-cell-440-ft-013.air-flow.reading	1.80
15	cam-440-ft-011-froth-camera.abs-froth-velocity.reading	1.31
16	cam-440-ft-014-froth-camera.level.reading	1.27
17	cam-440-ft-012-froth-camera.level.reading	1.28
18	cam-440-ft-012-froth-camera.froth-volume-produced.reading	1.79

The table above summarizes the 18 selected variables using RFE utilising random Forest as a base algorithm. The actual tag description is provided in Table A2 of the appendix. The selection was guided by F-score analysis and VIF which are important metrics in ensuring robust selection. This informs on the relevance of a particular variable to the predictive task of the model. Feature importance rankings discussed above was used as a reference guid keeping in mind that some variable will be deselected even if it has high feature importance but showing multi collinearity with a lot of variables will be deselected since the threshold for VIF was 3.

The selected variables reflect a balanced representation of both upstream and in-cell flotation circuit, in line with flotation theory. The inclusion of froth velocity and froth volume (e.g., variables 3, 15, and 18) aligns with the theory that froth behaviour is directly correlated to particle recovery and grade, as it indicates bubble-particle attachment and froth stability. The air flow rates across multiple flotation cells (variables 6, 11–14) are vital, as air dispersion strongly influences bubble size and residence time, which are key to particle-bubble collision probability. Sump levels and dilution flows (variables 4, 5, 7, 8) reflect upstream conditioning and residence time, affecting pulp density and reagent effectiveness, which in turn govern flotation selectivity and kinetics. Finally, PGM head grade (variable 1) is a crucial benchmark, as it links the feed quality to observed responses, aiding in understanding recovery efficiency across varying ore feed and types.

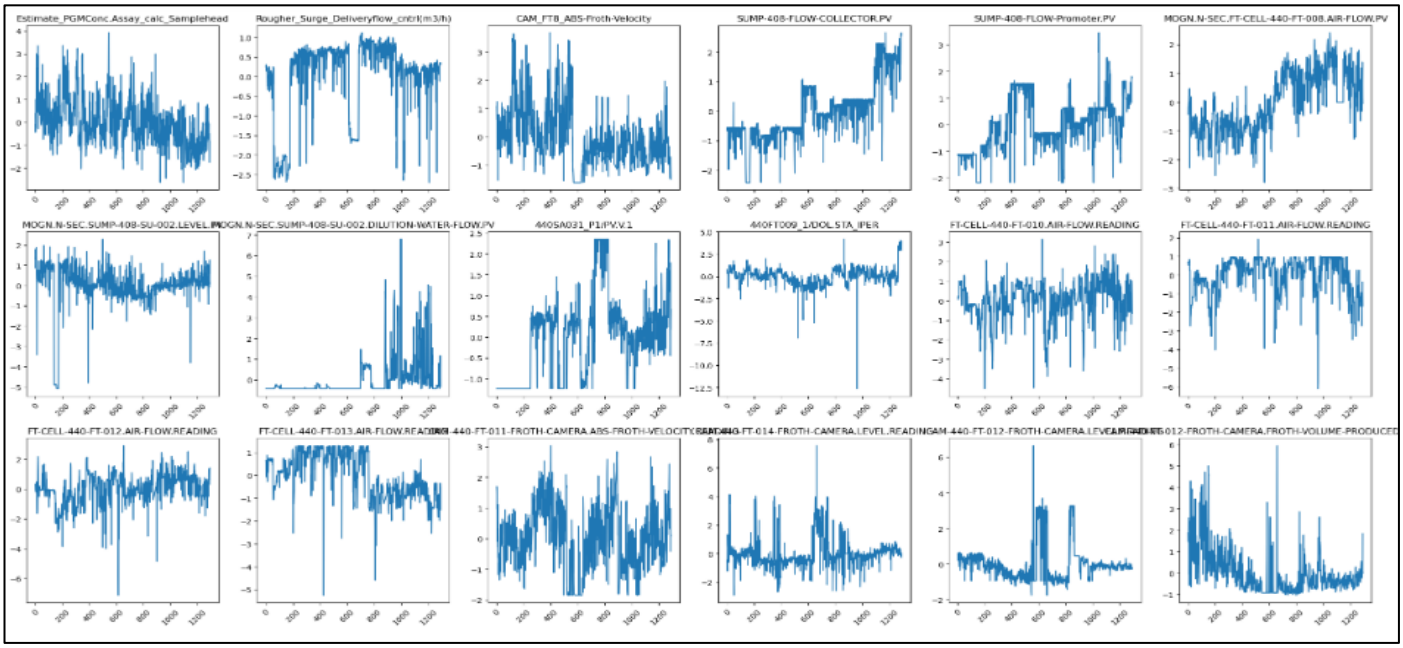


Figure 39: *Selected 18 input variables plot.*

The Ore Sample Head Assay emerged as the most critical variable, aligning with findings by Gomez-Flores et al. (2022) on the importance of physicochemical variables in flotation modelling. It also has a lower VIF value of 1.41 which indicates lower collinearity with other variables making it ideal for model training. Air Flow Rate was identified as vital, selected for five of seven cells in the secondary bank, excluding cells 9 and 14, while Absolute Froth Velocity for cells 8 and 11 highlights the relevance of operational variables, as also noted by Gupta et al. (2022). Four sump parameters namely Flow Collector, Flow Promoter, Level, and Dilution Water Flow were retained due to their stabilizing role in flotation. The selected features shown in the above table have VIF values below 3 which indicates minimal multicollinearity and high F-scores, reflecting their statistical significance. This balance ensures the features are independent and impactful, forming a strong basis for model development. Despite their recognized importance, variables like Mean Bubble Size and Superficial

Gas Velocity (Jg) were excluded due to extensive missing data, underscoring the need for robust sensing technologies.

As observed in the study by Brest et al. (2021), that suboptimal variable selection or exclusion due to data inadequacies can hinder modelling accuracy and limit operational insights. The selection of these key features balances predictive accuracy and operational reliability, enhancing the model's ability to optimize grade and recovery predictions. The time-series plots of the selected variables highlight their dynamic behaviour and importance in predicting grade and recovery in the flotation circuit. Key variables like Estimate_PGMConc.Assay_cale_Samplehead exhibit significant fluctuations, reflecting sensitivity to feed and operational changes, while flow and froth variables, such as SUMP-408-FLOW-COLLECTOR.PV and CAM_FTB_ABS_Froth_Velocity**,** show relatively stable trends with occasional deviations, indicating their critical role in process stability. Froth camera and airflow metrics display spikes linked to transient conditions, emphasizing the need for robust control and careful feature selection in predictive modelling. These patterns underscore the complex interactions in the flotation process and the necessity of advanced algorithms to capture them effectively.

The statistical analysis of the selected variables for training machine learning algorithms highlights critical aspects of the flotation circuit's operation. In figure 35 the variable Estimate_PGM_Conc.Assay_Samplehead exhibits the highest variability as reflected by its wide range and large standard deviation relative to the mean. This suggests that it plays a pivotal role in predicting grade and recovery outcomes. High variability in this parameter may indicate its sensitivity to operational changes, making it a key target for monitoring and control. Literature supports the significance of concentrate assays as they directly influence the metallurgical performance of the circuit (Jameson & Emer, 2007). Flow-related variables, such as SUMP-408-FLOW-COLLECTOR.PV and MOGN-N.SEC.SUMP-408-SU-002.DILUTION-WATER-FLOW.PV, show moderate variability, underscoring their importance in maintaining stable slurry conditions. Froth-level-related variables, including CAM-440-FT-014-FROTH-CAMERA.LEVEL.READING, exhibit lower variability but remain consistent contributors to recovery predictions.

Froth characteristics are well-documented as critical factors influencing flotation performance, as they determine the efficiency of mineral separation (Wills & Finch, 2015). These observations align with their importance in both statistical analysis and feature importance rankings. Figure 41 illustrates the statistical distribution of the training and testing datasets after sequential data splitting. Notably, there are only slight differences in the means, standard deviations, and ranges for most variables. This consistency suggests that the dataset is well-suited for reliable model training and evaluation. However, the variability in features like concentrate assays and flow rates underscores the need for preprocessing steps such as scaling to balance their influence during model training.

Additionally, feature engineering such as interaction terms between flow and froth variables may help capture non-linear relationships critical for accurate grade and recovery predictions (Hastie et al., 2009). Reducing operational variability in key parameters can further enhance flotation performance and model reliability.

	Train_count	Train_mean	Train_std	Train_min	Train_max	Test_count	Test_mean	Test_std	Test_min	Test_max
Estimate_PGMConc.Assay_calc_Samplehead	950,00	0,001	0,993	-2,630	3,910	342,00	-0,002	1,020	-2,450	3,230
Rougher_Surge_Deliveryflow_cntrl(m3/h)	950,00	-0,005	1,010	-2,710	1,090	342,00	0,013	0,980	-2,650	1,120
CAM_FT8_ABS-Froth-Velocity	950,00	0,005	0,995	-1,610	3,670	342,00	-0,014	1,020	-1,610	3,420
SUMP-408-FLOW-COLLECTOR.PV	950,00	0,001	1,010	-2,410	2,650	342,00	-0,004	0,987	-2,410	2,640
SUMP-408-FLOW-Promoter.PV	950,00	0,006	1,010	-2,170	3,440	342,00	-0,018	0,980	-2,170	3,210
MOGN.N-SEC.FT-CELL-440-FT-008.AIR-FLOW.PV	950,00	-0,018	1,020	-2,790	2,410	342,00	0,051	0,943	-2,160	2,190
MOGN.N-SEC.SUMP-408-SU-002.LEVEL.PV	950,00	0,009	0,999	-5,100	2,280	342,00	-0,025	1,010	-5,100	1,970
MOGN.N-SEC.SUMP-408-SU-002.DILUTION-WATER-FLOW.PV	950,00	0,011	1,010	-0,440	6,780	342,00	-0,030	0,977	-0,439	6,780
440SA031_P1/PV.V.1	950,00	0,011	1,000	-1,240	2,350	342,00	-0,029	0,997	-1,240	2,350
440FT009_1/DOI.STA_IPER	950,00	0,002	1,030	-12,700	4,160	342,00	-0,007	0,922	-5,230	3,900
FT-CELL-440-FT-010.AIR-FLOW.READING	950,00	-0,013	0,989	-4,520	2,380	342,00	0,035	1,030	-3,730	3,140
FT-CELL-440-FT-011.AIR-FLOW.READING	950,00	-0,009	1,010	-6,110	1,920	342,00	0,024	0,967	-3,640	1,010
FT-CELL-440-FT-012.AIR-FLOW.READING	950,00	-0,008	1,030	-7,140	2,920	342,00	0,022	0,902	-3,020	2,460
FT-CELL-440-FT-013.AIR-FLOW.READING	950,00	0,010	0,996	-5,250	1,300	342,00	-0,029	1,010	-5,250	1,290
CAM-440-FT-011-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING	950,00	0,010	1,010	-1,850	3,040	342,00	-0,028	0,978	-1,850	2,570
CAM-440-FT-014-FROTH-CAMERA.LEVEL.READING	950,00	0,018	1,030	-2,920	7,560	342,00	-0,050	0,927	-2,600	4,130
CAM-440-FT-012-FROTH-CAMERA.LEVEL.READING	950,00	-0,007	0,994	-1,750	6,660	342,00	0,021	1,020	-1,740	3,280
CAM-440-FT-012-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING	950,00	0,023	1,030	-1,040	5,940	342,00	-0,063	0,918	-1,000	4,530

Figure 40: *Statistics of the selected data for training and test set.*

5.3 Model Performance and Evaluation

5.3.1 Performance of Machine Learning Models

The evaluation of individual machine learning models is crucial in determining their effectiveness in predicting key flotation process variables. Each model's performance is assessed based on standard regression metrics as described in section 4.3.3 above. These metrics provide insights into model accuracy, error distribution, and generalisation capabilities. Linear regression serves as a baseline model to assess the linearity of the process and its ability in predicting the target variable. The advanced nonlinear models such as Random Forest, XGBoost, and Artificial Neural Networks (ANN) will confirm the existing nonlinear models' ability to learn the process dynamics. They will be evaluated for their ability to handle high-dimensional and interdependent process variables. Additionally, the impact of feature selection, hyperparameter tuning, and observer integration on model accuracy is examined. This section provides a detailed analysis of how each model performs on training and testing datasets, offering insights into their suitability for real-world industrial applications.

5.3.1.1 Machine learning performance between grade and recovery

Part of the focus of this research is to determine which process variable between the 4T grade and recovery can be more accurately predicted by machine learning algorithms. PGM grade and recovery are critical process variables that provide insight into the performance of an industrial process at any instant in time and further serve as indicators of efficiency and product quality. To make an informed decision on the most suitable target variable, the developed machine learning models were applied to predict both variables which enabled a direct comparison of their predictive accuracy.

The dataset from the Anglo-American archive had comprehensive records of both the recovery and grade. The availability of data for the two targeted process variables allowed for rigorous testing and evaluation of the algorithm's predictive performance. By comparing the results, the model with superior performance could be selected. To minimize bias, the evaluation metrics, as well as the size of the training and testing datasets, were kept consistent for both 4T recovery and grade prediction. Interestingly, random forest outperformed the rest of the algorithms in both grade and recovery prediction. The grade prediction offered improved performance as compared to recovery. The following figures are the performance plots for random forest showing its predictive capabilities towards 4T recovery. Only the Random Forest algorithm is presented, as it demonstrated the most competitive performance compared to the other models. The performance of the remaining algorithms is provided in Appendix C.

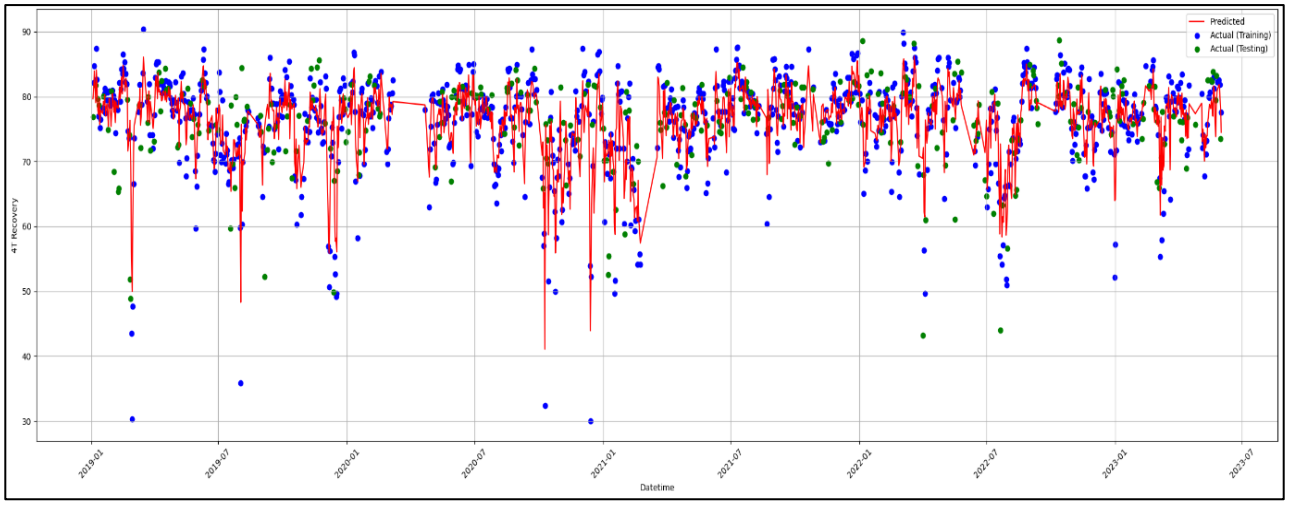


***Figure 41**: Random Forest performance plot in predicting recovery*

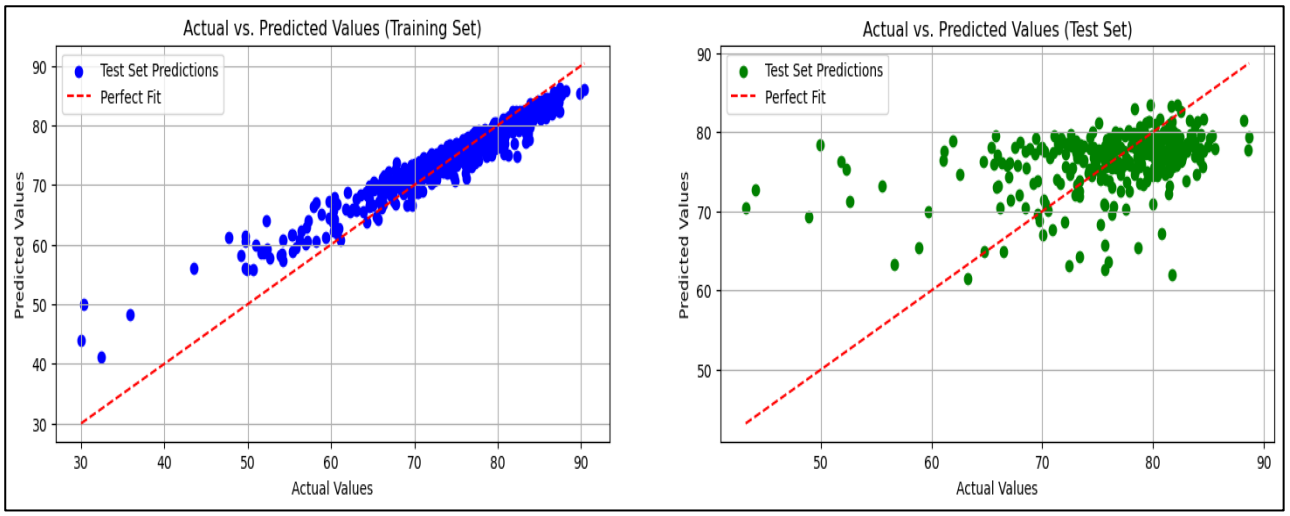


***Figure 42:** Random Forest Parity Plot obtained from 4T recovery prediction.*

Random forest was the best performing compared to linear regression, neural network, and XGBoost in predicting the 4T recovery. Figures 43 and 44 illustrate that the Random Forest model is capable of capturing some process averages for recovery prediction. Good generalisation is noted for the recovery range between 70 to 90 %, particularly within the training data. This is evident from the close clustering of points in the parity plot in figure 44 for the training set which is symbolized by blue marks as well as in the testing set marked by green. However, the model exhibits noticeable difficulty when applied to unseen data as represented by green marks. The parity plot for the testing set is more scattered compared to the training set which indicates that the model struggles to generalize its predictions beyond the data it was trained on. In both the training and testing set the models struggle to accurately predict lower recovery values below 60%. Interestingly, the opposite trend is observed in grade prediction, where the models have difficulty with higher values but perform moderately well at lower values which can be drawn back to this variable's relationship being inversely proportional to the recovery. This pattern is evident in the grade prediction parity plots presented in Appendix D.

The prediction of 4T grade demonstrates superior performance compared to the prediction of 4T recovery, as evidenced by the Random Forest results. The poor performance in recovery may be because recovery is calculated using data from various process variables, whereas grade is derived from lab analysis of composite samples and is only subjected to minimal error primarily from human error. The model's performance metrics, including lower error values and higher R-squared scores for grade prediction, indicate that the machine learning algorithms offer better generalisation for grade as compared to recovery. Based on these findings, 4T grade was selected as the primary target variable for modelling. Given the critical nature of selecting the right target variable for industrial process monitoring and optimisation the results are pivotal in ensuring the developed machine learning solution effectively addresses the operational challenges faced in dynamic industrial environments.

As part of the task to develop a soft sensor for predicting both grade and recovery, we evaluated the feasibility of using the model's predictions to calculate recovery. This calculated recovery was then compared to the original recovery representing the measured and also the calculated recoveries from plant data to determine the error margins. The figure below illustrates these error margins between the two recoveries, providing a visual representation of the discrepancies between the calculated and the plant's recovery data.

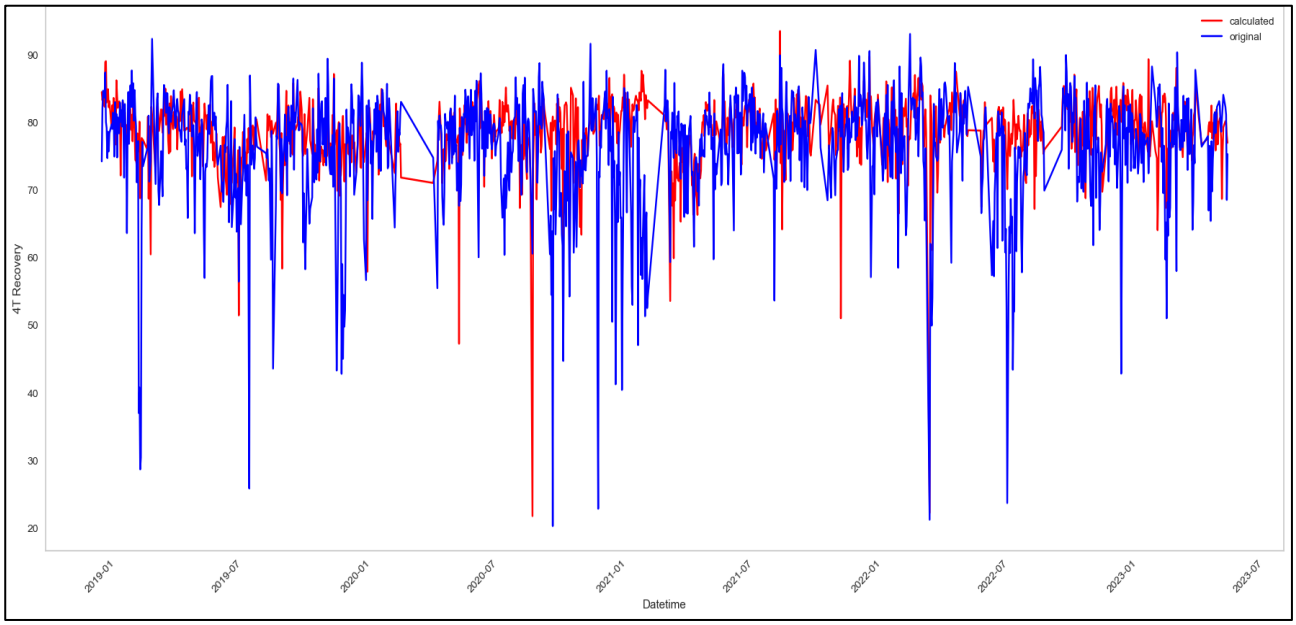


Figure 43: *Comparison between original recovery of PI Historian vs recovery calculated from the process data collected such as the sample head feed and tails grade.*

The is huge differences or errors between the two graphs and the are parts where the error is above 50% which is really concerning. As the error margin is too significant the recovery that will be calculated from the grade model's prediction will be compared to recovery to find the error margins. The error margins obtained are plotted in figure 46.

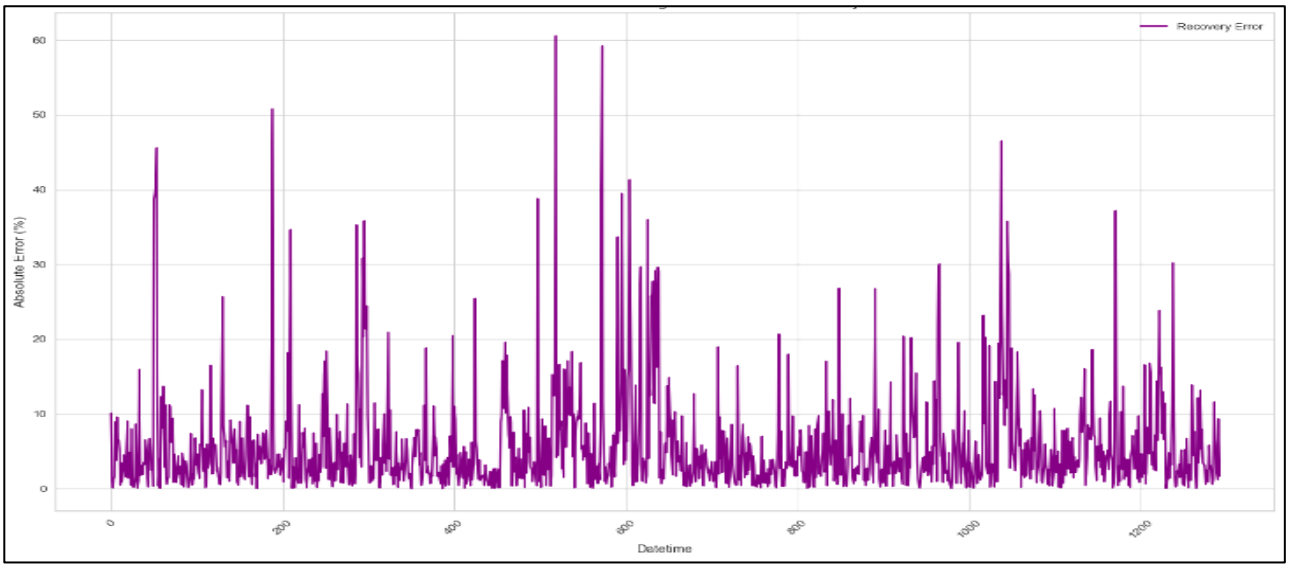


Figure 44: *Error percent between original recovery from measurements and model prediction calculated recovery.*

The significant discrepancy between the original recovery values and those calculated from the grade prediction model indicates that further refinement of the calculations is required. This inconsistency suggests that the recovery values derived from the model's predictions are not sufficiently reliable. However, improved results are observed when comparing the model-predicted recovery values to the recovery values calculated from process data (represented in red in Figure 43). In this case, the error margin falls below 1%, as illustrated in the error plot provided in Appendix D 3. This discrepancy warrants further investigation to determine if this is one of the reasons the algorithm demonstrated stronger performance in predicting grades compared to recoveries. Identifying the root cause of this divergence could clarify the limitations of the current methodology and inform adjustments to enhance the reliability of recovery predictions.

5.3.1.2 Data splitting effect on model performance

The technique used to split historical data into training and testing sets significantly impacts model performance. As outlined in the methodology, this research employs both random splitting and sequential splitting, with seven random days each month designated as the testing set. This approach helps identify the data-splitting method that enhances learning and ultimately improves model performance. The results demonstrate that different data-splitting techniques influence both linear and nonlinear models. The figures below illustrate the performance of the Linear Regression model and the Random Forest model using random data splitting.

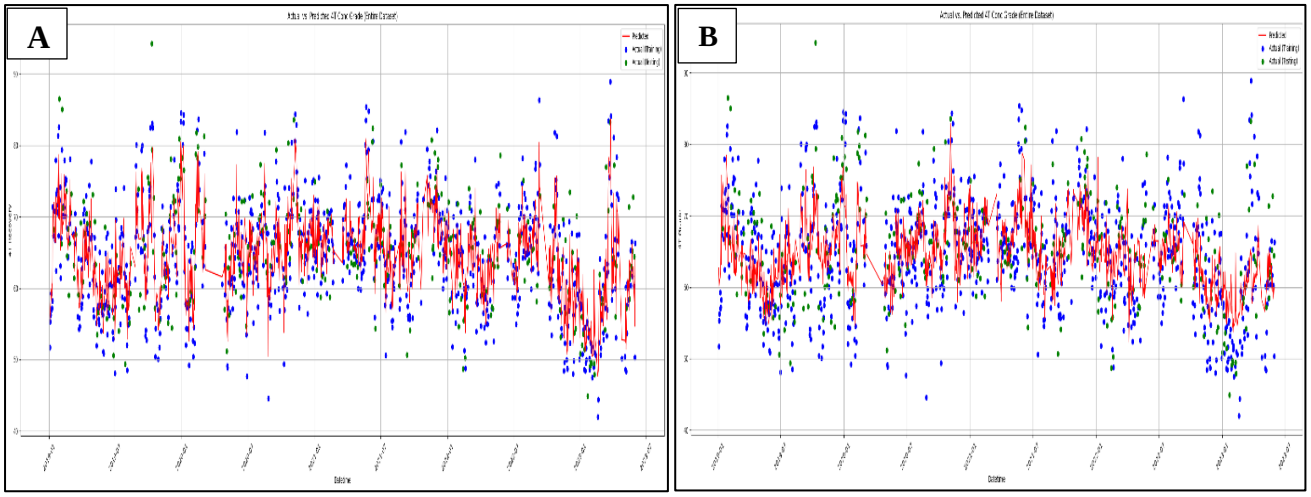


Figure 45: *The grade prediction performance plots obtained from random data splitting are presented, (A) Random Forest model performance and (B) Linear Regression model performance.*

From plots A and B above, it can be observed that the split between training (blue) and testing (green) data is evenly distributed around denser clusters but uneven in sparser regions with fewer data points. Notably, there are fewer testing values for the higher and lower extremes of the grade precisely the critical regions where the algorithm should be rigorously tested. Table 4 below compares model performances trained and tested with data that incorporated random data splitting and sequential splitting.

Table 4: *Comparison between randomly split data and sequential on the testing set.*

Evaluation Metrics	**Linear Regression**		**Random Forest**	
	Random data split	Sequential Split	Random data split	Sequential Split
MSE	43.3	53.7	36.9	35.5
RMSE	6.58	7.33	6.08	5.96
MAE	5.15	5.88	4.68	4.50
R_Squared	0.25	0.31	0.36	0.57

The results reveal that the is a significant difference in model performance with the sequential splitting having better performance. All the machine learning algorithms were tested to see their performance with the different data split. The table and figures focus solely on linear regression and random forest models. Linear regression was selected as a benchmark for linear models, while random forest was chosen due to its superior performance in nonlinear scenarios. This comparison aimed to evaluate how linear and nonlinear models behave under different data splitting strategies. Performance results for other data splitting strategies can be found in Appendix Figures E1 and E2, which show the Random Forest model's performance under sequential data splitting. This approach resulted in lower predictive accuracy compared to both random splitting and the final data splitting

method adopted in the study. The random split consisted of 26% testing and 74% training data, with the same percentages applied to the sequential split to ensure an unbiased evaluation. The model structures, selected variables, and total dataset size were kept consistent throughout. Sequential splitting consistently yields better performance metrics across models, particularly for the Random Forest, where the R-squared value improves from 0.36 under random splitting to 0.57, reflecting a substantial increase in the model's ability to explain variance when temporal order is preserved. These findings align with prior research emphasizing the critical role of temporal dependencies in industrial processes, where future outcomes are influenced by historical patterns (Zhou et al, 2020).

Random splitting disrupts the natural chronological structure of industrial data, often leading to over-optimistic results that fail to generalize to real-world scenarios (Bergmeir & Benítez, 2012). Conversely, sequential splitting maintains the temporal sequence, offering a more realistic assessment of model performance in deployment conditions, as supported by studies on time-series forecasting and process control (Hyndman, 2018). This comparison demonstrates that while random splitting may be adequate for exploratory analysis, sequential splitting is more appropriate for robust evaluation in time-sensitive industrial applications. This ensures models are better aligned with the operational realities of predictive maintenance and process optimisation tasks.

5.3.2 Hyperparameter Tuning Results

Hyperparameter tuning is essential for optimising models for better generalisation performance. Random Forest and XGBoost share almost similar hyperparameters as they are both from the decision trees family. They differ in their underlying mechanisms making tuning strategies distinct. Neural Networks, on the other hand, require a completely different set of parameters which makes direct comparisons with tree-based models challenging. The first comparison focuses on python built-in hyperparameter techniques which random search and grid search.

While built-in libraries offer automated hyperparameter tuning, manual tuning remains essential for gaining a deeper understanding of how different parameters impact model performance. This hands-on approach allows for better optimisation and interpretation of results which leads to more reliable and accurate predictions. The following figure illustrates performance improvements from the base model, followed by enhancements achieved through random search and further refinements using grid search. This comparison highlights the effectiveness of different tuning methods in improving model accuracy and overall predictive performance.

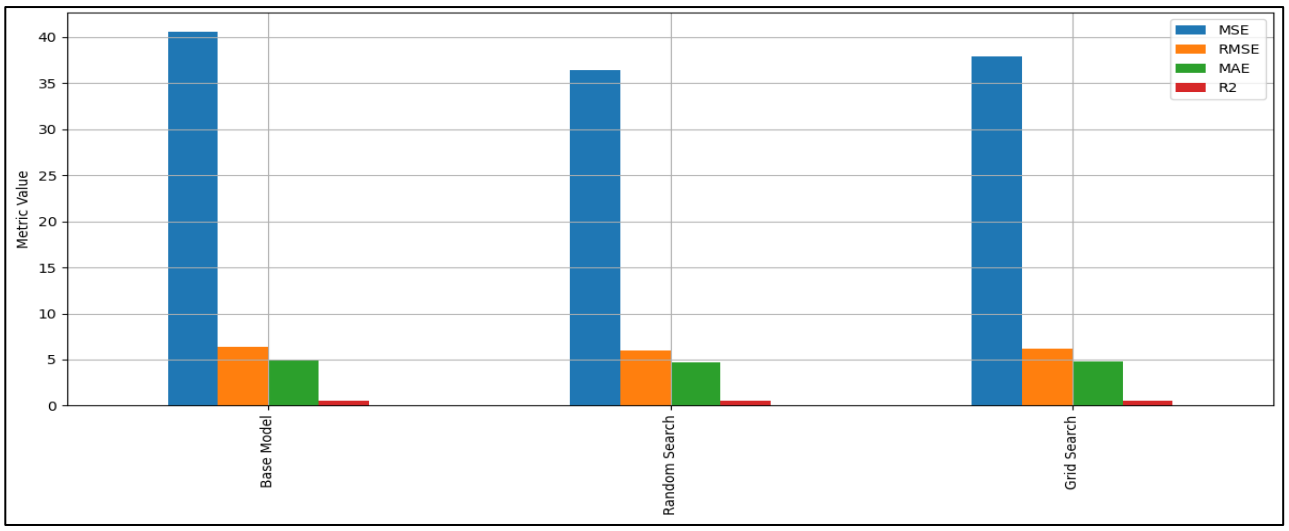


Figure 46: *Random Forest performance comparison between the base model and hyperparameter tuning.*

Hyperparameter tuning significantly improves the model's performances as can be noted in the above figure and the bar graphs in the Appendix. In the random forest plot presented above both random search and grid search reduce errors compared to the base model, leading to more accurate predictions. Grid search is also competitive in selecting the optimum parameters but a higher computational cost. The hyperparameter tuning was primarily performed manually using a trial-and-error approach to identify the optimal range of parameters for the search techniques. These models demand more tailored methods due to their complex architecture and numerous parameters, which require careful consideration and adjustment.

Grid search emerged as the more effective hyperparameter tuning strategy for both XGBoost and neural networks. It improved performance over the base models and random search as shown in the bar graphs in Appendix D1 and D2. XGBoost's gradient boosting framework relies heavily on precisely tuned parameters such as learning rate, maximum depth, and boosting rounds and benefits from exhaustive search to minimize overfitting while retaining predictive strength. Likewise, neural networks require careful adjustments to parameters like learning rate, activation functions, and layer configurations, making grid search a superior method of systematic exploration.

Overall, the analysis demonstrates that hyperparameter tuning is not a one-size-fits-all process. The choice between grid search and random search depends on the complexity of the model, the nature of the hyperparameters, and the computational constraints. While Random Forest benefits from the efficiency of random search, XGBoost and neural networks achieve optimal results through the exhaustive exploration provided by grid search. These findings align with existing literature, further validating the importance of tailoring hyperparameter optimisation techniques to the specific characteristics of each model.

5.4 Model Comparison and Evaluation Metrics

5.4.1 Model Performance Comparison

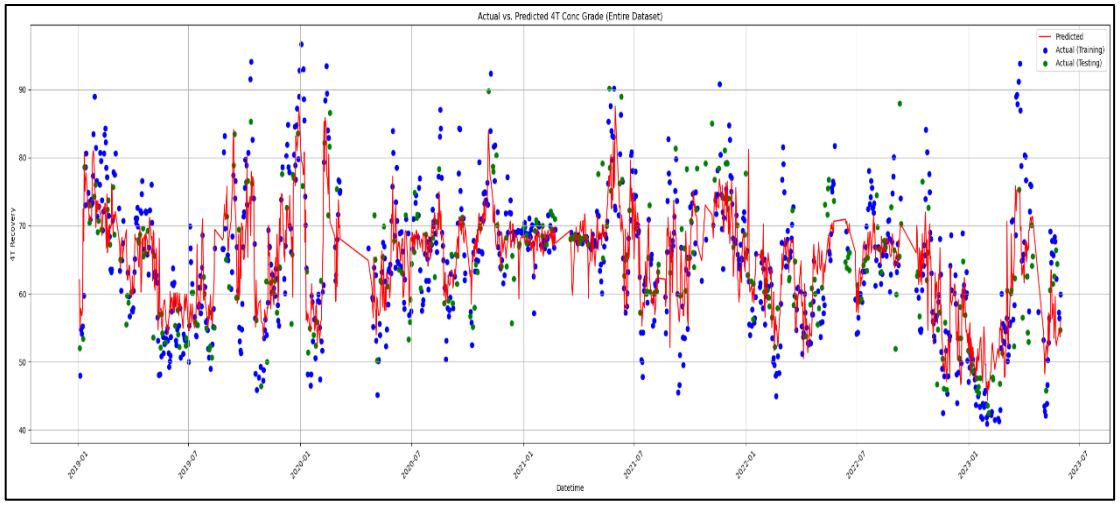


Figure 48: *Linear Regression performance plot*

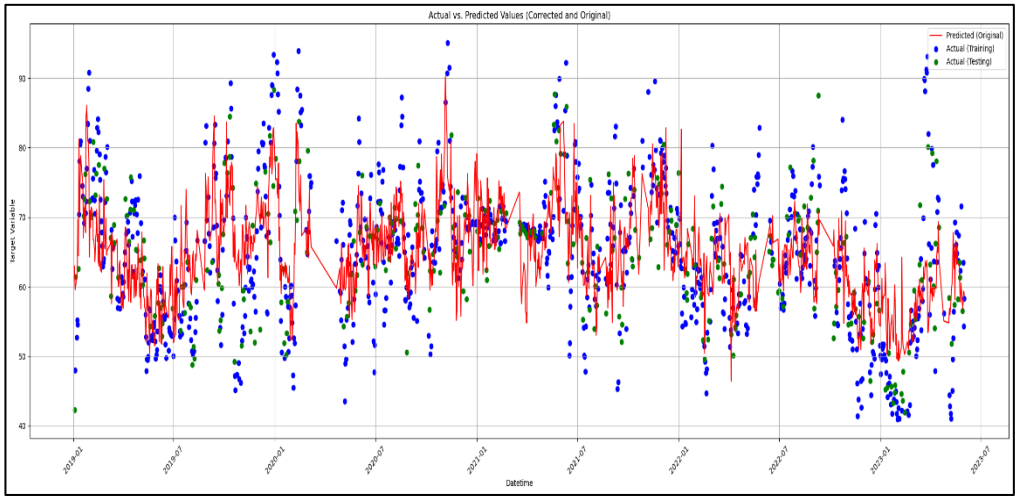


Figure 50: *Neural Network performance plot over time.*

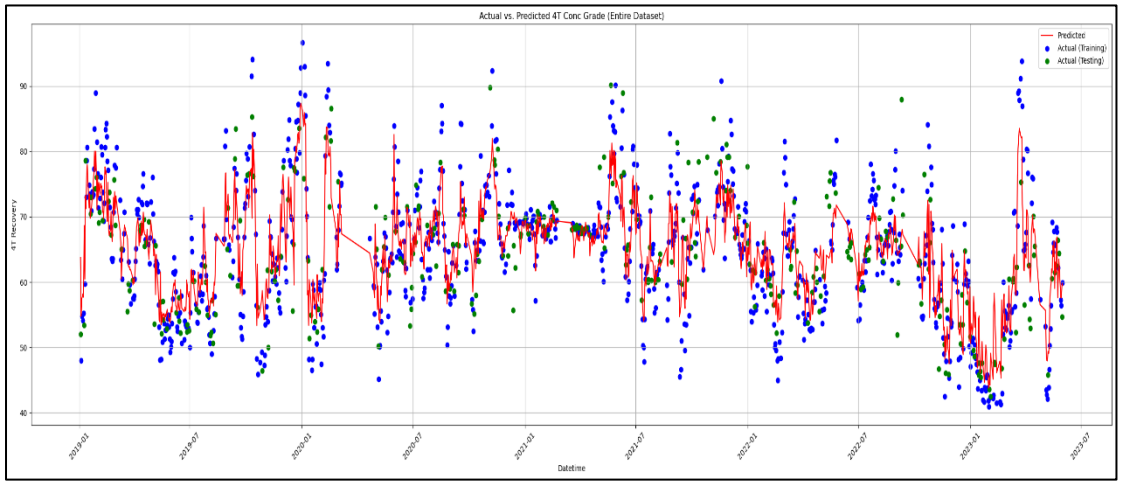


Figure 49: *Forest Regression performance plot*

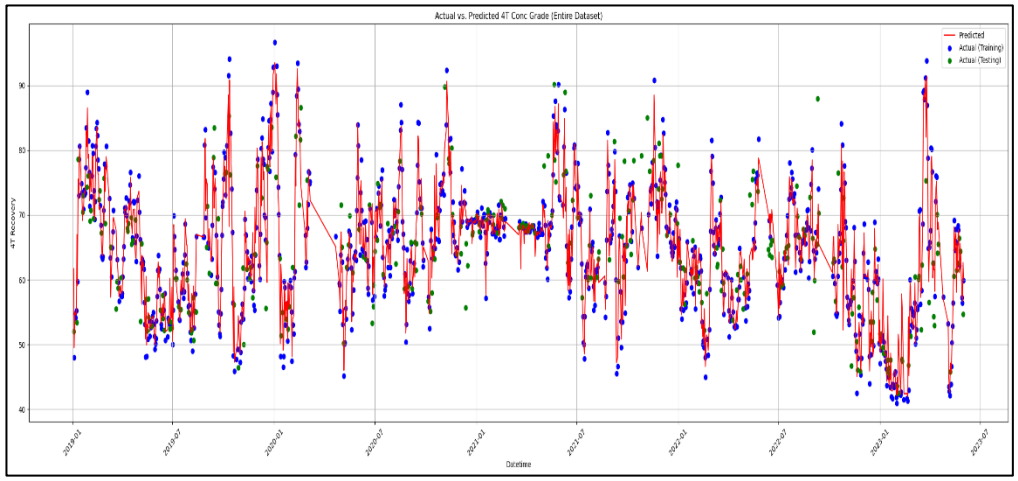


Figure 47: *XGBoost regression performance plot*

The comparison of model performance across the different models starting from linear to the nonlinear highlights their significant differences in their ability to capture patterns in the data and generalize to unseen samples. The evaluation metrics MSE, RMSE, MAE, and R-squared provide a comprehensive view of each algorithm's strengths and limitations.

Table 5: *Evaluation Metrics of the different machine learning algorithms.*

Evaluation Metrics	**ANN Regression**		**Random Forest**		**XGBoost**		**Linear Regression**	
	Training	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**
MSE	31.5	40.0	16.7	35.5	1.08	36.6	65.1	53.7
RMSE	5.61	6.32	4.09	5.96	1.04	6.05	8.07	7.33
MAE	4.33	4.80	3.11	4.50	0.75	4.47	6.25	5.88
R-squared	0.71	0.49	0.84	0.57	0.99	0.56	0.40	0.31

In the above table, Random Forest and XGBoost emerge as the best-performing models for grade prediction. Random Forest demonstrates consistent performance with relatively stable results across both the training and testing sets. In contrast, XGBoost shows excellent performance on the training set achieving a high R^2 value but only explains 56% of the variance in the unseen test data. This discrepancy suggests that XGBoost is overfitting the training data likely due to its complexity. The model may have learned to excessively rely on the training data's noise and specific patterns which do not generalize well to new data.

This is further supported by the performance plot of XGBoost in Figure 44, which shows that it is the only model capable of predicting higher-grade values an area where other models struggle to generalize effectively. On the other hand, random forest offers more reliable generalisation with an R^2 value of 87% on the training set and 57% on the test set. Its performance plot in figure 43 indicates that its predictions vary around regions where data points are more densely concentrated. This suggests that the model effectively learns from the training set while maintaining the ability to handle noise, which contributes to its generalisation performance. This is in line with its robustness in handling unseen data while avoiding overfitting. This aligns with existing literature which highlights random forest's ability to generalize effectively compared to more complex models like XGBoost (Breiman, 2001).

ANN regression demonstrates moderate performance rated 3rd in the performance comparison. Its training R2 vales of 0.71 and 0.49 in the testing set suggest that the model underfits the data, struggling to fully capture underlying patterns. While its testing MSE (40.0) and RMSE (6.32) are competitive, they lag behind those of XGBoost and Random Forest. ANNs are highly flexible but require significant data and careful tuning of architecture and hyperparameters to perform optimally, which may explain their relatively modest results in table 5 (Goodfellow et al., 2016). Linear Regression exhibits the weakest performance. Its high MSE on both the training and test set as well as the low R^2 values (0.40 on training, 0.31 on testing) indicate that it struggles to model the data effectively. This poor performance reflects the algorithm's limitation in handling non-linear relationships, which are common in real-world data (Montgomery et al., 2021).

5.5 Feature Importance and Model Interpretation

5.5.1 Interpretation of Important Features

In machine learning and metallurgical engineering, the interpretation of important features plays a pivotal role in uncovering the underlying mechanisms influencing material behaviour. Understanding which features significantly impact model predictions not only enhances the transparency of predictive models but also aligns with domain-specific insights in metallurgy, such as phase transformations, microstructure evolution, and mechanical properties. This section delves into the methodologies employed to identify and interpret key features, linking computational outputs to physical phenomena and enabling informed decision-making in alloy design, process optimisation, and quality control.

By bridging data-driven insights with metallurgical principles, this analysis fosters a deeper understanding of both the model and the material system. The different base models for the recursive feature elimination wrapper method are tested and their result compared in terms of error are presented in the following table. Table 17 shows the best performing base model are RF and gbm as they have a lower mean error compared to the other base models. Cart is the least performing base model and detects an outlier while the other models do not suggest a presents of an outlier.

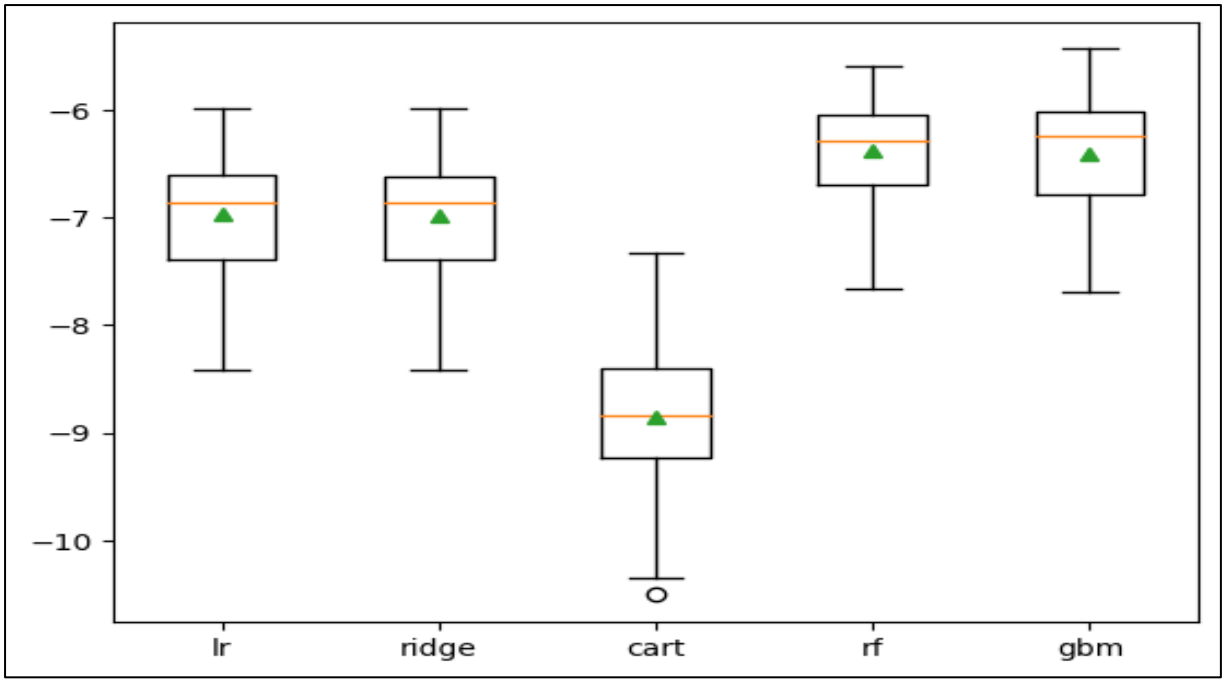


Figure 51: *Whisker box plots for the evaluation of the performance of different RFE base models.*

After the identification of the right base model to use, the question then became how many features should be selected and are a good presentation of the training set to ensure high performance. Figure 54A and B below gives us the guide on how many features to select for optimal performance. The whisker box in figure 54 A represent the error distribution from using the different number of selected variables from 10 up to 24 features. From the below figure we can see that numbers around 10 have wide range as well as interquartile range as compared to 15 and above.

The smaller the range of the whisker indicates stable performance as it would mean the algorithm is making errors around the same values. For example, the 18 variables selected had 50% of the negative mean absolute error between -4.5 to approximately -5.3. Figure 54B gives a clearer visualization as we can track the movement of the graph after the addition of a variable. In this section, the importance of adding an individual features are displayed, and the equilibrium point where any further addition of variables does not improve the performance. We can see the cross-validation score for around 18 variables has the point where the addition of a variable continues giving a constant error showing that it does not assisting in improving performance of the algorithm therefore to save the computing power the threshold of 18 variables was selected.

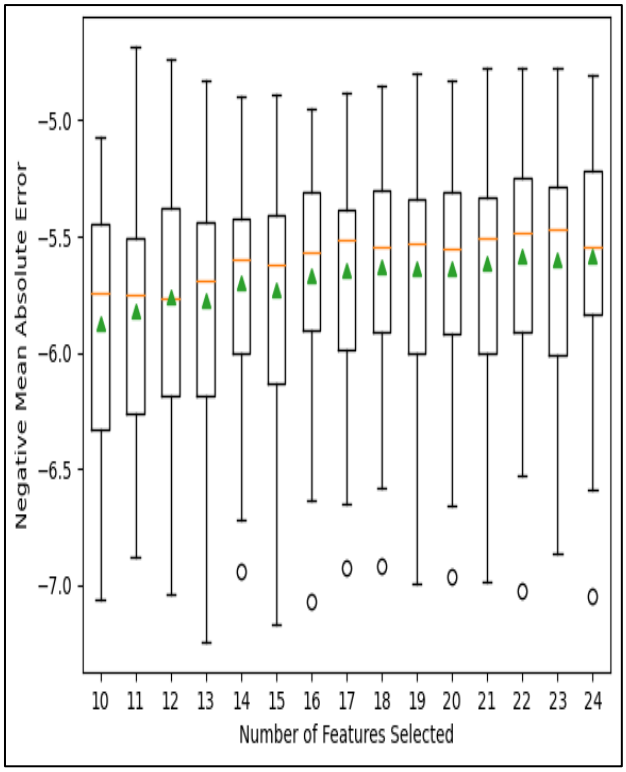


Figure 52 A: *Whisker boxplots of each selected number of variables.*

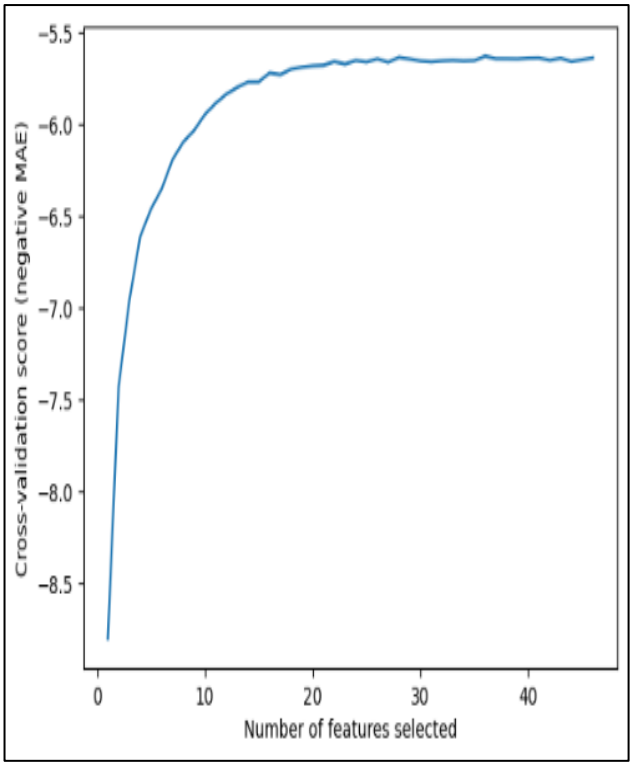


Figure 54 B: *Cross-validation Mean error against number of variables selected.*

5.5.2 Model Interpretability

It is essential to investigate how each feature influences the model's ability to generalize and make accurate predictions. This section explores the impact of individual features on the model's predictions, emphasizing the most and least influential ones. Additionally, the relationships between these features and the target variable are analysed to uncover the key drivers of model performance. To further evaluate the effectiveness of the algorithms, learning curves for both training and validation are presented, offering insights into their performance and generalisation capabilities.

This comprehensive analysis ensures that the model's predictions are both interpretable and aligned with the desired outcomes. The model will focus on interpreting the model performance in learning from the historical training data. The individual parameter influence in the generalisation of the algorithm is evaluated. The analysis will start by linear regression due to its simplicity for interpretation followed by the ensemble methods then neural network. An ideal model is one that is able to learn the valuable information from the data to make informed generalisation.

5.5.2.1Linear Regression

Linear regression exhibited the lowest performance in predicting grade and recovery compared to the non-linear models. Understanding how each feature influences model generalisation, as well as the extent of their impact is crucial for interpreting these results. Figure 54 illustrates how each input variable contributes to the model's generalisation capability.

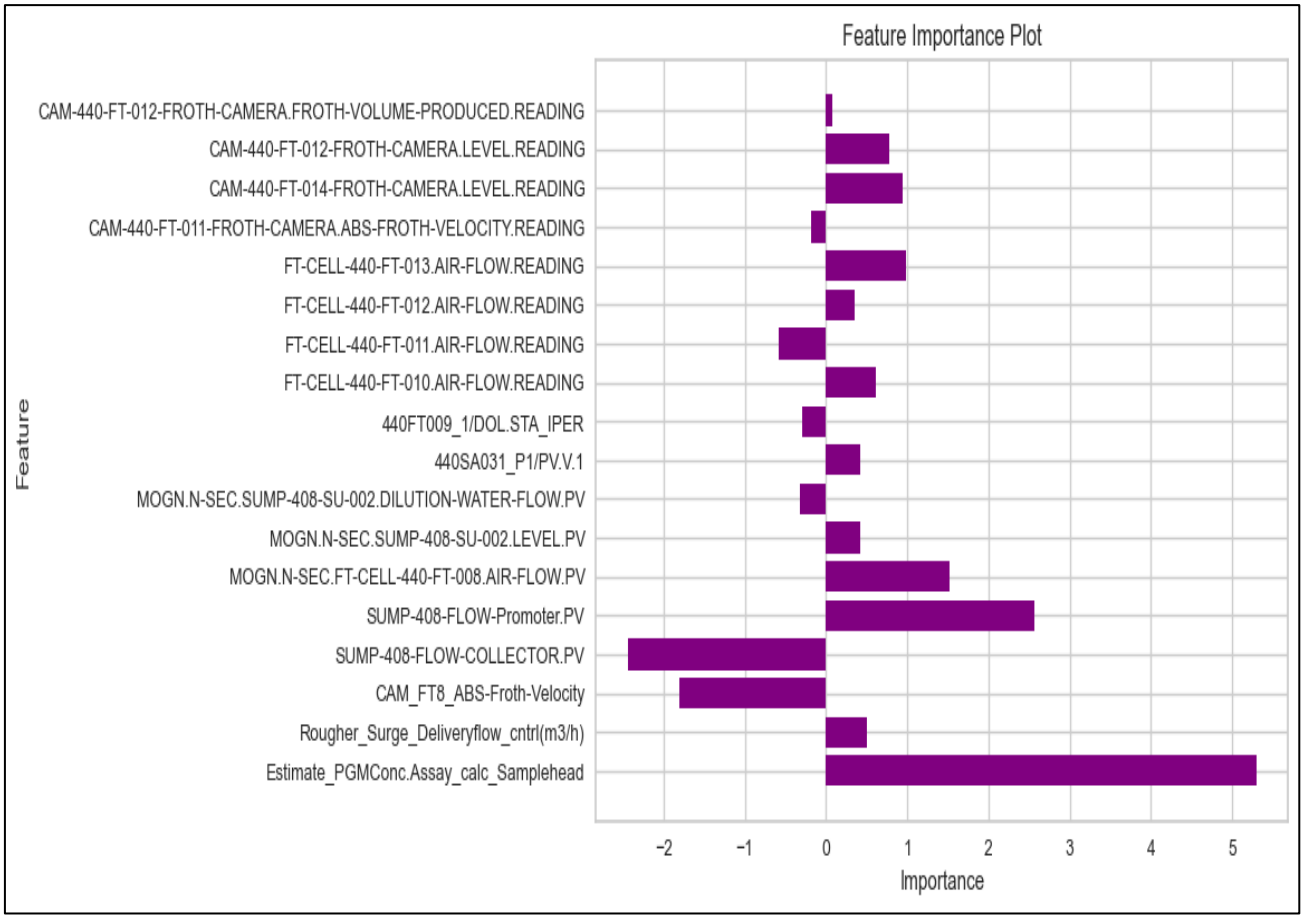


Figure 53: *Linear Regression Model feature importance*

The analysis demonstrates that feed grade, slurry flow rates, and reagent dosing are the most influential parameters for grade prediction in the linear regression model. The absolute froth velocity and air flow rate from Cell 11 were found to have negative importance, while corresponding variables from the other cells showed positive importance. This inconsistency may suggest potential issues with data capturing or sensor calibration for Cell 11 which may lead to sensor noise reading. These findings align with established flotation principles, where consistent feed quality, optimal slurry handling, and precise reagent control are vital for maximizing recovery and concentrate grade. The lower importance of froth imaging and airflow variables suggests opportunities for refinement in feature engineering or further investigation of their role in predictive modelling. By identifying high-impact variables, this analysis provides actionable insights for enhancing process control and underscores the value of integrating machine learning with metallurgical process optimisation.

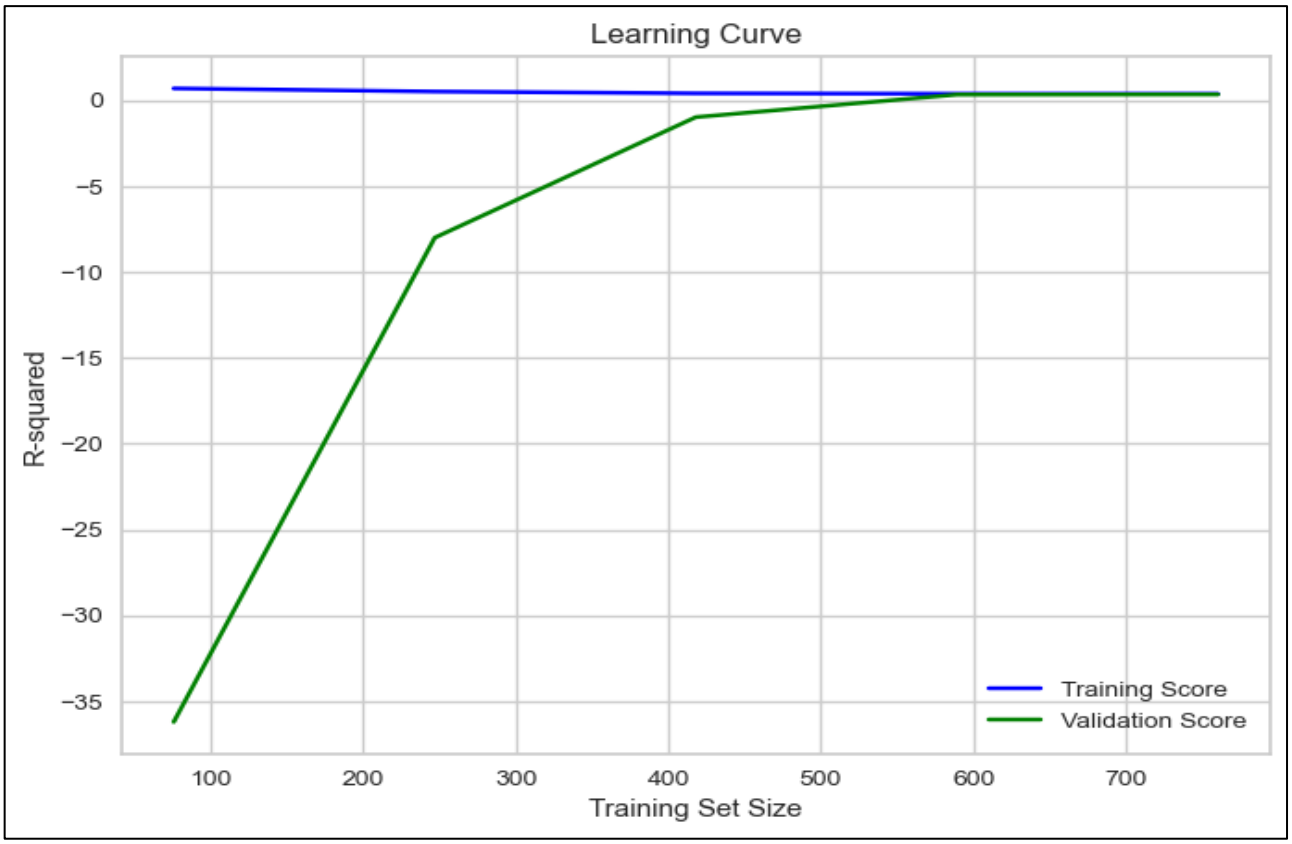


Figure 56: *Linear regression Learning curve*

The learning curve for the linear regression model shows that the training R_squared score remains slightly above zero across all training set sizes, indicating that the model achieves minimal but consistent predictive performance on the training data. In contrast, the validation R_squared score starts at a highly negative value with small training sizes and gradually improves as more data is added. The validation score starts at -35, showing poor initial performance, but gradually improves and intersects the training curve at the point (600, 0.2).

After this point, both the training and validation scores stabilize, suggesting that the model has reached a point of equilibrium with minimal further improvement in both training and validation performance. These results indicates that the model's complexity might be too low to capture the underlying trends in the data, and both training and validation scores have plateaued. Further model refinement, such as increasing complexity as adding more data might not help as the model has already converged. Try more complex models like the exponential and multivariable model as well as incorporating more data may be necessary to improve their performance. It would be interesting to see how the Leuenberger observer will improve the above plot since it has already converged.

5.5.2.2Random Forest

The feature importance plot demonstrates the random forest model's ability to assess the contribution of each input variable to its predictive performance. This is complemented by the model's learning curve, which provides further insight into how it adapts to the training data over time.

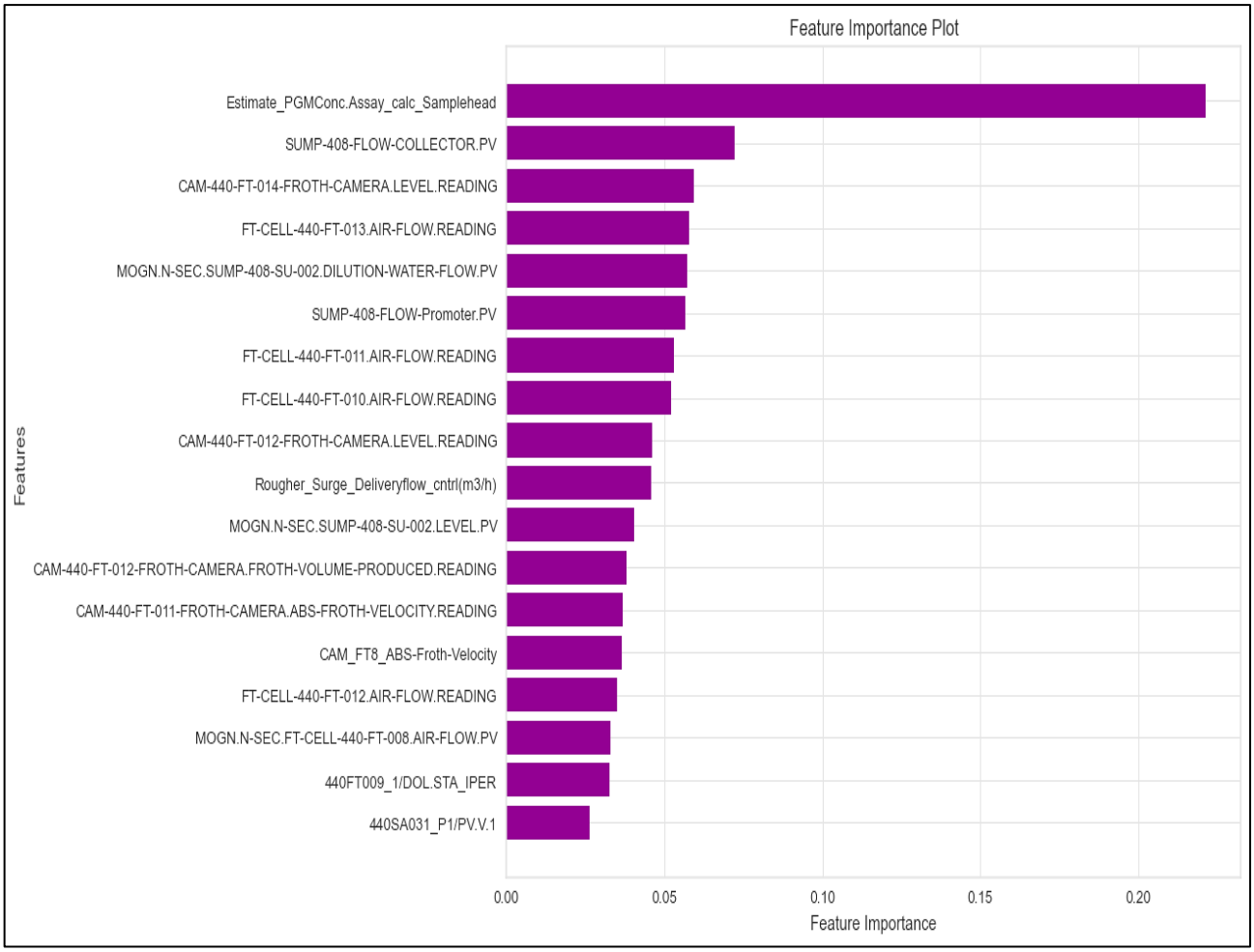


Figure 54: *Feature importance for the random forest algorithm*

The top 5 features account for 65–70% of total importance meaning the model heavily relies on a few variables which simplifies interpretability but risks overfitting if these features are noisy or collinear. The variable Estimate_PGMConc.Assay_cale_Samplehead is the most significant, with a feature importance value exceeding 0.20, demonstrating its dominant role in influencing the model's predictions. All the other features show decreasing importance all below 0.1 which is half of the PGM sample head. Random forest recognises all the 18 variables to contribute positively in the model generalisation unlike in linear regression. These results demonstrate that the random forest model effectively identifies and prioritizes key process variables, with the most influential ones relating to flow rates, froth camera measurements, and air flow readings. CAM_FT8_ABS-Froth-Velocity has low importance despite froth velocity being a known critical parameter. This may suggest data quality issues sensor noise or redundancy with other features.

Focusing on these critical features process optimisation can be guided more effectively, while features with low importance may warrant re-evaluation or potential exclusion to streamline the model further. Removal of low importance features such as 440SA031_P1/PV.V.1 and 440FT009_I/DOL.STAIPER may help to reduce complexity as model is overfitting.

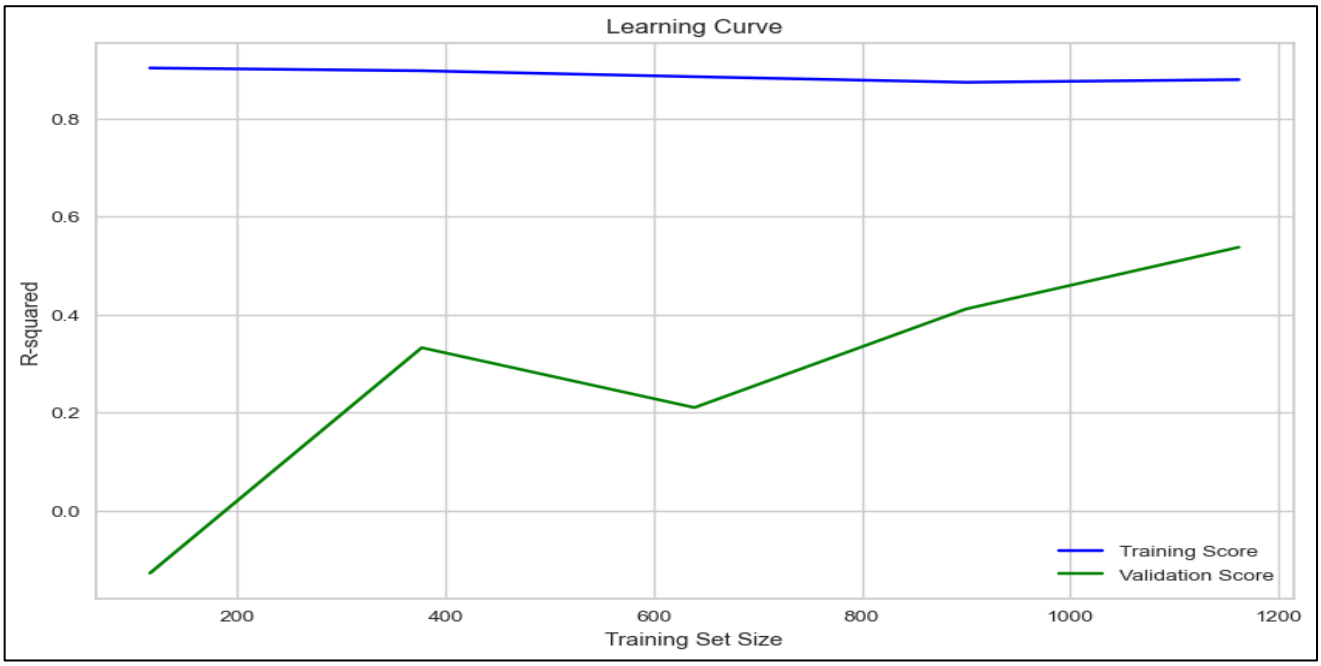


Figure 55: *Random Forest learning curve*

The learning curve for the random forest model provides insights into its predictive performance as the training set size increases. The training R^2 score remains consistently high, close to 0.9, across all training set sizes, reflecting the model's ability to fit the training data well due to its ensemble nature and capacity to capture complex, non-linear relationships (Breiman, 2001). In contrast, the validation R^2 score starts near zero with small training sizes but improves steadily reaching approximately 0.57 as the training set size approaches its maximum. This steady improvement underscores the random forest model's sensitivity to data availability which aligns with prior studies that emphasised the need for sufficient training data to fully leverage its predictive potential (Hastie et al., 2009).

The gap between the training and validation R^2 scores indicates some degree of overfitting, which is typical for random forest models when the model complexity exceeds the signal-to-noise ratio in the data. However, the validation curve's upward trajectory suggests that the model has not reached its saturation point in learning and may benefit from further increases in training data. Studies in machine learning literature highlight that ensemble methods, including random forests, tend to achieve optimal performance with large datasets, as they rely on averaging predictions across multiple decision trees to reduce variance and improve generalisation (Dietterich, 2000).

To address the observed gap and further improve the validation performance techniques such as hyperparameter tuning which include increasing the number of trees, adjusting maximum tree depth, or modifying the minimum samples per split can be explored. Additionally, ensuring that the training data is representative of the broader population and enhancing feature engineering efforts can provide further benefits. This analysis reaffirms the importance of learning curves as a diagnostic tool for assessing model performance and guiding decisions in model improvement and dataset augmentation.

5.5.2.3 XGBoost

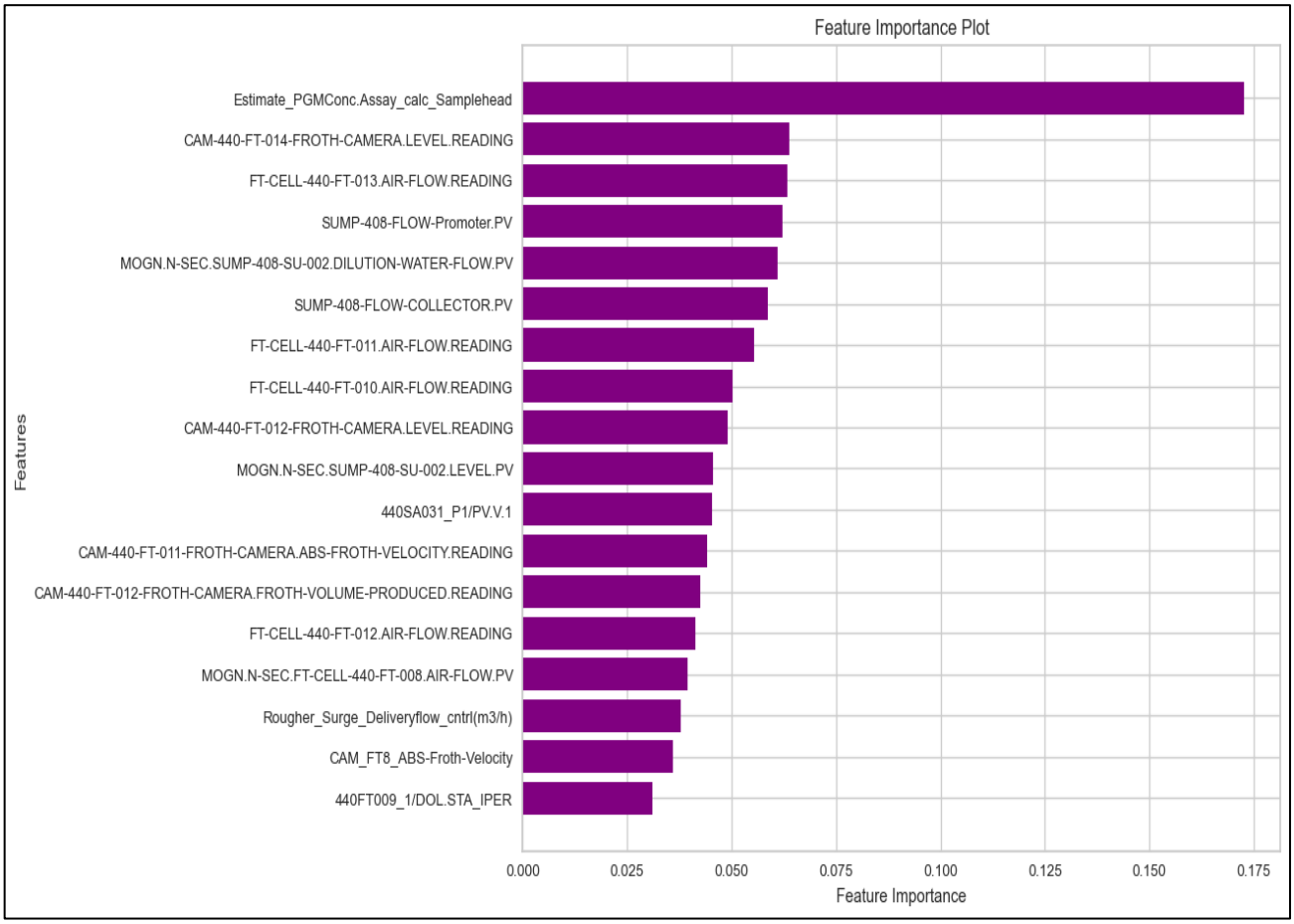


Figure 56: *Feature importance for XGBoost during training and validation.*

Feature importance analysis across all algorithms reveals that the estimated sample-head consistently emerges as the most influential variable in gold prediction. This aligns with existing studies that emphasize the strong correlation between feed characteristics and flotation performance (Gupta & Yan, 2006). However, variations in secondary important variables highlight differences in how each algorithm interprets the underlying process dynamics. For instance, while Random Forest identifies the sump flow collector as the second most critical variable, XGBoost prioritizes the froth camera level reading from cell 14 but both algorithms recognizes the froth level from cell 4 and the air flowarate from cell 13 of high importance in the model generalisation.

This discrepancy suggests that XGBoost, with its gradient-boosting mechanism, may be more sensitive to real-time flotation surface characteristics, whereas Random Forest relies more on process flow dynamics. Despite these variations, a key similarity across all models is the recognition of sump-related process variables as crucial factors in grade prediction. The fact that every algorithm ranks sump variables within the top 10 features reinforces their importance in flotation circuit optimisation.

Another interesting observation is the treatment of rougher surge delivery flow. XGBoost and Linear Regression rank it among the least important features, suggesting limited influence on overall process outcomes. However, Random Forest assigns it moderate importance, ranking it as the 10th most relevant variable. This divergence may be due to the inherent differences in model architecture Random Forest, being an ensemble of decision trees, captures a broader range of feature interactions, while XGBoost tends to favour variables with more direct impact on the target output (Chen & Guestrin, 2016). These findings have significant implications for process optimisation. Understanding which variables drive flotation efficiency can help metallurgists prioritize sensor calibration, data collection strategies, and real-time process adjustments. The importance of sump-related variables suggests that enhanced control of sump flow dynamics could lead to more stable and predictable flotation outcomes. Additionally, the role of froth camera readings in XGBoost highlights the potential value of integrating vision-based monitoring systems with machine learning models for improved real-time decision-making.

Figure 57: *Learning curve for the XGBoost*

The learning curve reveals the performance of the XGBoost model across varying training set sizes. The training score represented by the blue line starts high at approximately 1 and drops to less than 0.8 reflecting that the model is overfitting the data. Conversely, the validation score depicted by the green line steadily increases from very low values up to around 0.53 as the training set size grows. This positive trend suggests that the model's generalises better with an increase in the training set. We see the unseen data curve that starts low then goes to 0.35 then starts decreasing. It decreased with the addition of approximately 200 datapoints then started increasing again showing improvement with additional training examples. Notably, the absence of a plateau in the validation curve indicates that the model has not yet reached its optimal training size and could benefit from further data augmentation. Developing new features or transforming existing ones can improve the model's predictive accuracy by better capturing key patterns in the data. Applying L1 (Lasso) or L2 (Ridge) regularization could potentially address overfitting as it penalises overly complex models and their weights.

5.5.2.4 Neural Network

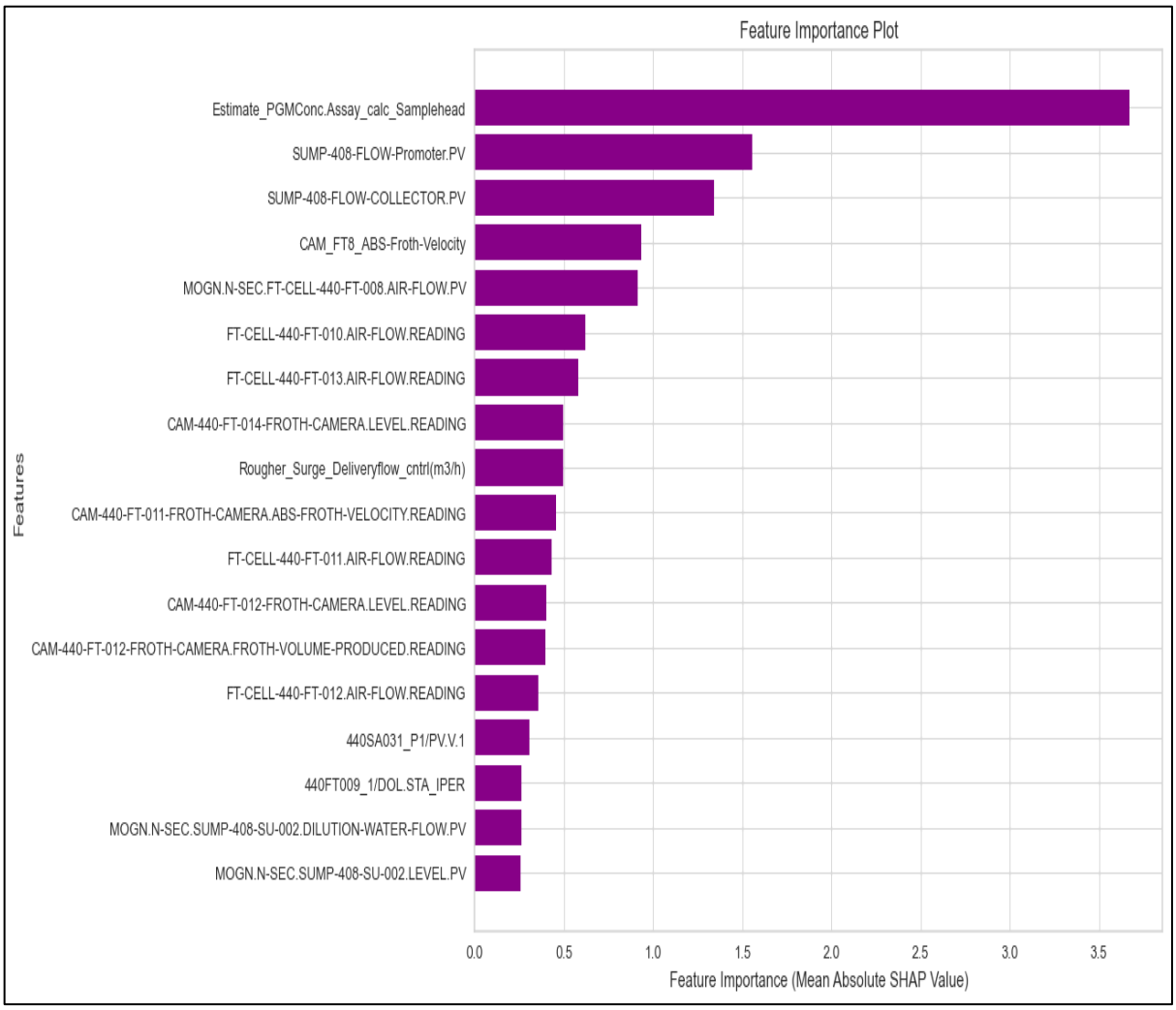


Figure 58: *Feature importance rating neural network model.*

Neural networks recognize all 18 selected variables as important for the model's generalisation, aligning with other nonlinear models such as XGBoost and Random Forest. Across all models, the PGM sample head emerges as the most influential input variable, reinforcing its critical role in predicting flotation performance. However, variable importance rankings vary slightly among models. For instance, in neural networks the top important variables explain about 80% of the model's generalisation and the rest accounting for the remaining 20%. The sump level is the least important variable in the neural network while it ranks 10th in XGBoost and 6th in random forest, indicating differences in how each model interprets process state.

The neural network highlights the sump collector and promoter as high-importance variables which is consistence with Hlotse et al (2023) findings that reagent dosages play a critical role in maximizing PGM recovery and the effectiveness of the separation process. A significant disparity exists between the most and least influential variables, with the highest-ranked inputs being directly or indirectly linked to cells 8 and 9 in the flotation circuit. This correlation is expected, as these cells play a crucial role in PGM recovery. The findings highlight the necessity of focusing on key process variables to enhance model accuracy and optimize control strategies in flotation operations. The learning curve presented in the figure 62 illustrates the relationship between training set size and model performance measured by R^2.

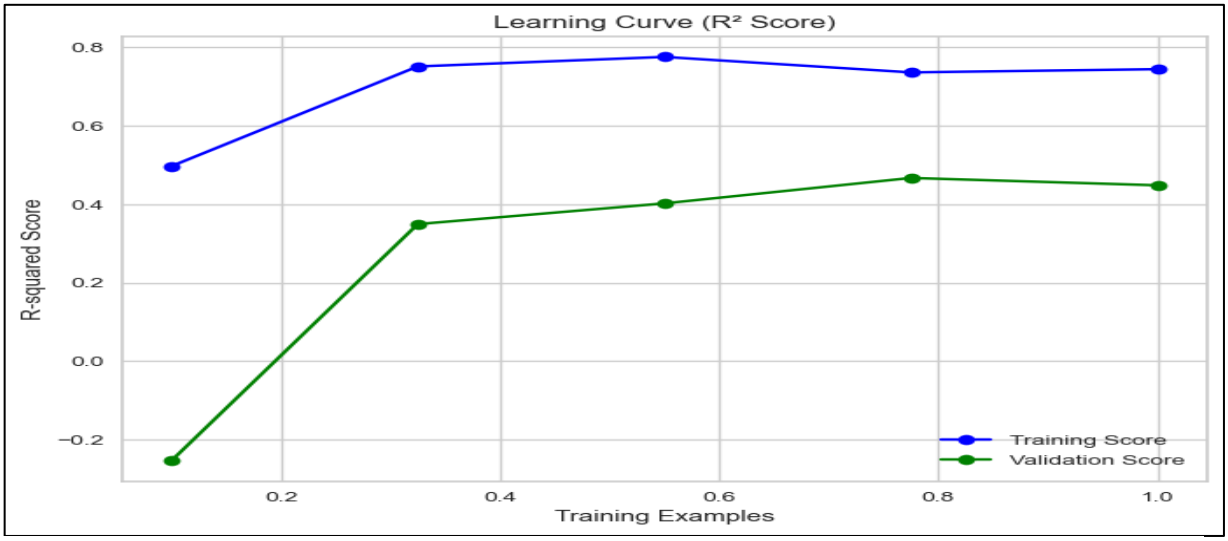


Figure 59: *Nueral Network learning curve*

The training error represented by the blue line, starts low when the dataset is small but gradually increases as more data is introduced. This indicates that the model initially overfits to the limited dataset but starts generalizing better with additional training samples. Eventually, the training error stabilizes suggesting that the model has reached a point where additional data does not significantly reduce its learning error.

On the other hand, the validation error represented by the green line starts low and increases as the training set size increases. This trend suggests that the model struggled with generalisation when trained on a small dataset but improves as it is exposed to more data. However, after about 60% the validation error plateaus and of then increases to a point where it stays nearly constant after, highlighting that adding more training data will not assist to better model generalisation.

A notable observation from the graph is the persistent gap between training and validation errors which indicates that a certain level of overfitting. The model performs better on the training data than on unseen validation data, which suggests that additional optimisation strategies, such as hyperparameter tuning, feature selection, or regularization, may be necessary to enhance its performance. Despite this, the overall decreasing trend in validation error confirms that increasing training data improves generalisation, making the model more reliable for practical applications.

5.5.2 Luenberger Observer Integration

The integration of the Luenberger Observer into machine learning models provides a robust framework for incorporating dynamic system state estimations. By leveraging the observer's ability to estimate unmeasured states in a control system, we enhance the predictive capabilities of data-driven models, especially in complex, nonlinear environments. In this section, we present the results of our machine learning models integrated with the Luenberger Observer. This integration aimed to achieve higher prediction accuracy, improved generalisation, and better interpretability of the model's behaviour.

Table 6: *Luenberger Observation Algorithms performance evaluation.*

Evaluation Metrics	ANN Regression		Random Forest		XGBoost		Linear Regression	
	Training	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**
MSE	34.4	35.1	11.7	38.3	3.08	38.0	50.9	39.4
RMSE	5.86	5.93	3.41	6.19	1.76	6.16	7.12	6.27
MAE	4.56	4.51	2.64	4.62	1.33	4.57	5.59	4.85
R-squared	0.67	0.53	0.89	0.54	0.97	0.54	0.51	0.52

Table 7: *Linear Regression before and after the integration of Leuenberger Observer*

	Evaluation Metrics Linear Regression			
	Training Set		**Testing Set**	
	Corrected	**Original**	**Corrected**	**Original**
MSE	50.9	65.2	39.4	44.7
RMSE	7.12	8.08	6.27	6.68
MAE:	5.59	6.29	4.85	5.25
R-squared	0.51	0.37	0.52	0.46

The Luenberger observer significantly improves the performance of linear models like Linear Regression and Support Vector Machines (SVM), while nonlinear models such as XGBoost and Random Forest show little reliance on it. This suggests that nonlinear models naturally capture complex system behaviours, whereas linear models benefit from state estimation to enhance their predictive accuracy. Table 7 highlights the impact on Linear Regression, where the observer reduces testing error by 10% and improves R-squared from 46% to 52% on the test set and from 37% to 51% on the training set. This improvement makes Linear Regression more competitive with nonlinear models, showing that observer-based corrections can refine simpler models for better performance. In industrial applications, this finding is valuable. While complex machine learning models often require high computational power and may be harder to interpret, integrating observer-based corrections offers a practical way to enhance traditional models. This approach can improve real-time process monitoring and optimisation, especially in environments with noisy or incomplete data.

Table 6 shows that using the Luenberger observer improves the training set performance of all models when compared to the results in Table 5 where the observer was not used. However, no significant improvement is observed for the non-linear models. However, in the testing set, results vary while linear models and Support Vector Machines (SVM) benefit from the observer, tree-based algorithms like Random Forest and XGBoost show a slight decline in performance. This suggests that tree-based models already capture complex relationships effectively, making external state estimation less useful. SVM was included to further test the observer's impact on linear models, and it exhibited a similar improvement as Linear Regression. This reinforces the idea that observer-based corrections enhance models relying on linear assumptions by refining their ability to capture process variations. Additional plots illustrating the model's performance and its integration within the machine learning framework are provided in Appendix Figure D_2. Neural networks, on the other hand, show inconsistent results across different runs, making it difficult to draw firm conclusions. Their performance appears to be highly sensitive to the data split method when key high-value data points are missed, the observer helps reduce error, but in other cases, it may introduce additional error.

This suggests that for deep learning models, data partitioning strategies play a crucial role in determining whether state estimation techniques like the Leuenberger observer provide real benefits. The plot in Figure 62 compares the original algorithm's predictions before incorporating the Leuenberger observer with the adjusted predictions after its inclusion, represented by the grade plot. The original plot is represented by the red plot and the blue dot are the actual process values.

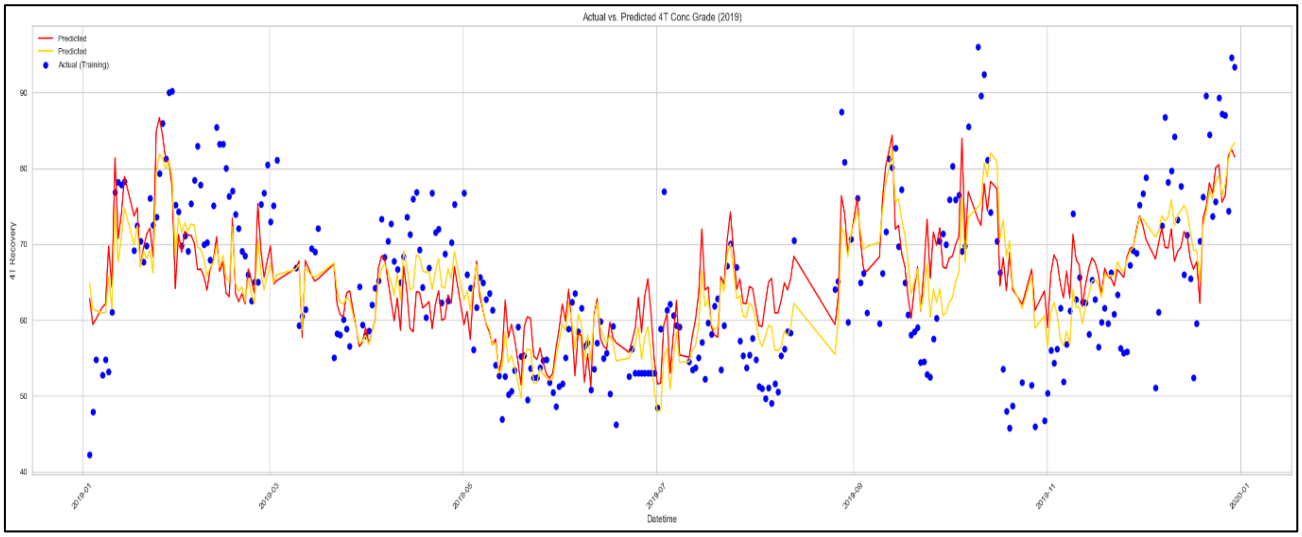


Figure 60: *Linear Regression performance with and without the Leuenberger observer.*

From March to May, the original model consistently underestimated the actual values. The observer corrected this by reducing the error by at least half, bringing the predictions closer to reality. A similar improvement is observed in September 2019, where the original model overestimated the values. The observer adjusted the predictions downward, reducing the error and enhancing overall model accuracy. These adjustments demonstrate the observer's effectiveness in refining predictions and improving model performance. The opposite effect is observed in nonlinear models, where the Leuenberger observer tends to shift predictions in the wrong direction, increasing the error instead of reducing it. This pattern is evident in the Random Forest, XGBoost, and Neural Network models, as shown in Appendix Figures 4A to 4D. This outcome supports the assumption that nonlinear models inherently capture complex process dynamics without needing observer-based corrections. The observer adjusts predictions based on past errors, but this approach can be problematic during state transitions. When the system shifts from one state to another, the observer assumes the previous error pattern still applies, leading to incorrect adjustments and reduced accuracy in nonlinear models. Sequential data splitting is expected to enhance the performance of the Leuenberger observer integration, as the data sequence reflects the actual process dynamics over time. This temporal alignment facilitates more effective learning and corrective measures, compared to a random data split which disrupts the natural order of changes. This observation highlights an opportunity for further research to explore the impact of data splitting strategies on observer-based model performance.

Chapter 6: Discussion

6.1 Interpretation of Key Findings

The study highlights a critical challenge in industrial machine learning applications which includes the inconsistency and abnormality of historical data. A significant portion of data is lost during cleaning and pre-processing, underscoring the need for more advanced data acquisition technologies that can provide reliable data at shorter sampling intervals. Improving data collection methods will ensure a richer dataset for training and testing machine learning models which will ultimately lead to more accurate and generalizable predictions.

A key insight from the study is that randomly splitting industrial data into training and testing sets is often impractical. Given that flotation processes are time-dependent and continuous, a structured data-splitting approach yields more efficient results compared to random splitting. This approach also provides deeper insight into the data used for training and testing the algorithm. By visualizing the data alongside predictions, it becomes easier to identify specific data points or regions where the model struggles to generalize, allowing for targeted improvements in model performance. Time-series-aware splitting methods, such as rolling windows or expanding windows, could improve model robustness by maintaining the temporal dependencies within the data.

Feature selection also plays a crucial role in model development. By selecting the most relevant features, the regression models can be optimized for both computational efficiency and predictive accuracy. Feature importance analysis suggests that better predictions are achieved when modelling grade rather than recovery. This could be attributed to the nature of the historical data, where grade-related variables exhibit clearer patterns, while recovery metrics may be influenced by additional unmeasured factors. The findings confirm that flotation circuit dynamics are inherently nonlinear, as evidenced by the superior performance of nonlinear algorithms like Random Forest and XGBoost over linear models.

Random Forest and XGBoost demonstrate consistently higher performance across both training and testing datasets reaffirming their effectiveness in handling complex, high-dimensional, and nonlinear industrial data. Additionally, the study suggests that increasing the amount of training data enhances the performance of nonlinear models, highlighting the importance of data availability in industrial machine learning applications. The integration of a Leuenberger observer into the modelling process appears to positively influence model performance. However, its impact varies across different nonlinear models is likely due to differences in data structuring and how each algorithm generalizes.

Nearly all nonlinear models successfully interpret at least 50% of unseen data, reinforcing their suitability for flotation process modelling. These findings suggest that further research into data structuring techniques and hybrid modelling approaches could yield even greater improvements in predictive accuracy and process optimisation.

6.2 Comparison with Existing Studies

The findings of this study align with existing literature on the application of machine learning in industrial processes, particularly in flotation circuit modelling. Similar to previous studies, the results confirm that nonlinear models such as XGBoost and Random Forest outperform linear regression models in predicting flotation outcomes. It was found that process variables from nearly all the second rougher cells contain valuable information that is essential for the learning and generalization of the machine learning models. Research in metallurgical process modelling has consistently demonstrated the limitations of linear regression due to the inherent nonlinearities in flotation circuits (Jorjani et al., 2018). The superior performance of nonlinear algorithms in this study reinforces the necessity of adopting machine learning approaches capable of capturing complex interactions within process variables. Unlike some studies that emphasize deep learning approaches such as Convolutional Neural Networks (CNNs) for image-based flotation analysis (Zhang et al., 2020), this research focuses solely on regression-based techniques using numerical process data. While CNNs have shown promise in extracting valuable insights from flotation images, integrating both historical process data and image-based features remains an underexplored avenue. This study highlights the potential benefits of such integration, suggesting that future research could combine flotation images with numerical data to enhance model generalisation and predictive power.

Furthermore, while previous studies often rely on randomly splitting datasets for model training and evaluation, this study emphasizes the importance of structured data splitting in time-dependent processes. The results confirm that time-aware partitioning improves predictive performance by maintaining temporal dependencies within the data. This insight addresses a key limitation in existing studies, where random splits may lead to misleading evaluations of model performance due to the sequential nature of industrial processes. Another notable distinction is the study's emphasis on the impact of data availability and pre-processing on model performance. Compared to prior research that assumes clean and well-structured datasets, this study underscores the challenges posed by inconsistencies and data loss during preprocessing. The findings suggest that improving data acquisition methods such as increasing sampling frequency and integrating real-time monitoring systems—could significantly enhance model accuracy and robustness.

6.3 Implications for Industry Practice

The study's findings have significant implications for industrial practice, particularly in the optimisation of flotation circuits through machine learning. The results demonstrate that nonlinear regression models, particularly XGBoost and Random Forest, provide real time prediction of grade and recovery with acceptable accuracy solving the limitation of traditional linear approaches. This insight suggests that industries employing flotation processes can achieve more reliable process control and optimisation by adopting advanced machine learning techniques to estimate grade in real time, rather than relying solely on delayed laboratory analyses of composite samples. However, the effectiveness of such models is constrained by the sampling frequency which limits the amount of data available to train the models. In this study, only daily average samples were available; more robust and responsive models could potentially be developed with higher-frequency data, such as hourly measurements.

One critical takeaway for industrial application is the importance of high-quality, high-frequency data collection. The study highlights how a substantial amount of industrial data is lost or deemed unusable due to inconsistencies, leading to reduced training efficiency. For example, the mean bubble size has been found as a important variable in the flotation cell but was removed from the data set due to high amount of missing data. To address this issue, industries should invest in robust data acquisition systems, such as automated sensors and real-time monitoring technologies, to ensure a more complete and accurate dataset for model training. Improved data collection would not only enhance predictive accuracy but also reduce model uncertainty, leading to better decision-making in process optimisation.

Additionally, the study challenges the conventional approach of randomly splitting industrial datasets for model validation. Since flotation processes are continuous and time-dependent, structured data partitioning methods such as rolling windows or sequential splits offer more reliable performance assessments. This insight can assist practitioners in implementing machine learning models that more accurately reflect real-world grade values. By preserving the temporal structure of the data, sequential splitting enables better prediction of future grades based on historical trends, thereby enhancing the reliability and effectiveness of model deployment in industrial settings. The study also suggests that flotation grade predictions are generally more accurate than recovery predictions due to the nature of the available data.

6.4 Limitations of the Study and future research recommendations

This study contributes to the ongoing research in the intersection of machine learning and industry, specifically focusing on regression-based algorithms for predictive modelling. While regression models like XGBoost and Random Forest have demonstrated strong performance, the study does not incorporate deep learning techniques such as Convolutional Neural Networks, which have gained significant traction in the literature for their ability to extract valuable insights from flotation images. CNNs have been particularly effective in image-based applications related to mineral processing, where visual features can complement traditional numerical data to enhance predictive accuracy (LeCun et al., 2015).

A potential avenue for future work involves integrating both historical process data and flotation images into the model training process. Combining structured numerical data with unstructured image data could improve generalisation and provide a more comprehensive understanding of the flotation process. Studies have shown that hybrid models leveraging both numerical and visual information can yield superior results compared to purely numerical models (Krizhevsky et al., 2012).

Additionally, while this study explores several nonlinear regression models, further investigation into alternative non-linear algorithms, such as K-Nearest Neighbors (KNN) or Gaussian Process Regression (GPR), could offer valuable insights. These models have been recognized for their ability to capture complex relationships in data and may provide a better comparative framework for flotation process optimisation (Smola & Schölkopf, 2004). Another limitation of this study is the exclusive use of Python as the software environment for model selection.

While Python is widely regarded for its extensive machine learning libraries (e.g., Scikit-learn, TensorFlow, and XGBoost), other platforms such as MATLAB and R offer distinct advantages for certain algorithms and data structures. Different computational environments may optimize model performance differently, and exploring multiple software frameworks could provide a more holistic perspective on algorithm selection. Future research could benefit from incorporating deep learning techniques like CNNs, integrating flotation images with numerical data, exploring additional nonlinear models, and evaluating models across multiple software environments to ensure robustness in algorithm selection.

Chapter 7: Conclusion

7.1 Summary of Findings

This study explored the performance of various machine learning regression algorithms in predicting the concentrate grade on a rougher flotation bank using industrial process data. The results highlight key trends in model performance, feature importance, and the impact of the Leuenberger observer on prediction accuracy. Among the evaluated models, nonlinear algorithms such as Random Forest and XGBoost demonstrated superior predictive capabilities, outperforming traditional linear models like Linear Regression. These tree-based models effectively captured the nonlinear dynamics of the flotation process, making them well-suited for complex metallurgical systems. However, Linear Regression and Support Vector Machines (SVM) benefited significantly from the integration of the Leuenberger observer, which improved their predictive accuracy by correcting systematic errors. The observer reduced testing errors by approximately 10% in linear regression, increasing its competitiveness with nonlinear models.

Feature importance analysis revealed that sump-related process variables consistently ranked among the most influential factors in predicting PGM grade. This highlights their critical role in regulating key physical parameters such as mass pull, residence time, and pulp density that directly impact flotation efficiency. As the sump serves as a dynamic buffer and flow control point within the circuit, its operating conditions significantly influence the stability and performance of the overall flotation process. Additionally, the study confirmed that historical industrial data contains significant inconsistencies, with substantial data loss occurring during preprocessing. This highlights the need for robust data acquisition and storage systems which will ensure that enough and reliable data is available for process improvement. Structured data-splitting techniques improved model performance particularly given the continuous and time-dependent nature of flotation processes.

The observer's impact varied across model types. While it significantly improved linear models by adjusting for systematic errors, it had a negative effect on nonlinear models like XGBoost and Random Forest. In these cases, the observer often moved predictions in the wrong direction which suggest that nonlinear models already capture the complex process variations effectively without additional state estimation. The observer's reliance on past errors appears to introduce inaccuracies during state transitions, reinforcing the importance of model selection based on process characteristics.

Overall, the findings emphasize the feasibility of the data-driven soft sensor in inferring the PGMs grade prediction in real time. Future research should explore integrating deep learning with machine learning to enhance grade prediction accuracy. By incorporating flotation camera images, the model

can recover valuable information lost during preprocessing, allowing for a more comprehensive understanding of the process. Additionally, leveraging real-time process data alongside image analysis could improve predictive performance and make the model more applicable for industrial use. Refining data collection strategies will further enhance model reliability and ensure more accurate, data-driven decision-making in flotation optimisation.

7.2 Contributions to Knowledge

This study makes significant contributions to both machine learning and metallurgical engineering, particularly in the optimisation of flotation processes through data-driven modelling. It demonstrates that the selected process variables are more effective in predicting PGMs grade rather than recovery, highlighting key factors that influence flotation performance. Additionally, nonlinear machine learning algorithms have demonstrated superior performance in estimating grade in real time, compared to traditional methods that rely on periodic composite samples and laboratory analysis. This makes them particularly well-suited for predictive control in industrial applications. Furthermore, decision tree-based approaches like Random Forest and XGBoost, exhibit superior performance compared to neural networks and linear regression as well as support vector machines.

It emphasizes the critical role of feature selection in enhancing model performance. In identifying and retaining only the most relevant process variables, the models achieved better predictive accuracy and computational efficiency. Sump-related process variables were consistently ranked among the most critical factors influencing PGMs grade. Their importance alongside traditionally studied flotation cell variables is noted even though much of the literature focuses on flotation cell parameters as primary predictors. The study reveals that sump conditions play a vital role in process performance. This insight emphasizes the need to consider sump variables in predictive modelling to improve overall flotation efficiency. Understanding these key variables allows metallurgists to refine control strategies and improve real-time process adjustments, contributing to more stable and efficient flotation operations. The research advances the application of state estimation techniques in machine learning for metallurgical processes. The Leuenberger observer was found to significantly enhance linear models in reducing error and improving predictive accuracy. However, its negative impact on nonlinear models underscores the importance of aligning observer-based corrections with model architecture.

This finding challenges the assumption that state estimation universally improves machine learning performance and suggests that such techniques should be selectively applied based on the underlying process dynamics. It also underscores the limitations of the existing industrial historical data, emphasizing the need for better data acquisition and structured data-splitting techniques. The

observation that random data splitting may not always be appropriate for continuous processes contributes to the growing discussion on best practices for handling time-dependent process data in machine learning applications. Finally, by demonstrating the varying effects of the Leuenberger observer across different models, this research provides a foundation for future studies to explore hybrid modelling approaches, combining physics-based process knowledge with data-driven techniques. This could lead to the development of more robust predictive models that integrate both process understanding and machine learning capabilities, ultimately enhancing industrial decision-making in metallurgical engineering.

7.3 Practical Implications

This study confirms that the flotation circuit operates as a highly nonlinear process, as only nonlinear models successfully learn and capture its dynamic behaviour. This finding underscores the potential of advanced machine learning techniques, such as XGBoost and Random Forest, for optimising flotation processes. By leveraging these models, process control strategies can be improved, leading to enhanced prediction accuracy and better operational efficiency. These models offer improved PGMs grade and recovery real time monitoring system which will lead to more efficient process operations compared to traditional linear models. Extending the area of focus enhances the model's generalization capabilities, as information from both pre- and post-flotation stages provides valuable insight into the overall state of the process. Models identifying sump-related process variables and airflows as critical predictors suggests that focusing on these key parameters can enhance process control and decision-making in ensuring optimum process performance.

The research also emphasizes the importance of structured data-splitting techniques in continuous processes like the flotation cell as time-aware partitioning leads to more reliable predictions than random splits. The Leuenberger observer mixed impact on model performance suggests that state estimation should be applied selectively rather than uniformly across all models. One way to improve its effectiveness is by limiting its use to predefined sections of the process where changes occur in a more linear manner. For instance, instead of applying it to daily averages as done in this study, it could be used over shorter time frames where the process follows a linear trend. This targeted approach would enhance its ability to support model accuracy without negatively affecting nonlinear models. Finally, the study underscores the need for better data acquisition systems and the integration of real-time sensor and image-based data, which could significantly improve the robustness of machine learning models for industrial applications.

8.References

A Ilemobayo, J. *et al.* (2024a) 'Hyperparameter Tuning in Machine Learning: A Comprehensive Review', *Journal of Engineering Research and Reports*, 26(6), pp. 388–395. Available at: https://doi.org/10.9734/JERR/2024/V26I61188.

A Ilemobayo, J. *et al.* (2024b) 'Hyperparameter Tuning in Machine Learning: A Comprehensive Review', *Journal of Engineering Research and Reports*, 26(6), pp. 388–395. Available at: https://doi.org/10.9734/JERR/2024/V26I61188.

Ai, E./ and Dltf, / (2024) 'European IT Certification Curriculum Self-Learning Preparatory Materials Deep Learning with TensorFlow'. Available at: https://eitca.org/certification/eitc-ai-dltf-deep-learning-with-tensorflow/ (Accessed: 9 March 2025).

Aldrich, C. *et al.* (2010) 'Online monitoring and control of froth flotation systems with machine vision: A review', *International Journal of Mineral Processing*. Available at: https://doi.org/10.1016/j.minpro.2010.04.005.

Ali, S.H. *et al.* (2017) 'Mineral supply for sustainable development requires resource governance', *Nature*, 543(7645), pp. 367–372. Available at: https://doi.org/10.1038/NATURE21359.

Andridge, R.R. and Little, R.J.A. (2010) 'A Review of Hot Deck Imputation for Survey Non-response', *International Statistical Review*, 78(1), pp. 40–64. Available at: https://doi.org/10.1111/J.1751-5823.2010.00103.X.

Awad, M. and Khanna, R. (2015) 'Support Vector Regression', *Efficient Learning Machines*, pp. 67–80. Available at: https://doi.org/10.1007/978-1-4302-5990-9_4.

Beaglehole, D., Mondelli, M. and Belkin, M. (2024) 'Average gradient outer product as a mechanism for deep neural collapse'.

Di Bella, A. *et al.* (2007) 'A comparative analysis of the influence of methods for outliers detection on the performance of data driven models', *Conference Record - IEEE Instrumentation and Measurement Technology Conference* [Preprint]. Available at: https://doi.org/10.1109/IMTC.2007.379222.

Bergmeir, C. and Benítez, J.M. (2012) 'On the use of cross-validation for time series predictor evaluation', *Information Sciences*, 191, pp. 192–213. Available at: https://doi.org/10.1016/J.INS.2011.12.028.

Bergstra, J. *et al.* (2011) 'Algorithms for Hyper-Parameter Optimization', *Advances in Neural Information Processing Systems*, 24.

Bishop, C. and Nasrabadi, N. (2006) *Pattern recognition and machine learning*. Available at: https://link.springer.com/book/9780387310732 (Accessed: 29 October 2024).

Breiman, L. (2001a) 'Random forests', *Machine Learning*, 45(1), pp. 5–32. Available at: https://doi.org/10.1023/A:1010933404324/METRICS.

Breiman, L. (2001b) 'Random forests', *Machine Learning*, 45(1), pp. 5–32. Available at: https://doi.org/10.1023/A:1010933404324/METRICS.

Breiman, L. (2001c) 'Random forests Machine Learning, 45 (1) (2001), pp. 5-32', *Machine Learning*, 45(1).

Brest, K.K. *et al.* (2021) 'Statistical investigation of flotation parameters for copper recovery from sulfide flotation tailings', *Results in Engineering*, 9, p. 100207. Available at: https://doi.org/10.1016/J.RINENG.2021.100207.

Brownlee, J. (2016) 'Machine Learning Mastery With Python Understand Your Data, Create Accurate Models and Work Projects End-To-End'.

Cai, Y.-W. *et al.* (2024) 'The application of "transfer learning" in optical microscopy: the petrographic classification of metallic minerals', *American Mineralogist* [Preprint]. Available at: https://doi.org/10.2138/AM-2023-9092.

Cateni, S. and Colla, V. (2018) 'A Hybrid Variable Selection Approach for NN-Based Classification in Industrial Context', *Smart Innovation, Systems and Technologies*, 69, pp. 173–180. Available at: https://doi.org/10.1007/978-3-319-56904-8_17.

Cattaldo, M., Ferrer, A. and Måge, I. (2024) 'Variable time delay estimation in continuous industrial processes', *Chemometrics and Intelligent Laboratory Systems*, 246, p. 105082. Available at: https://doi.org/10.1016/J.CHEMOLAB.2024.105082.

Chen, T. and Guestrin, C. (2016) 'XGBoost: A scalable tree boosting system', *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-August-2016, pp. 785–794. Available at: https://doi.org/10.1145/2939672.2939785/SUPPL_FILE/KDD2016_CHEN_BOOSTING_SYSTEM_01-ACM.MP4.

Corrigan, J. and Zhang, J. (2021) 'Developing accurate data-driven soft-sensors through integrating dynamic kernel slow feature analysis with neural networks', *Journal of Process Control*, 106, pp. 208–220. Available at: https://doi.org/10.1016/J.JPROCONT.2021.09.006.

Cortes, C., Vapnik, V. and Saitta, L. (1995) 'Support-vector networks', *Machine Learning 1995 20:3*, 20(3), pp. 273–297. Available at: https://doi.org/10.1007/BF00994018.

Cover, T.M. and Hart, P.E. (1967) 'Nearest Neighbor Pattern Classification', *IEEE Transactions on Information Theory*, 13(1), pp. 21–27. Available at: https://doi.org/10.1109/TIT.1967.1053964.

Crundwell, F. *et al.* (2011) *Extractive metallurgy of nickel, cobalt and platinum group metals*. Available at: https://books.google.com/books?hl=en&lr=&id=JIpJZQPrvuUC&oi=fnd&pg=PP1&dq=%09+Extractive+Metallurgy+of+Nickel,+Cobalt+and+Platinum+Group+Metals&ots=hGGYkSV4TW&sig=lSe-lmwK4azmsxDDQv29ecNxlpU (Accessed: 14 June 2025).

D1Etterich, T. (1995) 'Overfitting and undercomputing in machine learning', *ACM Comput. Surv.*, 27(3), pp. 326–327. Available at: https://doi.org/10.1145/212094.212114.

Davies, L. and Gather, U. (1993) 'The identification of multiple outliers', *Journal of the American Statistical Association*, 88(423), pp. 782–792. Available at: https://doi.org/10.1080/01621459.1993.10476339.

Dayo-Olupona, O. *et al.* (2023) 'Adoptable approaches to predictive maintenance in mining industry: An overview', *Resources Policy*, 86. Available at: https://doi.org/10.1016/J.RESOURPOL.2023.104291.

Ding, S. (2009) 'Feature selection based F-score and ACO algorithm in support vector machine', *2009 2nd International Symposium on Knowledge Acquisition and Modeling, KAM 2009*, 1, pp. 19–23. Available at: https://doi.org/10.1109/KAM.2009.137.

Eduardo, Z. *et al.* (1995) 'Heuristic and Model Predictive Control Strategies for a Simulated Flotation Circuit', *IFAC Proceedings Volumes*, 28(17), pp. 75–81. Available at: https://doi.org/10.1016/S1474-6670(17)46747-4.

Fabra-Boluda, R. *et al.* (2024) 'Cracking black-box models: Revealing hidden machine learning techniques behind their predictions', *Intelligent Data Analysis*, Preprint(Preprint), pp. 1–21. Available at: https://doi.org/10.3233/IDA-230707.

Fisher, O.J. *et al.* (2022) 'Data-driven modelling for resource recovery: Data volume, variability, and visualisation for an industrial bioprocess', *Biochemical Engineering Journal*, 185, p. 108499. Available at: https://doi.org/10.1016/J.BEJ.2022.108499.

Fortuna, L. *et al.* (2007) *Soft sensors for monitoring and control of industrial processes*. London, UK: Springer.

François-Lavet, V. *et al.* (2018) 'An Introduction to Deep Reinforcement Learning', *Foundations and Trends® in Machine Learning*, 11(3–4), pp. 219–354. Available at: https://doi.org/10.1561/2200000071.

François-Lavet, V. *et al.* (no date) 'An introduction to deep reinforcement learning', *nowpublishers.comV François-Lavet, P Henderson, R Islam, MG Bellemare, J PineauFoundations and Trends® in Machine Learning, 2018•nowpublishers.com* [Preprint]. Available at: https://doi.org/10.1561/2200000071.

Fu, Y. and Aldrich, C. (2020) 'Deep Learning in Mining and Mineral Processing Operations: A Review', *IFAC-PapersOnLine*, 53(2), pp. 11920–11925. Available at: https://doi.org/10.1016/J.IFACOL.2020.12.712.

Gaudin, A.M., Mular, A.L. and O'connor3, R.F. (1957) 'Separation of Microorganisms by Flotation IL. Flotation of Spores of Bacillus subtilis var. niger1'. Available at: https://journals.asm.org/journal/am (Accessed: 1 November 2024).

Gomez-Flores, A. *et al.* (2022) 'Prediction of grade and recovery in flotation from physicochemical and operational aspects using machine learning models', *Minerals Engineering*, 183, pp. 107627–107627. Available at: https://doi.org/10.1016/J.MINENG.2022.107627.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep learning An MIT Press Book*, *Nature*.

Grote-Ramm, W. *et al.* (2023) 'Continual learning for neural regression networks to cope with concept drift in industrial processes using convex optimisation', *Engineering Applications of Artificial Intelligence*, 120, p. 105927. Available at: https://doi.org/10.1016/J.ENGAPPAI.2023.105927.

Guo, W. *et al.* (2020) 'A review on data-driven approaches for industrial process modelling', *International Journal of Modelling, Identification and Control*, 34(2), pp. 75–89. Available at: https://doi.org/10.1504/IJMIC.2020.110352.

Gupta, A. and Yan, D.S. (2006) 'Process Control', *Mineral Processing Design and Operation*, pp. 622–671. Available at: https://doi.org/10.1016/B978-044451636-7/50019-X.

Gupta, M., Huang, K. and Yoon, R.H. (2022) 'Predicting the recovery and grade of a rougher flotation circuit from liberation data', *Minerals Engineering*, 188, pp. 107853–107853. Available at: https://doi.org/10.1016/J.MINENG.2022.107853.

Guyon, I. and Andre, E. (2003) 'An Introduction to Variable and Feature Selection André Elisseeff', *Journal of Machine Learning Research*, 3, pp. 1157–1182.

Guyon, I. and De, A.M. (2003) 'An Introduction to Variable and Feature Selection André Elisseeff', *Journal of Machine Learning Research*, 3, pp. 1157–1182.

Guyon, I. and Elisseefl, A. (2006) 'An Introduction to Feature Extraction', *Studies in Fuzziness and Soft Computing*, 207, pp. 1–25. Available at: https://doi.org/10.1007/978-3-540-35488-8_1.

Hashem, I.A.T. *et al.* (2015) 'The rise of "big data" on cloud computing: Review and open research issues', *Information Systems*, 47, pp. 98–115. Available at: https://doi.org/10.1016/J.IS.2014.07.006.

Hastie, T., Friedman, J. and Tibshirani, R. (2001) 'The Elements of Statistical Learning'. Available at: https://doi.org/10.1007/978-0-387-21606-5.

Hastie, T., Tibshirani, R. and Friedman, J. (2009) 'The Elements of Statistical Learning'. Available at: https://doi.org/10.1007/978-0-387-84858-7.

Haykin, S. (1999) 'Neural networks: a comprehensive foundation by Simon Haykin', *The Knowledge Engineering Review*, 13(4), pp. 409–412.

Herrera, N. *et al.* (2023) 'Soft Computing Application in Mining, Mineral Processing and Metallurgy with an Approach to Using It in Mineral Waste Disposal', *mdpi.comN Herrera, M Sinche Gonzalez, J Okkonen, R MollehuaraMinerals, 2023•mdpi.com* [Preprint]. Available at: https://doi.org/10.3390/min13111450.

Hinton, G., Srivastava, N. and Swersky, K. (2012) 'Neural Networks for Machine Learning Lecture 6a Overview of mini--batch gradient descent'.

Hlotse, D., Mbaya, R.K.K. and Shongwe, M.B. (2023) 'Optimization of flotation conditions in the beneficiation of PGMs tailings', *Physicochemical Problems of Mineral Processing*, 59(6). Available at: https://doi.org/10.37190/PPMP/176859.

Ho, T.K. (1995) 'Random decision forests', *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1, pp. 278–282. Available at: https://doi.org/10.1109/ICDAR.1995.598994.

Huang, Gao *et al.* (2015) 'Trends in extreme learning machines: A review', *Neural Networks*, 61, pp. 32–48. Available at: https://doi.org/10.1016/J.NEUNET.2014.10.001.

Hussain, U., Tammewar, S.S. and Bhattad, Y. (2024) 'Advancements in Process Control and Monitoring for Enhanced Efficiency and Quality', *Proceedings 2024, Vol. 105, Page 92*, 105(1), p. 92. Available at: https://doi.org/10.3390/PROCEEDINGS2024105092.

Hyndman, R. (2018) 'Forecasting: principles and practice'. Available at: https://www.academia.edu/download/64659947/Athanasopoulos__George__Hyndman__Rob_J._-_Forecasting__Principles_and_Practice_2018.pdf (Accessed: 19 February 2025).

Hyndman, R.J. and Koehler, A.B. (2006) 'Another look at measures of forecast accuracy', *International Journal of Forecasting*, 22(4), pp. 679–688. Available at: https://doi.org/10.1016/J.IJFORECAST.2006.03.001.

Isaac Abiodun, O. *et al.* (2018) 'State-of-the-art in artificial neural network applications: A survey', *Heliyon*, 4, p. e00938. Available at: https://doi.org/10.1016/j.heliyon.2018.

J, N. (2021) 'Overview of PGM processing', *angloplatinum.comJ NdlovuAnglo Platinum. Available online: http://www. angloamericanplatinum, 2021•angloplatinum.com* [Preprint]. Available at: https://www.angloplatinum.com/~/media/Files/A/Anglo-American-Platinum/presentations-and-speeches/standardbankconference-anglo-american-platinum-processing-111114.pdf (Accessed: 23 February 2025).

Jahedsaravani, A., Massinaei, M. and Zarie, M. (2023) 'Prediction of Froth Flotation Performance Using Convolutional Neural Networks', *Mining, Metallurgy and Exploration*, 40(3), pp. 923–937. Available at: https://doi.org/10.1007/S42461-023-00768-4.

Jiang, Y. *et al.* (2021) 'A Review on Soft Sensors for Monitoring, Control, and Optimization of Industrial Processes', *IEEE Sensors Journal*, 21(11), pp. 12868–12881. Available at: https://doi.org/10.1109/JSEN.2020.3033153.

Jin, H. *et al.* (2016) 'Dual learning-based online ensemble regression approach for adaptive soft sensor modeling of nonlinear time-varying processes', *Chemometrics and Intelligent Laboratory Systems*, 151, pp. 228–244. Available at: https://doi.org/10.1016/J.CHEMOLAB.2016.01.009.

Jumin, E. *et al.* (2020) 'Machine learning versus linear regression modelling approach for accurate ozone concentrations prediction', *Engineering Applications of Computational Fluid Mechanics*, 14(1), pp. 713–725. Available at: https://doi.org/10.1080/19942060.2020.1758792.

Jung, D. and Choi, Y. (2021) 'minerals Systematic Review of Machine Learning Applications in Mining: Exploration, Exploitation, and Reclamation'. Available at: https://doi.org/10.3390/min11020148.

Kadlec, P., Gabrys, B. and Strandt, S. (2009a) 'Data-driven Soft Sensors in the process industry', *Computers and Chemical Engineering*, 33(4), pp. 795–814. Available at: https://doi.org/10.1016/J.COMPCHEMENG.2008.12.012.

Kadlec, P., Gabrys, B. and Strandt, S. (2009b) 'Data-driven Soft Sensors in the process industry', *Computers & Chemical Engineering*, 33(4), pp. 795–814. Available at: https://doi.org/10.1016/J.COMPCHEMENG.2008.12.012.

Kadlec, P., Gabrys, B. and Strandt, S. (2009c) 'Data-driven Soft Sensors in the process industry', *Computers & Chemical Engineering*, 33(4), pp. 795–814. Available at: https://doi.org/10.1016/J.COMPCHEMENG.2008.12.012.

Kadlec, P., Gabrys, B. and Strandt, S. (2009d) 'Data-driven Soft Sensors in the process industry', *Computers & Chemical Engineering*, 33(4), pp. 795–814. Available at: https://doi.org/10.1016/J.COMPCHEMENG.2008.12.012.

Kadlec, P., Gabrys, B. and Strandt, S. (2009e) 'Data-driven Soft Sensors in the process industry', *Computers and Chemical Engineering*. Available at: https://doi.org/10.1016/j.compchemeng.2008.12.012.

Kadlec, P., Gabrys, B. and Strandt, S. (2009f) 'Data-driven Soft Sensors in the process industry', *Computers and Chemical Engineering*, 33(4), pp. 795–814. Available at: https://doi.org/10.1016/J.COMPCHEMENG.2008.12.012.

Kay, H. *et al.* (2024) 'Constructing a Symbolic Regression-Based Interpretable Soft Sensor for Industrial Data Analytics and Product Quality Control', *Industrial and Engineering Chemistry Research*, 63(9), pp. 4083–4092. Available at: https://doi.org/10.1021/ACS.IECR.3C04021/ASSET/IMAGES/LARGE/IE3C04021_0006.JPEG.

Kohavi, R. and John, G.H. (1997) 'Wrappers for feature subset selection', *Artificial Intelligence*, 97(1–2), pp. 273–324. Available at: https://doi.org/10.1016/S0004-3702(97)00043-X.

Kubosawa, S. *et al.* (2022) 'Soft sensors and process control using AI and dynamic simulation', *arxiv.orgS Kubosawa, T Onishi, Y TsuruokaarXiv preprint arXiv:2208.04373, 2022•arxiv.org* [Preprint]. Available at: https://doi.org/10.1252/kakoronbunshu.48.141.

Kuhn, M. and Johnson, K. (2013) 'Applied predictive modeling', *Applied Predictive Modeling*, pp. 1–600. Available at: https://doi.org/10.1007/978-1-4614-6849-3/COVER.

Laurila, H. *et al.* (2002) *Strategies for Instrumentation and Control of Flotation Circuits*, *Mineral Processing Plant Design, Practice, and Control Proceedings*. Edited by A.L. Mular, D.N. Halbe, and D.J. Barratt. Society for Mining, Metallurgy, and Exploration (Mineral processing plant design, practice, and control).

LeCun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning. Nature', *Nature*, 521(7553).

Lee, H. *et al.* (2007) 'Robust adaptive partial least squares modeling of a full-scale industrial wastewater treatment process', *ACS Publications* [Preprint]. Available at: https://doi.org/10.1021/ie061094.

Leiria, J. *et al.* (2023) 'Soft Sensors for Industrial Applications: Comparison of Variables Selection Methods and Regression Models', *2023 International Conference on Control, Automation and Diagnosis, ICCAD 2023* [Preprint]. Available at: https://doi.org/10.1109/ICCAD57653.2023.10152323.

Liaw, A. and Wiener, M. (2002) 'Classification and Regression by randomForest', 2(3). Available at: http://www.stat.berkeley.edu/ (Accessed: 1 November 2024).

Little, R.J.A. and Rubin, D.B. (2019) 'Statistical analysis with missing data', *Statistical Analysis with Missing Data*, pp. 1–449. Available at: https://doi.org/10.1002/9781119482260.

Liu, X. *et al.* (2009) 'Moving window kernel PCA for adaptive monitoring of nonlinear processes', *Chemometrics and Intelligent Laboratory Systems*, 96(2), pp. 132–143. Available at: https://doi.org/10.1016/J.CHEMOLAB.2009.01.002.

Loh, W.-Y. (2014) 'Fifty Years of Classification and Regression Trees Fifty Years of Classification and Regression Trees 1', *International Statistical Review*, 82, pp. 329–348. Available at: https://doi.org/10.1111/insr.12016.

Maldonado, M., Sbarbaro, D. and Lizama, E. (2007) 'Optimal control of a rougher flotation process based on dynamic programming', *Minerals Engineering*, 20(3), pp. 221–232. Available at: https://doi.org/10.1016/J.MINENG.2006.08.015.

Maulud, D., Maulud, D. and Abdulazeez, A.M. (2020) 'A Review on Linear Regression Comprehensive in Machine Learning', *Journal of Applied Science and Technology Trends*, 1(2), pp. 140–147. Available at: https://doi.org/10.38094/jastt1457.

McCoy, J.T. and Auret, L. (2019a) 'Machine learning applications in minerals processing: A review', *Minerals Engineering*, 132, pp. 95–109. Available at: https://doi.org/10.1016/J.MINENG.2018.12.004.

McCoy, J.T. and Auret, L. (2019b) 'Machine learning applications in minerals processing: A review', *Minerals Engineering*, 132, pp. 95–109. Available at: https://doi.org/10.1016/J.MINENG.2018.12.004.

Mehrabi, A., Mehrshad, N. and Massinaei, M. (2014) 'Machine vision based monitoring of an industrial flotation cell in an iron flotation plant', *International Journal of Mineral Processing*, 133, pp. 60–66. Available at: https://doi.org/10.1016/J.MINPRO.2014.09.018.

Mitchell, T. and Mitchell, T. (1997) *Machine learning*. Available at: http://www.pachecoj.com/courses/csc380_fall21/lectures/mlintro.pdf (Accessed: 3 November 2024).

Mohammadpoor, M. and Torabi, F. (2020) 'Big Data analytics in oil and gas industry: An emerging trend', *Petroleum*, 6(4), pp. 321–328. Available at: https://doi.org/10.1016/J.PETLM.2018.11.001.

Montesinos López, O.A., Montesinos López, A. and Crossa, J. (2022) 'Elements for Building Supervised Statistical Machine Learning Models', *Multivariate Statistical Machine Learning Methods for Genomic Prediction*, pp. 71–108. Available at: https://doi.org/10.1007/978-3-030-89010-0_3.

Montgomery, D.C.., Runger, G.C.. and Hubele, N.Faris. (2010a) 'Engineering statistics', p. 487. Available at: https://books.google.com/books/about/Engineering_Statistics_Student_Study_Edi.html?id=O-XmUTtpgSUC (Accessed: 18 November 2024).

Montgomery, D.C.., Runger, G.C.. and Hubele, N.Faris. (2010b) 'Engineering statistics', p. 487. Available at: https://books.google.com/books/about/Engineering_Statistics_Student_Study_Edi.html?id=O-XmUTtpgSUC (Accessed: 18 November 2024).

Murphy, K.P. (2022) *Probabilistic Machine Learning: An Introduction* . Available at: https://books.google.co.za/books?hl=en&lr=&id=wrZNEAAAQBAJ&oi=fnd&pg=PR27&dq=The+core+principle+behind+unsupervised+learning+is+that+it+identifies+inherent+similarities+within+the+data,+clustering+similar+instances+or+reducing+the+dimensionality+of+the+data+to+make+complex+datasets+more+interpretable+(Murphy,+2012).+&ots=LapdOFdkQk&sig=Z9gnoaKLBFW6XRa5c-tTcpR-XsY&redir_esc=y#v=onepage&q&f=false (Accessed: 22 March 2025).

Nakhaei, F. *et al.* (2022) 'Prediction of Sulfur Removal from Iron Concentrate Using Column Flotation Froth Features: Comparison of k-Means Clustering, Regression, Backpropagation Neural', *mdpi.comF Nakhaei, S Rahimi, M FathiMinerals, 2022•mdpi.com*, 12(11). Available at: https://doi.org/10.3390/min12111434.

Napier-Munn, T.J. *et al.* (1996) 'Mineral comminution circuits: their operation and optimisation', 2.

Navarra, A. *et al.* (2020) 'Quantitative Methods to Support Data Acquisition Modernization within Copper Smelters', *Processes 2020, Vol. 8, Page 1478*, 8(11), p. 1478. Available at: https://doi.org/10.3390/PR8111478.

Pack Kaelbling, L. *et al.* (1996) 'Reinforcement Learning: A Survey', *Journal of Artiicial Intelligence Research*, 4, pp. 237–285.

Pan, S.J. and Yang, Q. (2009) 'A Survey on Transfer Learning'. Available at: https://doi.org/10.1109/TKDE.2009.191.

Pan, S.J. and Yang, Q. (2010) 'A survey on transfer learning', *IEEE Transactions on Knowledge and Data Engineering*, 22(10), pp. 1345–1359. Available at: https://doi.org/10.1109/TKDE.2009.191.

Pani, A.K. and Mohanta, H.K. (2011) 'A survey of data treatment techniques for soft sensor design', *Chemical Product and Process Modeling*, 6(1). Available at: https://doi.org/10.2202/1934-2659.1536/HTML.

Pedregosa FABIANPEDREGOSA, F. *et al.* (2011) 'Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot', *Journal of Machine Learning Research*, 12, pp. 2825–2830. Available at: http://scikit-learn.sourceforge.net. (Accessed: 3 November 2024).

Perera, Y.S. *et al.* (2023) 'The role of artificial intelligence-driven soft sensors in advanced sustainable process industries: A critical review', *Engineering Applications of Artificial Intelligence*, 121, p. 105988. Available at: https://doi.org/10.1016/J.ENGAPPAI.2023.105988.

Popli, K. *et al.* (2018) 'Development of online soft sensors and dynamic fundamental model-based process monitoring for complex sulfide ore flotation', *Minerals Engineering*, 124, pp. 10–27. Available at: https://doi.org/10.1016/J.MINENG.2018.04.006.

Prasetyo, S.Y. *et al.* (2024) 'Exploring Cosine and Euclidean Metrics in K-Nearest Neighbors for Breast Cancer Diagnosis', *Proceedings - International Conference on Informatics and Computational Sciences*, pp. 330–335. Available at: https://doi.org/10.1109/ICICOS62600.2024.10636883.

Quinlan, J.R. (1986) ‘Induction of decision trees’, *Machine Learning 1986 1:1*, 1(1), pp. 81–106. Available at: https://doi.org/10.1007/BF00116251.

Raschka, S. (2018) ‘Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning’. Available at: http://arxiv.org/abs/1811.12808 (Accessed: 3 November 2024).

Rexie, J.A.M. *et al.* (2023) ‘Early Prediction of Diabetes using Several Machine Learning Algorithms’, *Proceedings of the 7th International Conference on Intelligent Computing and Control Systems, ICICCS 2023*, pp. 449–453. Available at: https://doi.org/10.1109/ICICCS56967.2023.10142749.

Rodriguez-Galiano, V. *et al.* (2015) ‘Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines’, *Ore Geology Reviews*, 71, pp. 804–818. Available at: https://doi.org/10.1016/J.OREGEOREV.2015.01.001.

Le Roux, J.D., Padhi, R. and Craig, I.K. (2015) ‘Optimal Control of Grinding Mill Circuit using Model Predictive Static Programming: A New Nonlinear MPC Paradigm’.

Russell, S.J. and Norvig, P. (2016) *Artificial intelligence : a modern approach*, *Pearson*. Pearson. Available at: https://thuvienso.hoasen.edu.vn/handle/123456789/8967 (Accessed: 29 October 2024).

Sah, S. (2020) ‘Machine learning: a review of learning types’. Available at: https://www.preprints.org/manuscript/202007.0230 (Accessed: 18 November 2024).

Schönert, K. (1988) ‘A first survey of grinding with high-compression roller mills’, *International Journal of Mineral Processing*, 22(1–4), pp. 401–412. Available at: https://doi.org/10.1016/0301-7516(88)90075-0.

Shanthamallu, U., (IISA), A.S.-… & A. and 2017, undefined (no date) ‘A brief survey of machine learning methods and their sensor and IoT applications’, *ieeexplore.ieee.orgUS Shanthamallu, A Spanias, C Tepedelenlioglu, M Stanley2017 8th International Conference on Information, Intelligence, 2017•ieeexplore.ieee.org* [Preprint]. Available at: https://ieeexplore.ieee.org/abstract/document/8316459/ (Accessed: 24 September 2024).

Shanthamallu, U.S. *et al.* (2017) ‘A brief survey of machine learning methods and their sensor and IoT applications’, *2017 8th International Conference on Information, Intelligence, Systems and Applications, IISA 2017*, 2018-January, pp. 1–8. Available at: https://doi.org/10.1109/IISA.2017.8316459.

Shardt, Y.A.W. and Huang, B. (2011) ‘Closed-loop identification with routine operating data: Effect of time delay and sampling time’, *Journal of Process Control*, 21(7), pp. 997–1010. Available at: https://doi.org/10.1016/J.JPROCONT.2011.06.015.

Singh, N. (2023) *Refining KNN Regressor: Optimizing Model Fit with Distance Metrics, AI Mind*. Available at: https://pub.aimind.so/refining-knn-regressor-optimizing-model-fit-with-distance-metrics-edd5a59ef291 (Accessed: 3 October 2024).

Singh, S. and Vazirani, V. (2022) ‘Classification vs Clustering: Ways for Diabetes Detection’, *2022 IEEE 7th International conference for Convergence in Technology, I2CT 2022* [Preprint]. Available at: https://doi.org/10.1109/I2CT54291.2022.9825053.

Singh, V. and Mohan Rao, S. (2005) ‘Application of image processing and radial basis neural network techniques for ore sorting and ore classification’, *Minerals Engineering*, 18(15), pp. 1412–1420. Available at: https://doi.org/10.1016/J.MINENG.2005.03.003.

Smola, A.J. and Schölkopf, B. (2004) ‘A tutorial on support vector regression’, *Statistics and Computing*, 14(3), pp. 199–222. Available at: https://doi.org/10.1023/B:STCO.0000035301.49549.88.

Souza, F.A.A., Araújo, R. and Mendes, J. (2016) 'Review of soft sensor methods for regression applications', *Chemometrics and Intelligent Laboratory Systems*, 152, pp. 69–79. Available at: https://doi.org/10.1016/J.CHEMOLAB.2015.12.011.

Steyn, C. and Sandrock, C. (2021) 'Causal model of an industrial platinum flotation circuit', *Control Engineering Practice*, 109, p. 104736. Available at: https://doi.org/10.1016/J.CONENGPRAC.2021.104736.

Sutton, R.S. and Barto, A.G. (2018) *Sutton & Barto Book: Reinforcement Learning: An Introduction*. 2nd edn. Cambridge: MIT Press. Available at: http://www.incompleteideas.net/book/the-book-2nd.html (Accessed: 22 March 2025).

Szmigiel, A. *et al.* (2024) 'Advancements in Machine Learning for Optimal Performance in Flotation Processes: A Review', *Minerals 2024, Vol. 14, Page 331*, 14(4), p. 331. Available at: https://doi.org/10.3390/MIN14040331.

Walczak, B. and Massart, D.L. (2001) 'Dealing with missing data: Part I', *Chemometrics and Intelligent Laboratory Systems*, 58(1), pp. 15–27. Available at: https://doi.org/10.1016/S0169-7439(01)00131-9.

Wang, M., Lu, Y. and Qin, J. (2020) 'A dynamic MLP-based DDoS attack detection method using feature selection and feedback', *Computers & Security*, 88, p. 101645. Available at: https://doi.org/10.1016/J.COSE.2019.101645.

Wang, X. *et al.* (2005) 'Process monitoring approach using fast moving window PCA', *ACS PublicationsX Wang, U Kruger, GW IrwinIndustrial & engineering chemistry research, 2005•ACS Publications*, 44(15), pp. 5691–5702. Available at: https://doi.org/10.1021/ie048873f.

Widodo, A. and Yang, B.S. (2007) 'Support vector machine in machine condition monitoring and fault diagnosis', *Mechanical Systems and Signal Processing*, 21(6), pp. 2560–2574. Available at: https://doi.org/10.1016/J.YMSSP.2006.12.007.

Wills, B. and Finch, J. (2015) *Wills' mineral processing technology: an introduction to the practical aspects of ore treatment and mineral recovery*. Available at: https://books.google.com/books?hl=en&lr=&id=uMWcBAAAQBAJ&oi=fnd&pg=PP1&dq=%E2%80%A2%09Wills.+B.A.+and+Finch.+J.A.+(2015)+Wills%27+Mineral+Processing+Technology:+An+Introduction+to+the+Practical+Aspects+of+Ore+Treatment+and+Mineral+Recovery.+8th+ed.+Amsterdam:+Elsevier&ots=LBlgPsmF8T&sig=wDHV8U8i6qKTGe9uOf9aJyzePWw (Accessed: 29 October 2024).

Wold, S. (1995) 'Chemometrics; what do we mean with it, and what do we want from it?', *Chemometrics and Intelligent Laboratory Systems*, 30(1), pp. 109–115. Available at: https://doi.org/10.1016/0169-7439(95)00042-9.

Xiong, Q. *et al.* (2024) 'Hyperparameter tuning of an artificial neural network (ANN) for laboratory acoustic emission source location', *onepetro.orgQ Xiong, EK Johnson, CS Wu, JC HamptonARMA US Rock Mechanics/Geomechanics Symposium, 2024•onepetro.org* [Preprint]. Available at: https://onepetro.org/ARMAUSRMS/proceedings-abstract/ARMA24/ARMA24/549381 (Accessed: 2 October 2024).

Yang, Q. (2024) 'Performance analysis of k-Nearest Neighbors classification on Reuters news article datasets', *Applied and Computational Engineering*, 55(1), pp. 184–189. Available at: https://doi.org/10.54254/2755-2721/55/20241444.

Ye, L. *et al.* (2020) 'An active approach to space-reduced NCO tracking and output feedback optimizing control for batch processes with parametric uncertainty', *Journal of Process Control*, 89, pp. 30–44. Available at: https://doi.org/10.1016/J.JPROCONT.2020.03.003.

Yosinski, J. *et al.* (no date) 'How transferable are features in deep neural networks?', *proceedings.neurips.ccJ Yosinski, J Clune, Y Bengio, H LipsonAdvances in neural information processing systems, 2014•proceedings.neurips.cc* [Preprint]. Available at: https://proceedings.neurips.cc/paper/5347-how-transferable-are-features-in-deep-n%E2%80%A6 (Accessed: 7 October 2024).

Young, A. and Rogers, P. (2019) 'A Review of Digital Transformation in Mining', *Mining, Metallurgy and Exploration*, 36(4), pp. 683–699. Available at: https://doi.org/10.1007/S42461-019-00103-W.

Zhou, P. *et al.* (2020) 'Data-driven monitoring and diagnosing of abnormal furnace conditions in blast furnace ironmaking: An integrated PCA-ICA method', *ieeexplore.ieee.orgP Zhou, R Zhang, J Xie, J Liu, H Wang, T ChaiIEEE Transactions on Industrial Electronics, 2020•ieeexplore.ieee.org* [Preprint]. Available at: https://ieeexplore.ieee.org/abstract/document/8967236/ (Accessed: 18 January 2025).

Zhou, Y., Li, D. and Xi, Y. (2020) 'Data-Driven Predictive Control for Continuous-Time Industrial Processes with Completely Unknown Dynamics'. Available at: https://arxiv.org/abs/2012.03452v1 (Accessed: 19 February 2025).

Zou, H. and Hastie, T. (2005) 'Regularization and Variable Selection Via the Elastic Net', *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2), pp. 301–320. Available at: https://doi.org/10.1111/J.1467-9868.2005.00503.X.

Zulu, N.G., Mvita, ; M J and Thethwayo, B. (no date) 'A Systematic Review on Applications of Artificial Neural Networks in the Extraction Metallurgy Industry'.

9. Appendices

Appendix A – Process and Process Data

Table A 1: *Raw process data collected*

0	datetime	1915 non-null object
1	407WIT107B/PV.V	1908 non-null object
2	L:MOGN.4T.Recovery.Daily.Substitute	1876 non-null object
3	L:MOGN.Conc.PGM.Mass.DailyEstimate.4T	1915 non-null float64
4	L:MOGN.4T.Conc.Assay.Daily.Substitute	1861 non-null object
5	L:MOGN.Conc.PGM.Assay.DailyEstimate.4T	1915 non-null float64
6	L:MOGN.4T.Tails.Assay.Daily.Substitute	1867 non-null object
7	L:MOGN.4T.SH.Assay.Daily.Substitute	1886 non-null object
8	440FIC071/PID.PV_IN	1908 non-null object
9	440FIC072/PID.PV_IN	1907 non-null object
10	FT-CELL-440-FT-008.MASS-PULL-FRACTION-PV.READING	1915 non-null float64
11	FT-CELL-440-FT-009.MASS-PULL-FRACTION-PV.READING	1915 non-null float64
12	CAM-440-FT-008-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING	1847 non-null object
13	CAM-440-FT-009-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING	1792 non-null object
14	CAM-440-FT-008-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING	1516 non-null object
15	CAM-440-FT-009-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING	1516 non-null object
16	440LIC108/PID.PV_IN	1907 non-null object
17	440LIC109/PID.PV_IN	1907 non-null object
18	440FIC128/PID.PV_IN	1898 non-null object
19	440FIC129/PID.PV_IN	1904 non-null object
20	FT-CELL-440-FT-008.FLOW-FROTHER.READING	1908 non-null object
21	MOGN.N-SEC.SUMP-408-SU-002.FLOW-COLLECTOR.PV	1788 non-null object
22	MOGN.N-SEC.FT-CELL-440-FT-008.FLOW-COLLECTOR.PV	1700 non-null object
23	MOGN.N-SEC.SUMP-408-SU-002.FLOW-PROMOTER.PV	1786 non-null object
24	MOGN.N-SEC.FT-CELL-440-FT-008.FLOW-PROMOTER.PV	1690 non-null object
25	MOGN.N-SEC.PUMP-440-PP-202-03.FLOW.PV	1792 non-null object
26	MOGN.N-SEC.PUMP-440-PP-202-03.SG.PV	1792 non-null object

27	CAM-440-FT-008-FROTH-CAMERA.FROTH-VELOCITY.READING	1847 non-null object
28	CAM-440-FT-008-FROTH-CAMERA.LEVEL.READING	1676 non-null object
29	440FT008.froth_height	1549 non-null object
30	440FT009.froth_height	1549 non-null object
31	CAM-440-FT-009-FROTH-CAMERA.FROTH-VELOCITY.READING	1792 non-null object
32	CAM-440-FT-009-FROTH-CAMERA.LEVEL.READING	1704 non-null object
33	440FT008.pulp_bubble_size_mean_mm	565 non-null object
34	440FT009.pulp_bubble_size_mean_mm	590 non-null object
35	440FT008.pulp_Jg	565 non-null object
36	440FT009.pulp_Jg	590 non-null object
37	FT-CELL-440-FT-008.AIR-FLOW.READING	1905 non-null object
38	FT-CELL-440-FT-009.AIR-FLOW.READING	1912 non-null object
39	CAM-440-FT-008-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING	1711 non-null object
40	CAM-440-FT-009-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING	1711 non-null object
41	CAM-440-FT-009-FROTH-CAMERA.ANGLE-VELOCITY.READING	1741 non-null object
42	CAM-440-FT-008-FROTH-CAMERA.ANGLE-VELOCITY.READING	1741 non-null object
43	FT-CELL-440-FT-008.AIR-FLOW.READING.1	1905 non-null object
44	FT-CELL-440-FT-008.FLOW-COLLECTOR.READING	1913 non-null object
45	FT-CELL-440-FT-008.FLOW-DEPRESSANT.READING	1908 non-null object
46	FT-CELL-440-FT-008.FLOW-FROTHER.READING.1	1908 non-null object
47	FT-CELL-440-FT-008.FLOW-MP-PV.RAW	1905 non-null object
48	FT-CELL-440-FT-008.FLOW-PROMOTER.READING	1877 non-null object
49	FT-CELL-440-FT-008.LEVEL.READING	1912 non-null object
50	FT-CELL-440-FT-008.MASS-PULL-FRACTION-PV.FROTH-CAM	1915 non-null float64
51	FT-CELL-440-FT-008.RESIDENCE-TIME.RAW	1905 non-null object
52	MOGN.N-SEC.FT-CELL-440-FT-008.AIR-FLOW.PV	1643 non-null object
53	FT-CELL-440-FT-009.AIR-FLOW.READING.1	1912 non-null object
54	FT-CELL-440-FT-009.LEVEL.READING	1912 non-null object
55	FT-CELL-440-FT-009.MASS-PULL-FRACTION-PV.FROTH-CAM	1912 non-null object
56	FT-CELL-440-FT-009.RESIDENCE-TIME.RAW	1911 non-null object
57	MOGN.N-SEC.SUMP-408-SU-002.LEVEL.PV	1780 non-null object
58	MOGN.N-SEC.SUMP-408-SU-002.DILUTION-WATER-FLOW.PV	1756 non-null object
59	MOGN.N-SEC.SUMP-408-SU-002.CONCENTRATION-01.PV	1791 non-null object
60	440SA031_P1/PV.V.1	1895 non-null object
61	440FT008_1/DOL.STA_IPER	1914 non-null float64
62	440FT009_1/DOL.STA_IPER	1914 non-null float64
63	FT-CELL-440-FT-010.AIR-FLOW.READING	1902 non-null object
64	FT-CELL-440-FT-011.AIR-FLOW.READING	1910 non-null object
65	FT-CELL-440-FT-012.AIR-FLOW.READING	1911 non-null object
66	FT-CELL-440-FT-013.AIR-FLOW.READING	1909 non-null object
67	440FIT325/PV.V	1903 non-null object
68	440DT326/PV.V	1912 non-null object
69	440FIT523/PV.V	1903 non-null object
70	440DT524/PV.V	1906 non-null object
71	CAM-440-FT-010-FROTH-CAMERA.FROTH-VELOCITY.READING	1820 non-null object
72	CAM-440-FT-010-FROTH-CAMERA.LEVEL.READING	1668 non-null object
73	*CAM-440-FT-011-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING*	1815 non-null object

74	CAM-440-FT-010-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING	1515 non-null object
75	CAM-440-FT-011-FROTH-CAMERA.FROTH-VELOCITY.READING	1815 non-null object
76	CAM-440-FT-011-FROTH-CAMERA.LEVEL.READING	1682 non-null object
77	CAM-440-FT-011-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING	1515 non-null object
78	CAM-440-FT-010-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING	1704 non-null object
79	CAM-440-FT-013-FROTH-CAMERA.FROTH-VELOCITY.READING	1823 non-null object
80	CAM-440-FT-013-FROTH-CAMERA.LEVEL.READING	1698 non-null object
81	CAM-440-FT-014-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING	1830 non-null object
82	CAM-440-FT-013-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING	1515 non-null object
83	CAM-440-FT-014-FROTH-CAMERA.FROTH-VELOCITY.READING	1856 non-null object
84	CAM-440-FT-014-FROTH-CAMERA.LEVEL.READING	1674 non-null object
85	CAM-440-FT-014-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING	1515 non-null object
86	CAM-440-FT-012-FROTH-CAMERA.LEVEL.READING	1677 non-null object
87	CAM-440-FT-012-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING	1704 non-null object
88	CAM-440-FT-013-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING	1703 non-null object
89	CAM-440-FT-014-FROTH-CAMERA.FROTH-VOLUME-PRODUCED.READING	1712 non-null object
90	CAM-440-FT-012-FROTH-CAMERA.FROTH-VELOCITY.READING	1849 non-null object
91	CAM-440-FT-012-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING	1849 non-null object

***Table A2**: Process tag, description and units of collected data*

Process Tag	Description	Units
407WIT107B/PV.V	Ore Feed Transfer Conveyor #1 Belt Scale (High Accuracy) process value	T/h
440FIC071/PID.PV_IN	Rougher Surge 440TK020 440PP201 Delivery Flw Control process variable	m^3/h
440FIC072/PID.PV_IN	Rougher Surge 440TK020 Level Flow Control process variable	m^3/h
440DIT071/PV.V	Rougher Surge 440TK020 Line A Tank Discharge Density Output value	kg/L
440DIT072/PV.V	Rougher Surge 440TK020 Line C Tank Discharge Density Output value	kg/L
440FT008_1/DOL.STA_IPER	Rougher Flot Cell Agitator 440FT008 current draw percentage	%
440FT009_1/DOL.STA_IPER	Rougher Flot Cell Agitator 440FT009 current draw percentage	%
FT-CELL-440-FT-00-8-9.MASS-PULL-PV.READING	Rghr-02 Dorr-Oliver Grouped Flotation Cell 440-FT-00-8-9 Mass-Pull-Pv Reading	%

FT-CELL-440-FT-008.MASS-PULL-FRACTION-PV.READING	Rghr-02 Dorr-Oliver Flotation Cell 440-FT-008 Mass-Pull-Fraction-Pv Reading	QUANTITY
FT-CELL-440-FT-009.MASS-PULL-FRACTION-PV.READING	Rghr-02 Dorr-Oliver Flotation Cell 440-FT-009 Mass-Pull-Fraction-Pv Reading	QUANTITY
CAM-440-FT-008-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING	Rghr-02 Camera 440-FT-008-Froth-Camera Abs-Froth-Velocity Reading	mm/s
CAM-440-FT-009-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING	Rghr-02 Camera 440-FT-009-Froth-Camera Abs-Froth-Velocity Reading	mm/s
CAM-440-FT-008-FROTH-CAMERA.LEVEL.READING	Rghr-02 Camera 440-FT-008-Froth-Camera Level Reading	mm
CAM-440-FT-008-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING	Rghr-02 Camera 440-FT-008-Froth-Camera Mean-Bubble-Size Reading	QUANTITY
CAM-440-FT-009-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING	Rghr-02 Camera 440-FT-009-Froth-Camera Mean-Bubble-Size Reading	QUANTITY
440LIC108/PID.PV_IN	Rough Float Bank2 Cell8 440LT108 Froth Lvl Cntrl process variable	%
440LIC109/PID.PV_IN	Rough Float Bank2 Cell9 440LT109 Froth Lvl Cntrl process variable	%
440FIC128/PID.PV_IN	Rough Float Bank2 Cell8 440FT128 Air Flow Cntrl process variable	m/s
440FIC129/PID.PV_IN	Rough Float Bank2 Cell9 440FT129 Air Flow Cntrl process variable	m/s
FT-CELL-440-FT-008.FLOW-FROTHER.READING	Rghr-02 Dorr-Oliver Flotation Cell 440-FT-008 - FLOW-FROTHER Reading	l/hr
MOGN.N-SEC.SUMP-408-SU-002.FLOW-COLLECTOR.PV	S-Milling Discharge Sump 408-SU-002 Flow-Collector Raw	l/hr
MOGN.N-SEC.FT-CELL-440-FT-008.FLOW-COLLECTOR.PV	Rghr-02 Dorr-Oliver Flotation Cell 440-FT-008 Flow-Collector Raw	l/hr
MOGN.N-SEC.SUMP-408-SU-002.FLOW-PROMOTER.PV	S-Milling Discharge Sump 408-SU-002 Flow-Promoter Raw	l/hr
MOGN.N-SEC.FT-CELL-440-FT-008.FLOW-PROMOTER.PV	Rghr-02 Dorr-Oliver Flotation Cell 440-FT-008 Flow-Promoter Raw	l/hr
MOGN.N-SEC.PUMP-440-PP-202-03.FLOW.PV	Rghr-Feed-Surge Pump 440-PP-202-03 Flow Raw	m^3/hr
MOGN.N-SEC.PUMP-440-PP-202-03.SG.PV	Rghr-Feed-Surge Pump 440-PP-202-03 Sg Raw	SG
440SA031_P1/PV.V	BlueCube Sampler Particles Pass 75 Output value	%
407CV001_2/DOL.QRUN	407CV001 Hydraulic Pump Motor Drive StdBy running	
407WIT107B/PV.V	Ore Feed Transfer Conveyor #1 Belt Scale (High Accuracy) process value	T/h
L:MOGN.4T.Tails.Assay.Daily.Substitute	Daily Estimate Concentrator PGM calculated from smelter receipts	g/t

L:MOGN.4T.Conc.Assay.Daily.Substitute	Daily Concentrator Assay Concentrator PGM Assay Substitute	g/t
L:MOGN.4T.SH.Assay.Daily.Substitute	Daily Concentrator PGM Assay Estimate calculated from tails Estimate Concentrator PGM Assay calculated from Sample head	g/t
L:MOGN.4T.Tails.Assay.Daily.Substitute	Daily Estimate Concentrator PGM calculated from smelter receipts	g/t
MOGN.N-SEC.SUMP-408-SU-002.FLOW-COLLECTOR.PV	S-Milling Discharge Sump 408-SU-002 Flow-Collector Raw	l/hr
MOGN.N-SEC.FT-CELL-440-FT-008.FLOW-COLLECTOR.PV	Rghr-02 Dorr-Oliver Flotation Cell 440-FT-008 Flow-Collector Raw	l/hr
MOGN.N-SEC.SUMP-408-SU-002.FLOW-PROMOTER.PV	S-Milling Discharge Sump 408-SU-002 Flow-Promoter Raw	l/hr
MOGN.N-SEC.FT-CELL-440-FT-008.FLOW-PROMOTER.PV	Rghr-02 Dorr-Oliver Flotation Cell 440-FT-008 Flow-Promoter Raw	l/hr
MOGN.N-SEC.PUMP-440-PP-202-03.FLOW.PV	Rghr-Feed-Surge Pump 440-PP-202-03 Flow Raw	m^3/hr
MOGN.N-SEC.PUMP-440-PP-202-03.SG.PV	Rghr-Feed-Surge Pump 440-PP-202-03 Sg Raw	SG
440SA031_P1/PV.V	BlueCube Sampler Particles Pass 75 Output value	%
CAM-440-FT-009-FROTH-CAMERA.ANGLE-VELOCITY.READING	Froth camera angle velocity Flotation Cell 440-FT-009	mm/s
440FT009.pulp_bubble_size_mean_mm	Rghr-02 Camera 440-FT-009-Pulp- Mean-Bubble-Size Reading	mm
440FT008.pulp_Jg	Rghr-02 Camera 440-FT-008- Pulp Superficial velocity Reading	mm/s
440FT009.froth_height	Rghr-02 Flotation Cell 9 froth_height	mm
FT-CELL-440-FT-008. RESIDENCE-TIME.RAW	Rghr-02 Flotation Cell 8 RESIDENCE-TIME	1/s

The bar plot in Figure A1 illustrates the number of missing values identified in the dataset prior to data cleaning. The data is organized in descending order, presenting columns with the highest number of missing values first, followed by those with fewer missing values. Among the process variables the pulp data was found to be the least reliable exhibiting the highest number of missing values compared to the other variables.

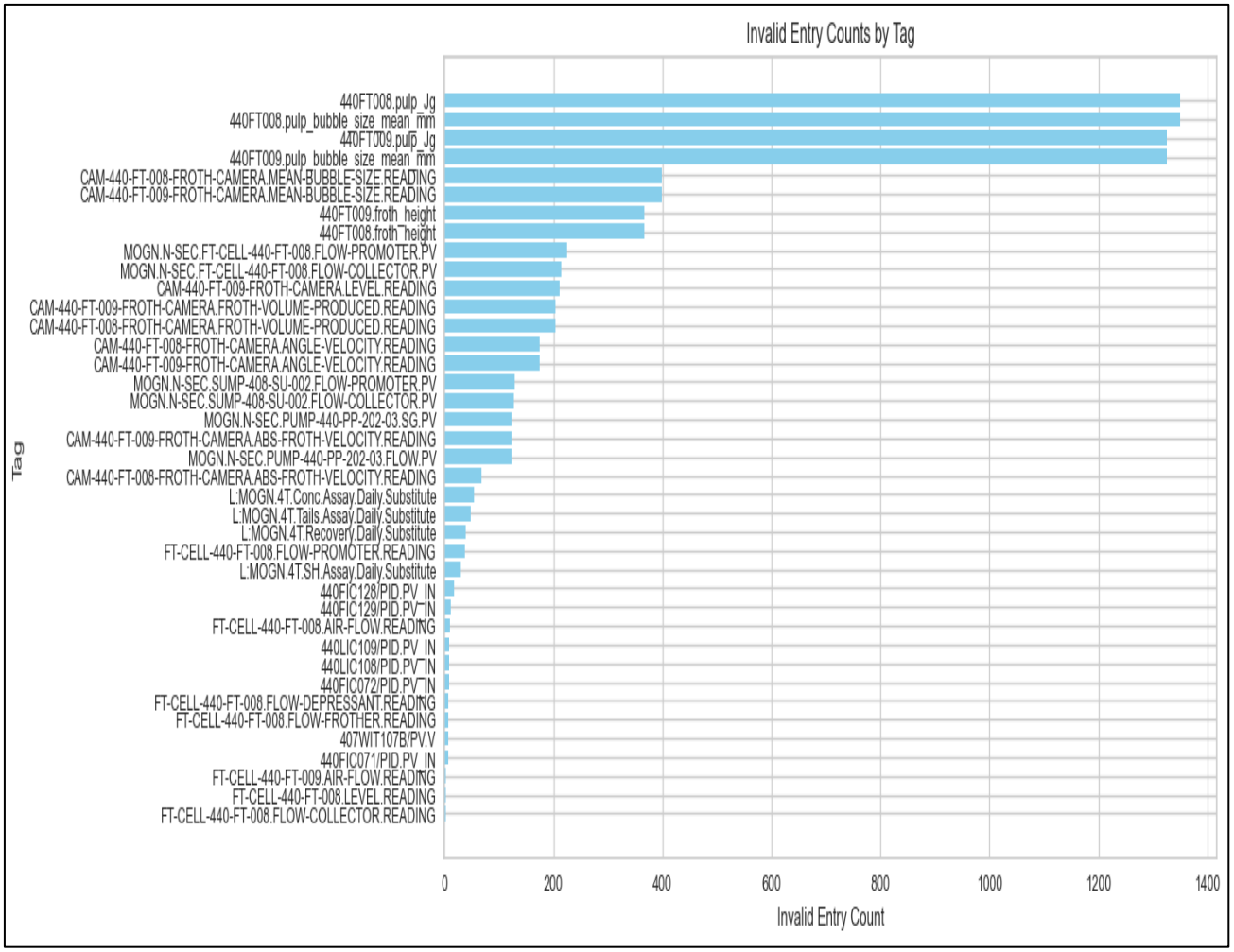


Figure A 1: *The amount of non-numeric values recognised as invalid in python for each process variables.*

Feature Engineering was conducted in the research where PGMs mass flows were suggested to be our target variables due to the concentrate grade being subject to gauge. The mass flows were calculate using the following formular:

$$Mass\ flowrate = Density\ x\ volumetric\ flowrate \quad \text{.....2A}$$

$$m(\frac{kg}{h}) = \rho(\frac{kg}{m^3}) * \nu(\frac{m^3}{h}) \quad \text{.....3A}$$

This feature engineering approach was implemented to enhance the model's generalization capabilities, motivated by the understanding that concentrate measurements are prone to gauge variability. Such inconsistencies are further exacerbated by fluctuations originating from flotation cells, scavengers, and cleaners in the production process. To mitigate the impact of these measurement challenges, mass flow rates were explicitly modeled as input features. This adjustment aimed to reduce noise in the data and address potential performance degradation caused by unreliable gauge readings. The following figures are the code created to implement the above catering both tank 8 and 9 as well as all the remaining tank 10 to 14.

```
# Calculate the Mass Flow Rate
Data_avg_final_cleaned1['Volumetric_Flow_l_h_FT8_9'] = Data_avg_final_cleaned1['440FIT325/PV.V'] * 1000
Data_avg_final_cleaned1['Volumetric_Flow_l_h_FT10_14'] = Data_avg_final_cleaned1['440FIT523/PV.V'] *
1000

Data_avg_final_cleaned1['Mass_Flow_Rate(FT8_9)'] = Data_avg_final_cleaned1['Volumetric_Flow_l_h_FT8_9']
* Data_avg_final_cleaned1['440DT326/PV.V']
Data_avg_final_cleaned1['Mass_Flow_Rate(FT10_14)'] =
Data_avg_final_cleaned1['Volumetric_Flow_l_h_FT10_14'] * Data_avg_final_cleaned1['440DT524/PV.V']

Data_avg_final_cleaned1['Mass_Flow_Rate_Total'] = Data_avg_final_cleaned1['Mass_Flow_Rate(FT10_14)'] +
Data_avg_final_cleaned1['Mass_Flow_Rate(FT8_9)']
print(Data_avg_final_cleaned1)
```

```
Data_avg_final_cleaned1['mass_flow_rate_total_tons_per_hour'] =
Data_avg_final_cleaned1['Mass_Flow_Rate_Total'] * 0.001

Data_avg_final_cleaned1['mass_flow_rate_from Recovery_grams_per_hour'] =
(Data_avg_final_cleaned1['recovery_4T_percentage'] *
Data_avg_final_cleaned1['mass_flow_rate_total_tons_per_hour']/1000)

Data_avg_final_cleaned1['mass_flow_rate_total_grams_per_hour'] =
(Data_avg_final_cleaned1['L:MOGN.4T.Conc.Assay.Daily.Substitute'] *
Data_avg_final_cleaned1['mass_flow_rate_total_tons_per_hour']/1000)

Data_avg_final_cleaned1['PGM_mass_flow_rate_total_recovery(%)'] = (0.645 *
Data_avg_final_cleaned1['mass_flow_rate_total_tons_per_hour']/1000)
Data_avg_final_cleaned1['4T.Recovery PGMS MASS FLOW daily'] = Data_avg_final_cleaned1['4T.Recovery
daily'] * Data_avg_final_cleaned1['mass_flow_rate_total_tons_per_hour']

# Percentage of Mass_Flow_Rate(FT8_9)
percentage_of_massflowFT8_9 = (sum(Data_avg_final_cleaned1['Mass_Flow_Rate(FT8_9)']) /
sum(Data_avg_final_cleaned1['Mass_Flow_Rate_Total'])) * 100

print(percentage_of_massflowFT8_9)
```

Figure A 2 A: *Python Code for mass flowrate calculations*

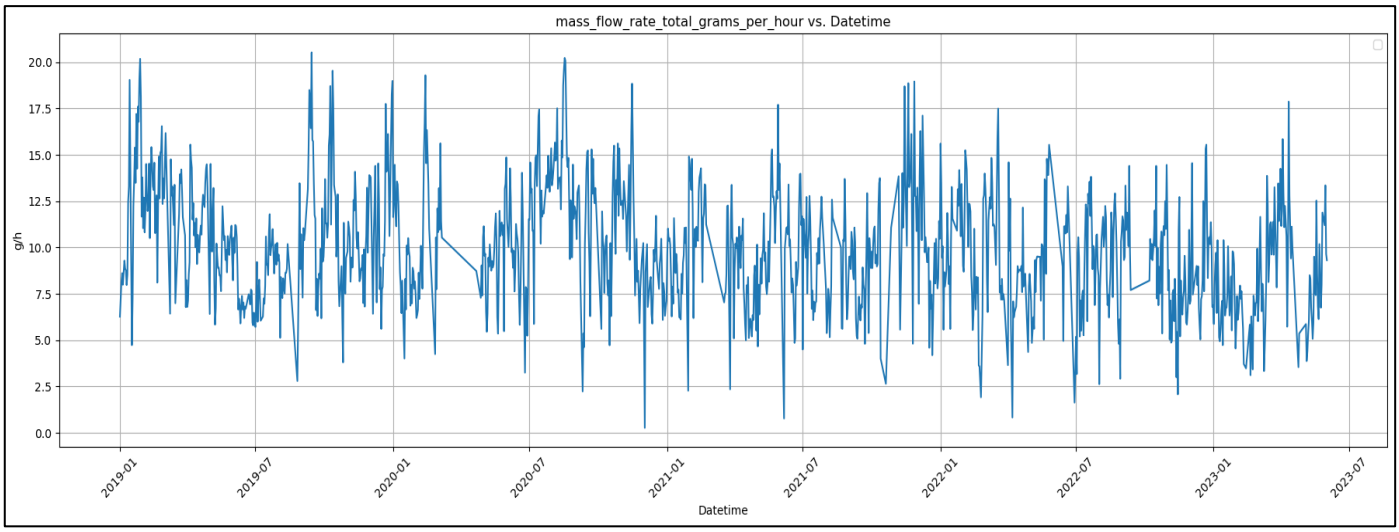


Figure A 3: *Mass flowrate sum for cells 8 and 9 overtime in tonnes per hour*

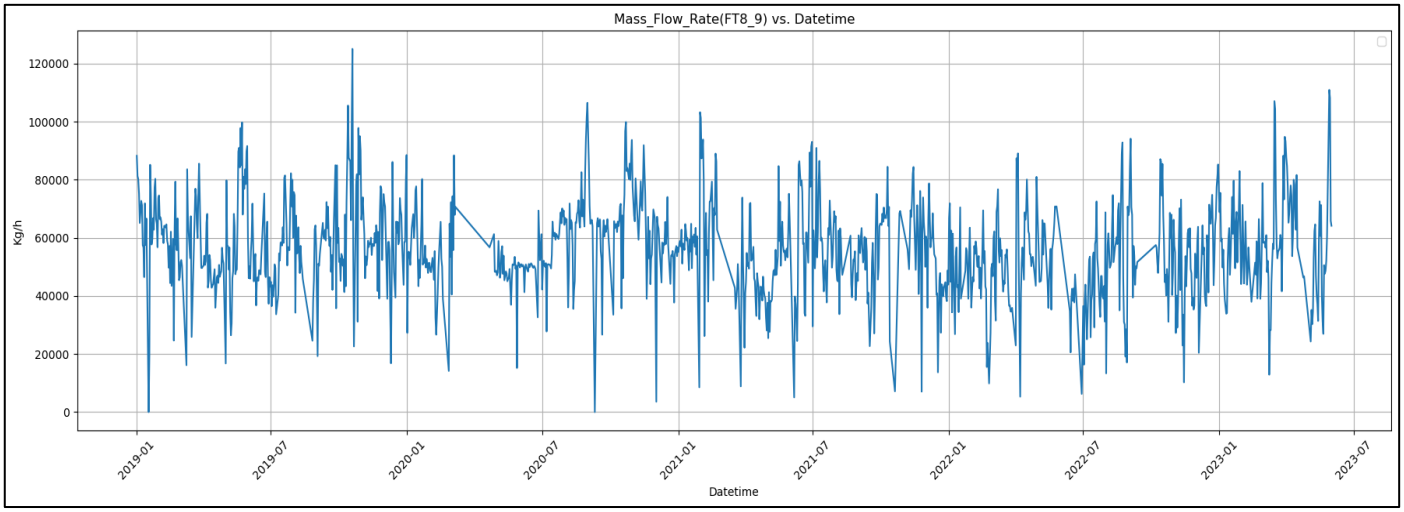


Figure A 3: *Mass flowrate sum for cells 10 and 14 overtime in kg per hour*

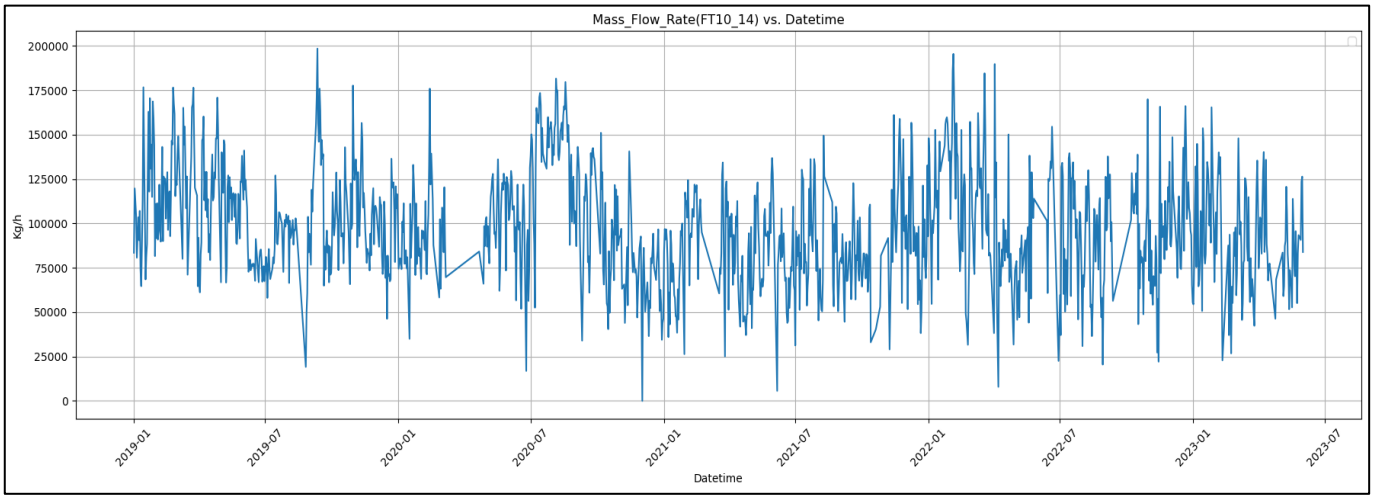
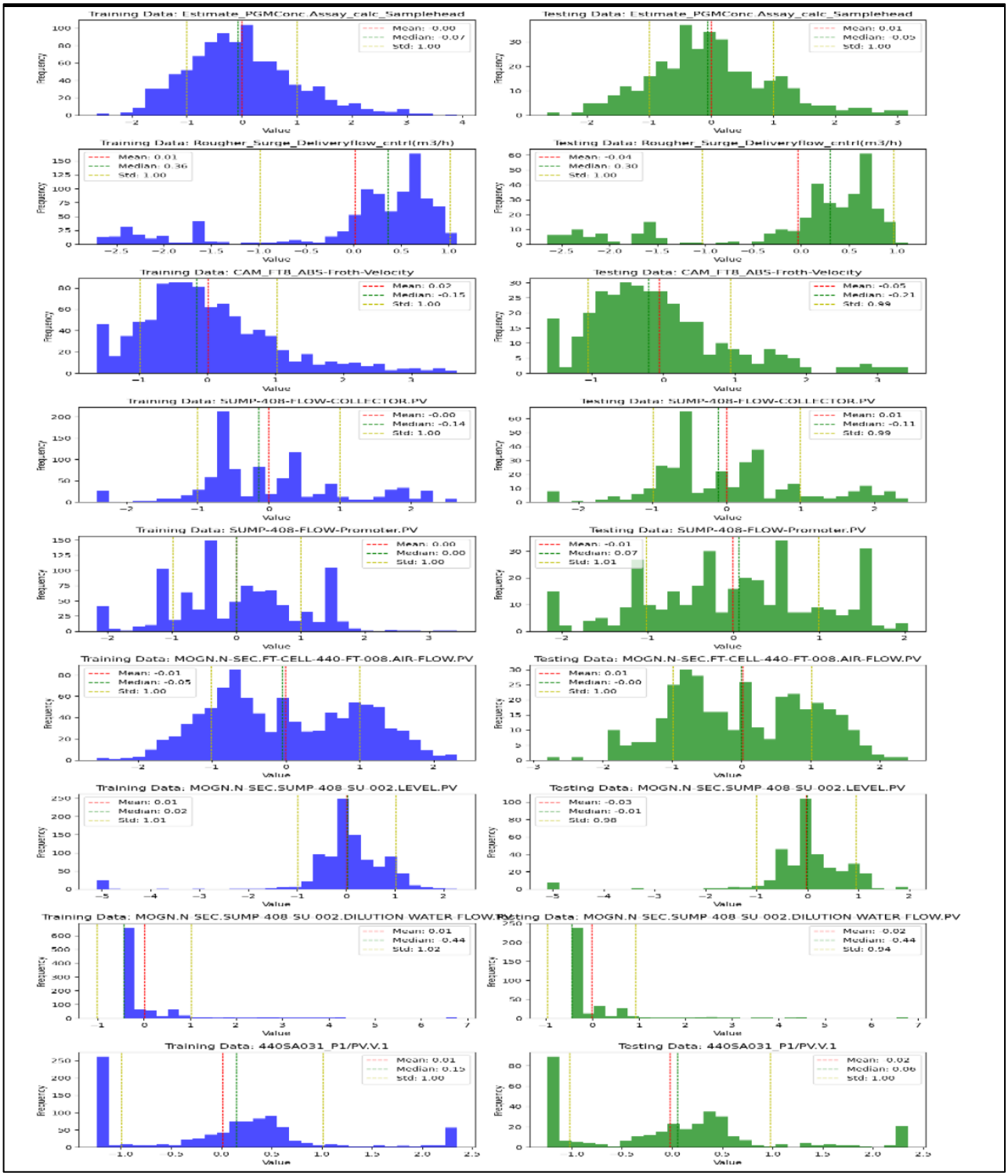


Figure A 5: *Mass flowrate sum for cells all the cells from 8 to 14 overtime in kg per hour.*

Appendix B- Descriptive Statistics

The below represents the statistics of the selected variables for both the training and testing set. This is essential to ensure the two sets almost have the same distribution so that the algorithm navigates easily in trying to predict the unseen variables. The green graphs represent the testing unseen data set, and the blue represents the training set.



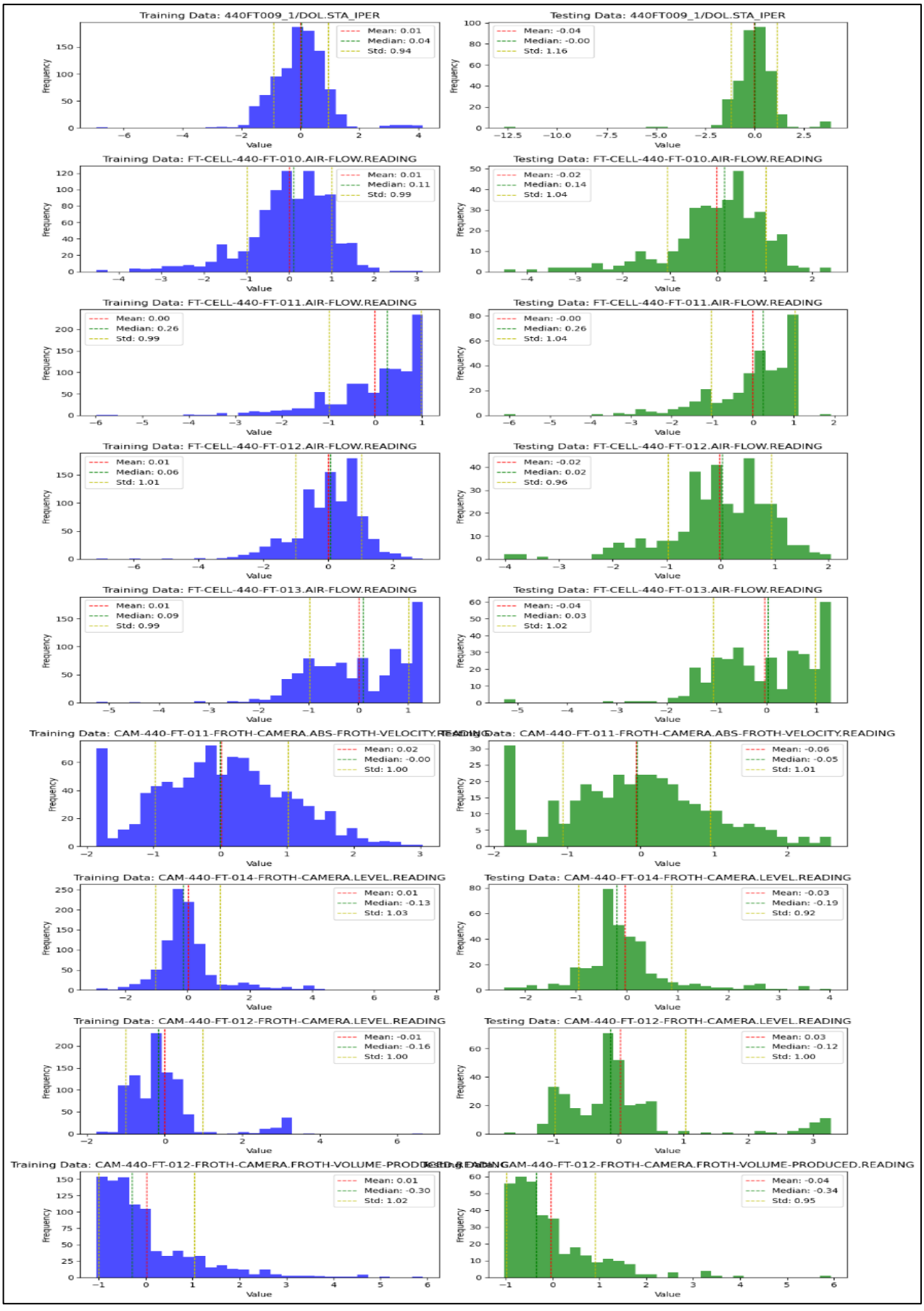


Figure B 1: *Statistics for the selected variables training and testing set.*

Appendix C - Python Codes

```
In [3]: import pandas as pd
        pd.plotting.register_matplotlib_converters()
        %matplotlib inline
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        import sklearn as skl
        from sklearn.linear_model import LinearRegression
        from sklearn.preprocessing import StandardScaler
        from sklearn.decomposition import PCA
        from sklearn.feature_selection import SelectKBest, f_regression
        print('Setup Complete')

        Setup Complete
```

Figure C 2: *Python Third-party Libraries and built-in modules*

```
In [27]: import pandas as pd

         def convert_to_nan(Data_avg_final, datetime_col_name):
           """Converts all text in columns except the datetime column to NaN.

           Args:
               df (pandas.DataFrame): The DataFrame to modify.
               datetime_col_name (str): The name of the column containing datetime data.

           Returns:
               pandas.DataFrame: The modified DataFrame with text converted to NaN.
           """
           # Select all columns except the datetime column
           non_datetime_cols = [col for col in Data_avg_final.columns if col != datetime_col_name]

           # Convert text to NaN in non-datetime columns
           Data_avg_final[non_datetime_cols] = Data_avg_final[non_datetime_cols].apply(pd.to_numeric, errors='coerce')

           return Data_avg_final
```

Figure C 3: *Python code to change text data into NAN values*

```
In [35]: missing_values = Data_avg_final.isnull().sum()
         columns_to_drop = missing_values[missing_values >= 500].index

         # Drop the columns with 67 or more missing values
         Data_avg_final_cleaned = Data_avg_final.drop(columns=columns_to_drop)

         # Display information about the dropped columns
         print(f"Dropped columns: {', '.join(columns_to_drop)}")

         # Optionally, you can save the cleaned dataset to a new file
         Data_avg_final_cleaned.to_csv("Data_avg_Final_cleaned.csv", index=False)

         Dropped columns: 440FT008.pulp_bubble_size_mean_mm, 440FT009.pulp_bubble_size_mean_mm, 440FT008.pulp_Jg, 440FT009.pulp_Jg

In [36]: Data_avg_final_cleaned = Data_avg_final_cleaned.drop(['FT-CELL-440-FT-008.MASS-PULL-FRACTION-PV.FROTH-CAM',
                                                              'FT-CELL-440-FT-009.MASS-PULL-FRACTION-PV.FROTH-CAM',
                                                              'CAM-440-FT-011-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING',
                                                              'CAM-440-FT-013-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING',
                                                              'CAM-440-FT-010-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING',
                                                              'CAM-440-FT-014-FROTH-CAMERA.MEAN-BUBBLE-SIZE.READING'],axis= 1)

In [194]: Data_avg_final_cleaned= Data_avg_final_cleaned.dropna(subset=["407WIT107B/PV.V",
                                                                       "L:MOGN.4T.Recovery.Daily.Substitute",
                                                                       "L:MOGN.4T.Conc.Assay.Daily.Substitute"], how="any")
          Data_avg_final_cleaned = Data_avg_final_cleaned.drop(Data_avg_final_cleaned[Data_avg_final_cleaned["407WIT107B/PV.V"] < 500].inde
```

Figure C 4: *Adopted data cleaning strategy for removing missing data columns and rows.*

```
In [48]:
         # Identify columns with missing data
         missing_data = Data_avg_final_cleaned.isnull().sum()
         missing_data = missing_data[missing_data > 0]  # Filter columns with missing values

         # Create a summary DataFrame for missing data
         missing_data_summary = pd.DataFrame({
             "Column": missing_data.index,
             "Missing Count": missing_data.values
         }).reset_index(drop=True)

         # Display the summary table
         print("Columns with Missing Data:")
         print(missing_data_summary)
```

Figure C 5: *Python code for checking the number of values present in the dataset.*

```
In [58]: from sklearn.impute import KNNImputer
         # Specify the column to impute
         column_to_impute = "CAM-440-FT-010-FROTH-CAMERA.LEVEL.READING"

         # Create the KNN imputer instance
         imputer = KNNImputer(n_neighbors=5)  # Adjust n_neighbors as needed

         # Apply imputation to the specified column
         Data_avg_final_cleaned[column_to_impute] = imputer.fit_transform(Data_avg_final_cleaned[[column_to_impute]])
         Data_avg_final_cleaned.info()
         # The imputed data is now in the 'df' DataFrame
         print(Data_avg_final_cleaned)
```

Figure C 6: *Imputing code that was applied to froth camera level and some other variables*

The above code starts by calling all the necessary python built-in modules and third-party libraries to prepare the environment for the code to execute. This is followed by normalizing the data to eliminate biasness. The model is then developed with 60 estimators running at 42 random states and set to select the best 15 informative features from the scaled data named X. The results are then printed along with the indices.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import RFE
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Normalize the data
scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

# Initialize the random forest regressor model
model = RandomForestRegressor(n_estimators=60, random_state=42)

rfe = RFE(model, n_features_to_select=18)  # Adjust the number of features as needed

# Fit RFE
rfe.fit(X_scaled, y)

# Get the ranking of the features
ranking = rfe.ranking_

# Get the indices of selected features
selected_feature_indices = [i for i, rank in enumerate(ranking) if rank == 1]

# Printing the selected features
print("Selected features:")
for idx in selected_feature_indices:
    print(X_scaled.columns[idx])

# New DataFrame with only the selected features
X_selected = X_scaled.iloc[:, selected_feature_indices]
```

Figure C 7: *Feature Selection and Multicollinearity Code*

The following code is for calculating the Variance Inflation Factor (VIF) for each variable in a dataset to check for multicollinearity. High multicollinearity can make an algorithm less stable and reduce its ability to generalize well on new data. VIF quantifies how much the variance of a regression coefficient is inflated due to multicollinearity, and values above 5 (or sometimes 10) typically indicate a problematic level of multicollinearity

```python
# To calculate VIF selected variables
def calculate_vif(X):
    vif_data = pd.DataFrame()
    vif_data['Feature'] = X.columns
    vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    return vif_data

# Calculate VIF for the selected features
vif = calculate_vif(X_selected)

# Print the VIF values
print("\nVIF values for the selected features:")
print(vif)

# Remove features with VIF greater than a threshold (e.g., 3)
vif_threshold = 3
filtered_features = vif[vif['VIF'] <= vif_threshold]['Feature']

# Create a DataFrame with features that have acceptable VIF values
X_filtered = X_selected[filtered_features]

# Calculate VIF for the filtered features
vif_filtered = calculate_vif(X_filtered)

# Print the filtered VIF values
print("\nVIF values after removing highly correlated features:")
print(vif_filtered)

# Plot VIF values before and after filtering
plt.figure(figsize=(14, 6))
```

Figure C 8: *Variance Inflation Factor code created in Python.*

```python
selected_features = [
    'Estimate_PGMConc.Assay_calc_Samplehead',
    'Rougher_Surge_Deliveryflow_cntrl(m3/h)',
    'FT8_Flow_Frother_Reading',
    'SUMP-408-FLOW-COLLECTOR.PV',
    'SUMP-408-FLOW-Promoter.PV',
    'FT-CELL-440-FT-008.FLOW-DEPRESSANT.READING',
    'FT8_Mass-pull_Reading',
    'CAM_FT8_ABS-Froth-Velocity',
    'FT-CELL-440-FT-008.RESIDENCE-TIME.RAW',
    'FT8_Airfolw',
    'CAM_FT8_Froth-Level',
    'CAM_FT9_ABS-Froth-Velocity',
    'FT9_Mass-pull_Reading',
    'FT-CELL-440-FT-009.RESIDENCE-TIME.RAW',
    'FT9_Airfolw',
    'FT-CELL-440-FT-012.AIR-FLOW.READING',
    'FT-CELL-440-FT-010.AIR-FLOW.READING',
    'CAM-440-FT-010-FROTH-CAMERA.LEVEL.READING',
    'FT-CELL-440-FT-011.AIR-FLOW.READING',
    'FT-CELL-440-FT-013.AIR-FLOW.READING',
    'CAM-440-FT-010-FROTH-CAMERA.FROTH-VELOCITY.READING',
    'CAM-440-FT-011-FROTH-CAMERA.ABS-FROTH-VELOCITY.READING'
]

X2 = X[selected_features]
print(X2)
    return res ? go(f, res[1], acc.concat([res[0]])) : acc
  }
  return go(f, seed, [])
}
```

Figure C 9: *manual Feature Selection Code on Python.*

The manual feature selection approach focused primarily on variables related to second rougher cells 8 and 9. This decision was driven by process knowledge, as over 65% of the final concentrate grade is typically recovered from this section of the flotation circuit. While this provided a logical and process-informed basis, it introduced a high degree of bias by overrepresenting a specific part of the circuit. As a result, potentially valuable information from earlier or peripheral stages of flotation was excluded, which may have limited the model's ability to generalise across broader operating conditions. This highlights a key limitation of manually guided selection: although informed by domain expertise, it may restrict the diversity and balance of features needed for robust machine learning performance. This observation opens a compelling avenue for future research to systematically investigate the extent to which manual variable selection influences model generalisation, bias, and performance in complex metallurgical systems.

```python
def filter_data_by_date(Data_avg_final_cleaned1):
    """Filters data based on datetime column to get training and testing sets as specified."""
    try:
        # Ensure datetime column is in datetime format
        Data_avg_final_cleaned1['datetime'] = pd.to_datetime(Data_avg_final_cleaned1['datetime'])

        # Initialize an empty list for test data indices
        test_data_indices = []

        # Define the range of years and months
        years = range(2019, 2024)
        months = range(1, 13)

        # Loop through each year and each month
        for year in years:
            for month in months:
                # Filter data for the current month and year
                month_data = Data_avg_final_cleaned1[
                    (Data_avg_final_cleaned1['datetime'].dt.year == year) &
                    (Data_avg_final_cleaned1['datetime'].dt.month == month)
                ]

                # Check the number of data points in the month
                num_data_points = len(month_data)

                if num_data_points >= 7:
                    # Select 6 data points if less than 24, otherwise 7 data points
                    if num_data_points < 24:
                        selected_days = np.random.choice(month_data.index, size=6, replace=False)
                    else:
                        selected_days = np.random.choice(month_data.index, size=7, replace=False)

                    test_data_indices.extend(selected_days)
```

Figure C 10: *Systematic data splitting into training and testing set code for sequential data*

The following highlights the implementation of machine learning models developed to predict PGMs grade in the study. The code and workflows adhere to a standardized framework for reproducibility and transparency, encompassing Data Preprocessing which involves Feature scaling and train-test splits. The Model Architectures also include Evaluation Metrics and cross-validation Strategies and ends with visualization using time-series predictions, residual plots. The models including linear regressor, support vector regression, Random Forest, Gradient Boosting, and Neural Network were designed to address distinct aspects of the problem.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.feature_selection import SelectFromModel

def train_and_evaluate_modellr(X_train, y_train, X_test, y_test,y,X, cv=5):
    try:
        # Apply StandardScaler to scale the input dataset
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)
        X_scaled = scaler.transform(X.drop(['datetime'], axis=1))


        # Training the Linear Regression model
        lin_reg = LinearRegression()
        lin_reg.fit(X_train_scaled, y_train)

        # Making predictions
        y_pred_train = lin_reg.predict(X_train_scaled)
        y_pred_test = lin_reg.predict(X_test_scaled)
        y_pred = lin_reg.predict(X_scaled)

        # Calculate evaluation metrics
        mse_train = mean_squared_error(y_train, y_pred_train)
        rmse_train = np.sqrt(mse_train)
        mae_train = mean_absolute_error(y_train, y_pred_train)
        r2_train = r2_score(y_train, y_pred_train)

        mse_test = mean_squared_error(y_test, y_pred_test)
        rmse_test = np.sqrt(mse_test)
        mae_test = mean_absolute_error(y_test, y_pred_test)
        r2_test = r2_score(y_test, y_pred_test)

        print("Training Set Metrics:")
        print(f"MSE: {mse_train:.4f}")
        print(f"RMSE: {rmse_train:.4f}")
        print(f"MAE: {mae_train:.4f}")
        print(f"R-squared: {r2_train:.4f}")

        print("Test Set Metrics:")
        print(f"MSE: {mse_test:.4f}")
        print(f"RMSE: {rmse_test:.4f}")
        print(f"MAE: {mae_test:.4f}")
        print(f"R-squared: {r2_test:.4f}")

        # Plot actual vs predicted
        fig, ax = plt.subplots(figsize=(24, 8))
        ax.plot(Data_avg_final_cleaned1.index, y_pred, label='Predicted', color='red')
        ax.scatter(X_train_time, y_train, label='Actual (Training)', color='blue')
        ax.scatter(X_test_time, y_test, label='Actual (Testing)', color='green')
        ax.set_xlabel('Datetime')
        ax.set_ylabel('4T Grade')
        ax.set_title('Actual vs. Predicted 4T Conc Grade (Entire Dataset)')
        ax.legend()
        ax.grid(True)
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        # Plot actual vs predicted values for training and testing sets
        fig, axs = plt.subplots(1, 2, figsize=(18, 4))
        axs[0].scatter(y_train, y_pred_train, color='blue', label='Test Set Predictions')
        axs[0].plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[0].set_xlabel('Actual Values')
        axs[0].set_ylabel('Predicted Values')
        axs[0].set_title('Actual vs. Predicted Values (Training Set)')
        axs[0].legend()
        axs[0].grid(True)

        axs[1].scatter(y_test, y_pred_test, color='green', label='Test Set Predictions')
        axs[1].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[1].set_xlabel('Actual Values')
        axs[1].set_ylabel('Predicted Values')
        axs[1].set_title('Actual vs. Predicted Values (Test Set)')
        axs[1].legend()
        axs[1].grid(True)

    except Exception as e:
        print(f"An error occurred: {e}")

train_and_evaluate_modellr(X_train, y_train, X_test, y_test,y,X)
```

Figure C 11: *Linear Regression model python code for training, validation and performance plots.*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import LinearSVR
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

def train_and_evaluate_modelsvm(X_train, y_train, X_test, y_test, y, X):
    try:
# Apply StandardScaler
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)
        X_scaled = scaler.transform(X.drop(['datetime'], axis=1))

        # Initialize and train SVM model
        model = LinearSVR()
        model.fit(X_train_scaled, y_train)

        # Make predictions
        y_pred_train = model.predict(X_train_scaled)
        y_pred_test = model.predict(X_test_scaled)
        y_pred = model.predict(X_scaled)

        # Calculate evaluation metrics
        mse_train = mean_squared_error(y_train, y_pred_train)
        rmse_train = np.sqrt(mse_train)
        mae_train = mean_absolute_error(y_train, y_pred_train)
        r2_train = r2_score(y_train, y_pred_train)

        mse_test = mean_squared_error(y_test, y_pred_test)
        rmse_test = np.sqrt(mse_test)
        mae_test = mean_absolute_error(y_test, y_pred_test)
        r2_test = r2_score(y_test, y_pred_test)

        print("Training Set Metrics:")
        print(f"MSE: {mse_train:.4f}")
        print(f"RMSE: {rmse_train:.4f}")
        print(f"MAE: {mae_train:.4f}")
        print(f"R-squared: {r2_train:.4f}")

        print("Test Set Metrics:")
        print(f"MSE: {mse_test:.4f}")
        print(f"RMSE: {rmse_test:.4f}")
        print(f"MAE: {mae_test:.4f}")
        print(f"R-squared: {r2_test:.4f}")

        # Plot actual vs predicted
        fig, ax = plt.subplots(figsize=(24, 8))
        ax.plot(Data_avg_final_cleaned1.index, y_pred, label='Predicted', color='red')
        ax.scatter(X_train_time, y_train, label='Actual (Training)', color='blue')
        ax.scatter(X_test_time, y_test, label='Actual (Testing)', color='green')
        ax.set_xlabel('Datetime')
        ax.set_ylabel('4T Grade')
        ax.set_title('Actual vs. Predicted 4T Conc Grade (Entire Dataset)')
        ax.legend()
        ax.grid(True)
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        # Plot actual vs predicted values for training and testing sets
        fig, axs = plt.subplots(1, 2, figsize=(18, 4))
        axs[0].scatter(y_train, y_pred_train, color='blue', label='Test Set Predictions')
        axs[0].plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[0].set_xlabel('Actual Values')
        axs[0].set_ylabel('Predicted Values')
        axs[0].set_title('Actual vs. Predicted Values (Training Set)')
        axs[0].legend()
        axs[0].grid(True)

        axs[1].scatter(y_test, y_pred_test, color='green', label='Test Set Predictions')
        axs[1].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[1].set_xlabel('Actual Values')
        axs[1].set_ylabel('Predicted Values')
        axs[1].set_title('Actual vs. Predicted Values (Test Set)')
        axs[1].legend()
        axs[1].grid(True)

    except Exception as e:
        print(f"An error occurred: {e}")

train_and_evaluate_modelsvm(X_train, y_train, X_test, y_test, y, X)
```

Figure C 12: *Support Vector Regression model python code for training, validation and performance plots.*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.ensemble import RandomForestRegressor

def train_and_evaluate_model_rf(X_train, y_train, X_test, y_test, y, X, cv=5):
    try:
        # Apply StandardScaler
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)
        X_scaled = scaler.transform(X.drop(['datetime'], axis=1))

        # Train the Random Forest regressor on the reduced feature set
        rf_regressor = RandomForestRegressor(
            n_estimators=70, random_state=42
        )
        rf_regressor.fit(X_train_scaled, y_train)

        # Make predictions
        y_pred_train = rf_regressor.predict(X_train_scaled)
        y_pred_test = rf_regressor.predict(X_test_scaled)
        y_pred = rf_regressor.predict(X_scaled)

        # Calculate evaluation metrics
        mse_train = mean_squared_error(y_train, y_pred_train)
        rmse_train = np.sqrt(mse_train)
        mae_train = mean_absolute_error(y_train, y_pred_train)
        r2_train = r2_score(y_train, y_pred_train)

        mse_test = mean_squared_error(y_test, y_pred_test)
        rmse_test = np.sqrt(mse_test)
        mae_test = mean_absolute_error(y_test, y_pred_test)
        r2_test = r2_score(y_test, y_pred_test)

        print("Training Set Metrics:")
        print(f"MSE: {mse_train:.4f}")
        print(f"RMSE: {rmse_train:.4f}")
        print(f"MAE: {mae_train:.4f}")
        print(f"R-squared: {r2_train:.4f}")

        print("Test Set Metrics:")
        print(f"MSE: {mse_test:.4f}")
        print(f"RMSE: {rmse_test:.4f}")
        print(f"MAE: {mae_test:.4f}")
        print(f"R-squared: {r2_test:.4f}")

        # Perform cross-validation
        cross_val_scores = cross_val_score(rf_regressor, X_scaled, y, cv=cv, scoring='r2')
        print(f"Cross-Validation R-squared scores: {cross_val_scores}")
        print(f"Mean Cross-Validation R-squared score: {np.mean(cross_val_scores):.4f}")

        # Plot actual vs predicted
        fig, ax = plt.subplots(figsize=(24, 8))
        ax.plot(Data_avg_final_cleaned1.index, y_pred, label='Predicted', color='red')
        ax.scatter(X_train_time, y_train, label='Actual (Training)', color='blue')
        ax.scatter(X_test_time, y_test, label='Actual (Testing)', color='green')
        ax.set_xlabel('Datetime')
        ax.set_ylabel('4T Recovery')
        ax.set_title('Actual vs. Predicted 4T Conc Grade (Entire Dataset)')
        ax.legend()
        ax.grid(True)
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        # Plot actual vs predicted values for training and testing sets
        fig, axs = plt.subplots(1, 2, figsize=(18, 4))
        axs[0].scatter(y_train, y_pred_train, color='blue', label='Test Set Predictions')
        axs[0].plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[0].set_xlabel('Actual Values')
        axs[0].set_ylabel('Predicted Values')
        axs[0].set_title('Actual vs. Predicted Values (Training Set)')
        axs[0].legend()
        axs[0].grid(True)

        axs[1].scatter(y_test, y_pred_test, color='green', label='Test Set Predictions')
        axs[1].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[1].set_xlabel('Actual Values')
        axs[1].set_ylabel('Predicted Values')
        axs[1].set_title('Actual vs. Predicted Values (Test Set)')
        axs[1].legend()
        axs[1].grid(True)

    except Exception as e:
        print(f"An error occurred: {e}")
train_and_evaluate_model_rf(X_train, y_train, X_test, y_test,y,X) {
    const res = f(seed)
    return res ? go(f, res[1], acc.concat([res[0]])) : acc
  }
  return go(f, seed, [])
}
```

Figure C 13: *Random Forest Regression model python code for training, validation and performance plots.*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization, Dropout, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from lime.lime_tabular import LimeTabularExplainer
import shap
import warnings

warnings.filterwarnings("ignore", category=UserWarning)

def train_and_evaluate_modelNN(X_train, y_train, X_test, y_test, y, X, cv=5):
    try:
        # Apply StandardScaler
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)
        X_scaled = scaler.transform(X.drop(['datetime'], axis=1))

        # Define the neural network model
        model = Sequential([
            Input(shape=(X_train_scaled.shape[1],)),
            Dense(64, activation='relu'),
            Dropout(0.25),
            BatchNormalization(),
            Dense(32, activation='relu'),
            Dropout(0.25),
            BatchNormalization(),
            Dense(16, activation='relu'),
            Dropout(0.25),
            BatchNormalization(),
            Dense(8, activation='relu'),
            Dropout(0.25),
            BatchNormalization()
            Dense(1) ])

        # Compiling the model
        model.compile(optimizer=Adam(learning_rate=0.01), loss='mean_squared_error', metrics=['mae'])

        # Early stopping and learning rate reduction callbacks
        early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
        reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=0.0001)

        # Training the NN model
        history = model.fit(X_train_scaled, y_train, epochs=100, batch_size=32, validation_data=
(X_test_scaled, y_test),
                            callbacks=[early_stopping, reduce_lr], verbose=1)

        # Make predictions
        y_pred_train = model.predict(X_train_scaled).flatten()
        y_pred_test = model.predict(X_test_scaled).flatten()
        y_pred = model.predict(X_scaled).flatten()

        # Calculate evaluation metrics
        mse_train = mean_squared_error(y_train, y_pred_train)
        rmse_train = np.sqrt(mse_train)
        mae_train = mean_absolute_error(y_train, y_pred_train)
        r2_train = r2_score(y_train, y_pred_train)

        mse_test = mean_squared_error(y_test, y_pred_test)
        rmse_test = np.sqrt(mse_test)
        mae_test = mean_absolute_error(y_test, y_pred_test)
        r2_test = r2_score(y_test, y_pred_test)

        print("Training Set Metrics:")
        print(f"MSE: {mse_train:.4f}")
        print(f"RMSE: {rmse_train:.4f}")
        print(f"MAE: {mae_train:.4f}")
        print(f"R-squared: {r2_train:.4f}")

        print("Test Set Metrics:")
        print(f"MSE: {mse_test:.4f}")
        print(f"RMSE: {rmse_test:.4f}")
        print(f"MAE: {mae_test:.4f}")
        print(f"R-squared: {r2_test:.4f}")

        # Plot actual vs predicted for the entire dataset
        fig, ax = plt.subplots(figsize=(26, 8))
        ax.plot(Data_avg_final_cleaned1.index, y_pred, label='Predicted', color='red')
        ax.scatter(X_train_time, y_train, label='Actual (Training)', color='blue')
        ax.scatter(X_test_time, y_test, label='Actual (Testing)', color='green')
        ax.set_xlabel('Datetime')
        ax.set_ylabel('4T Recovery')
        ax.set_title('Actual vs. Predicted 4T Conc Grade (Entire Dataset)')
        ax.legend()
        ax.grid(True)
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        # Filter data for the year 2019
        start = '2019-01-01'
        end = '2019-12-31'
        mask = (Data_avg_final_cleaned1.index >= start) & (Data_avg_final_cleaned1.index <= end)

        # Plot actual vs predicted for the year 2019
        fig, ax = plt.subplots(figsize=(26, 8))
        ax.plot(Data_avg_final_cleaned1.index[mask], y_pred[mask], label='Predicted', color='red')
        ax.scatter(Data_avg_final_cleaned1.index[mask], y[mask], label='Actual (Training)',
color='blue')
        ax.set_xlabel('Datetime')
        ax.set_ylabel('4T Recovery')
        ax.set_title('Actual vs. Predicted 4T Conc Grade (2019)')
        ax.legend()
        ax.grid(True)
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        # Plot of actual vs predicted values for training and testing sets
        fig, axs = plt.subplots(1, 2, figsize=(18, 4))
        axs[0].scatter(y_train, y_pred_train, color='blue', label='Training Set Predictions')
        axs[0].plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[0].set_xlabel('Actual Values')
        axs[0].set_ylabel('Predicted Values')
        axs[0].set_title('Actual vs. Predicted Values (Training Set)')
        axs[0].legend()
        axs[0].grid(True)

        axs[1].scatter(y_test, y_pred_test, color='green', label='Test Set Predictions')
        axs[1].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[1].set_xlabel('Actual Values')
        axs[1].set_ylabel('Predicted Values')
        axs[1].set_title('Actual vs. Predicted Values (Test Set)')
        axs[1].legend()
        axs[1].grid(True)

    except Exception as e:
        print(f"An error occurred: {e}")

train_and_evaluate_modelNN(X_train, y_train, X_test, y_test, y, X)
```

Figure C 14: *Neural Networks model python code for training, validation and performance plots.*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.feature_selection import SelectFromModel
from sklearn.svm import LinearSVR
import xgboost as xgb

def train_and_evaluate_modelxgbc(X_train, y_train, X_test, y_test,y,X, cv=5):
    try:
        # Apply StandardScaler
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)
        X_scaled = scaler.transform(X.drop(['datetime'], axis=1))


        # Train the XGBoost regressor on the reduced feature set
        xgb_regressor = xgb.XGBRegressor(objective='reg:squarederror',learning_rate=0.2,max_depth=
3,colsample_bytree= 0.6,
            random_state=42
        )
        xgb_regressor.fit(X_train_scaled, y_train)

        # Make predictions
        y_pred_train = xgb_regressor.predict(X_train_scaled)
        y_pred_test = xgb_regressor.predict(X_test_scaled)
        y_pred = xgb_regressor.predict(X_scaled)

        # Calculate evaluation metrics
        mse_train = mean_squared_error(y_train, y_pred_train)
        rmse_train = np.sqrt(mse_train)
        mae_train = mean_absolute_error(y_train, y_pred_train)
        r2_train = r2_score(y_train, y_pred_train)

        mse_test = mean_squared_error(y_test, y_pred_test)
        rmse_test = np.sqrt(mse_test)
        mae_test = mean_absolute_error(y_test, y_pred_test)
        r2_test = r2_score(y_test, y_pred_test)

        print("Training Set Metrics:")
        print(f"MSE: {mse_train:.4f}")
        print(f"RMSE: {rmse_train:.4f}")
        print(f"MAE: {mae_train:.4f}")
        print(f"R-squared: {r2_train:.4f}")

        print("Test Set Metrics:")
        print(f"MSE: {mse_test:.4f}")
        print(f"RMSE: {rmse_test:.4f}")
        print(f"MAE: {mae_test:.4f}")
        print(f"R-squared: {r2_test:.4f}")

        # Plot actual vs predicted
        fig, ax = plt.subplots(figsize=(26, 8))
        ax.plot(Data_avg_final_cleaned1.index, y_pred, label='Predicted', color='red')
        ax.scatter(X_train_time, y_train, label='Actual (Training)', color='blue')
        ax.scatter(X_test_time, y_test, label='Actual (Testing)', color='green')
        ax.set_xlabel('Datetime')
        ax.set_ylabel('4T Recovery')
        ax.set_title('Actual vs. Predicted 4T Conc Grade (Entire Dataset)')
        ax.legend()
        ax.grid(True)
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        # Plot actual vs predicted values for training and testing sets
        fig, axs = plt.subplots(1, 2, figsize=(18, 4))
        axs[0].scatter(y_train, y_pred_train, color='blue', label='Test Set Predictions')
        axs[0].plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[0].set_xlabel('Actual Values')
        axs[0].set_ylabel('Predicted Values')
        axs[0].set_title('Actual vs. Predicted Values (Training Set)')
        axs[0].legend()
        axs[0].grid(True)

        axs[1].scatter(y_test, y_pred_test, color='green', label='Test Set Predictions')
        axs[1].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
                    color='red', linestyle='--', label='Perfect Fit')
        axs[1].set_xlabel('Actual Values')
        axs[1].set_ylabel('Predicted Values')
        axs[1].set_title('Actual vs. Predicted Values (Test Set)')
        axs[1].legend()
        axs[1].grid(True)

    except Exception as e:
        print(f"An error occurred: {e}")

# Example usage:
train_and_evaluate_modelxgbc(X_train, y_train, X_test, y_test,y,X)
```

***Figure C 15**: XGBoost Regression base model python code for training, validation and performance plots.*

Appendix D - Algorithms Performances

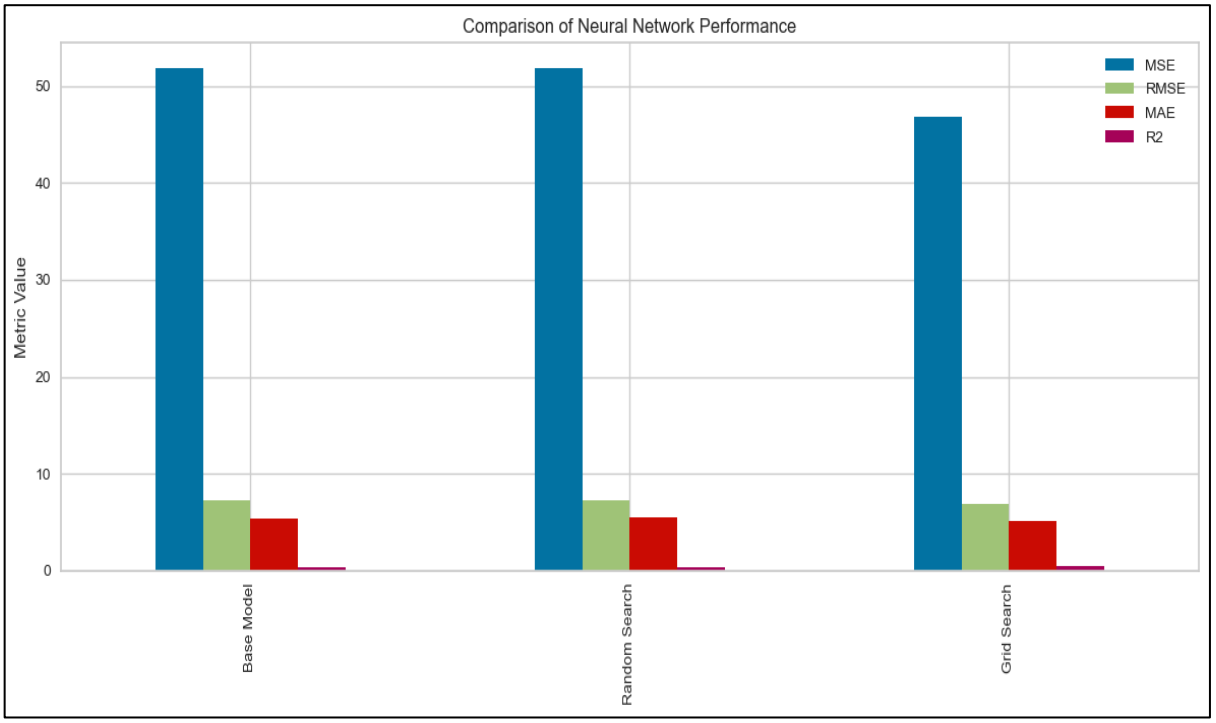


Figure D 1: *Neural network before and after hyperparameter tuning with random search and grid search*

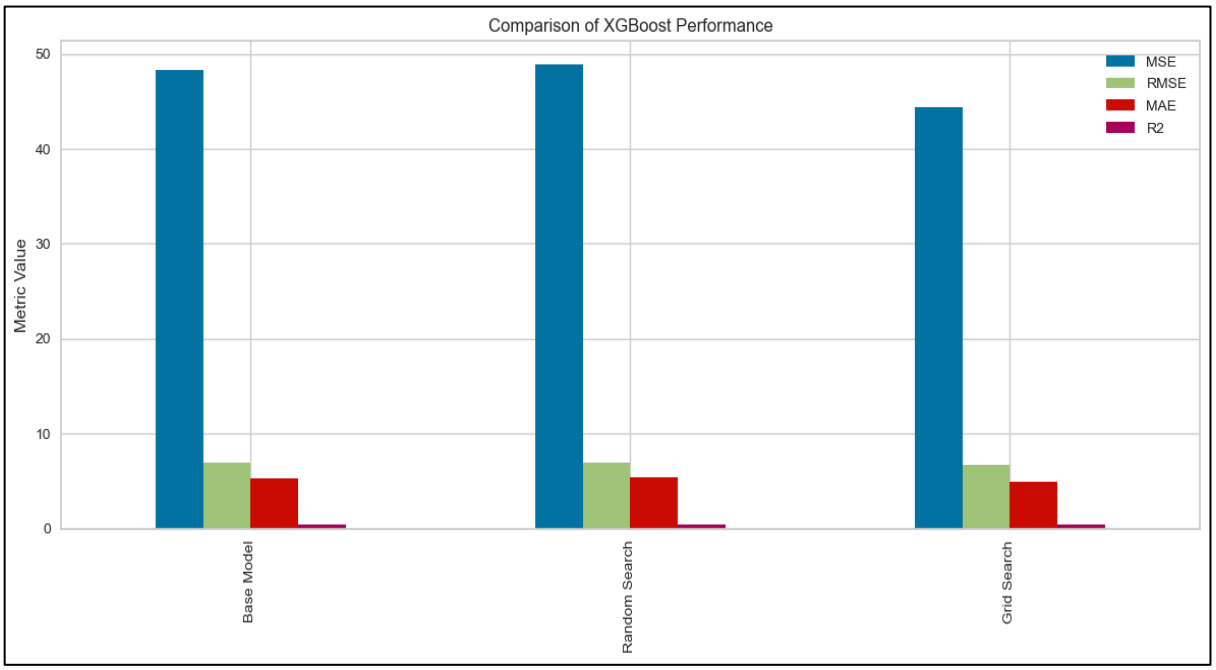


Figure D 2: *XGBoost performance before and after hyperparameter tuning with grid search and random search*

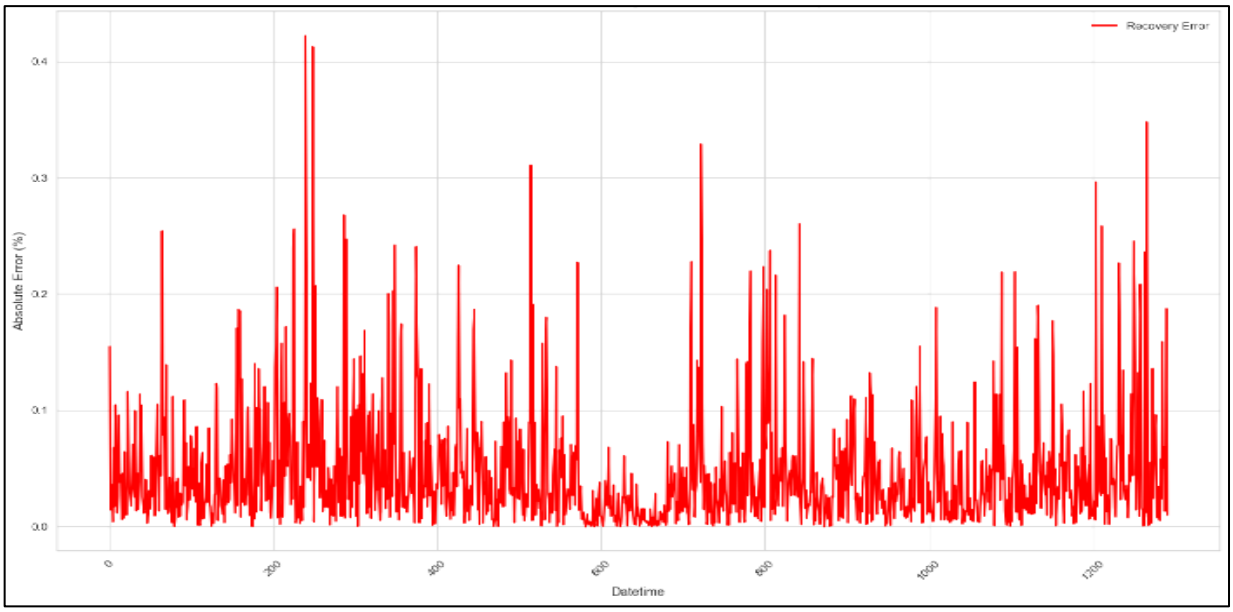


Figure D 3: *Error margin between recovery calculated from models grade prediction and the recovery calculated from process data.*

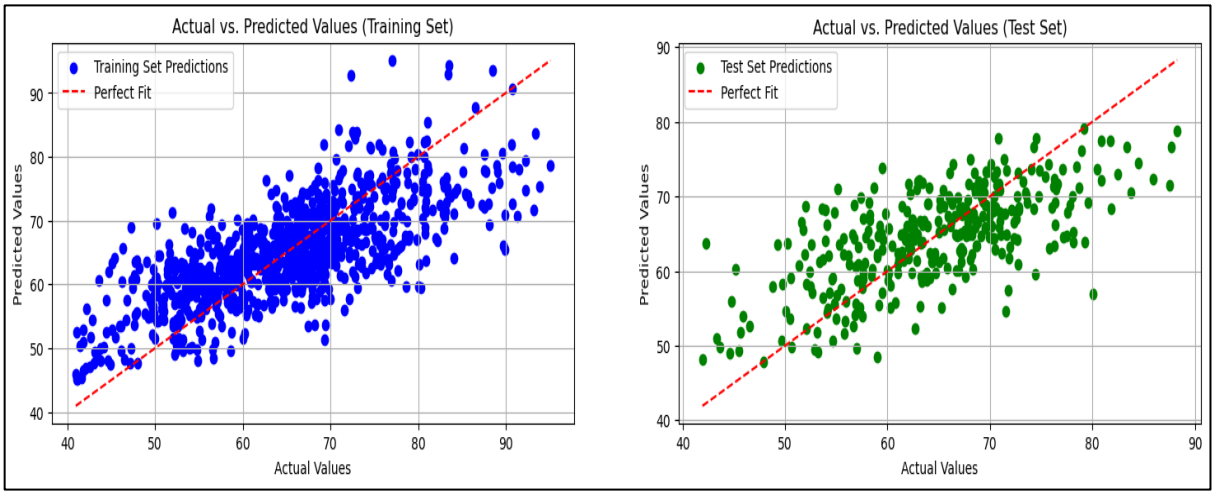


Figure D 5: *Linear regression parity plot for grade prediction*

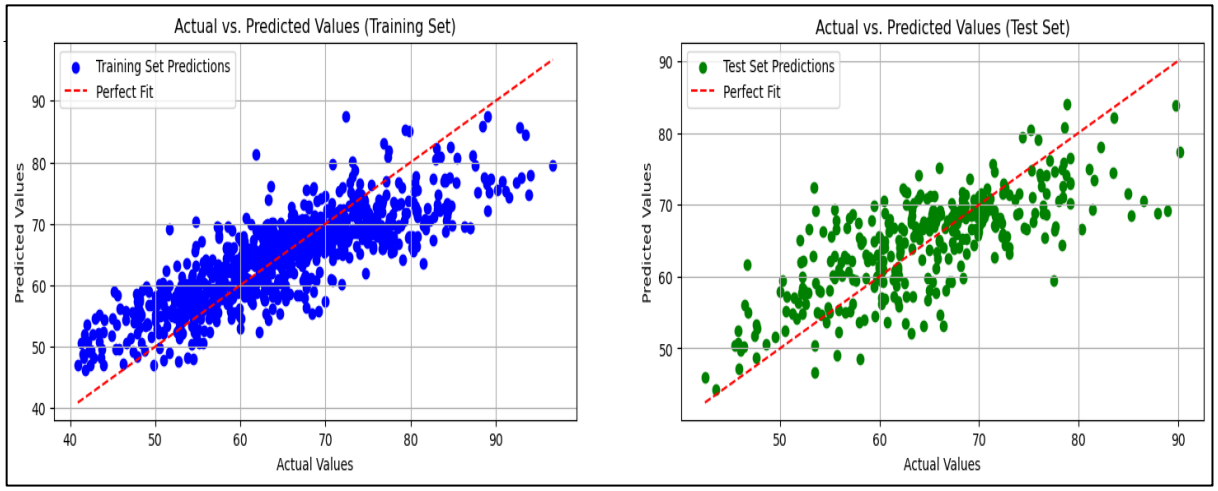


Figure D 4*: Neural Network parity plot for PGMs grade prediction*

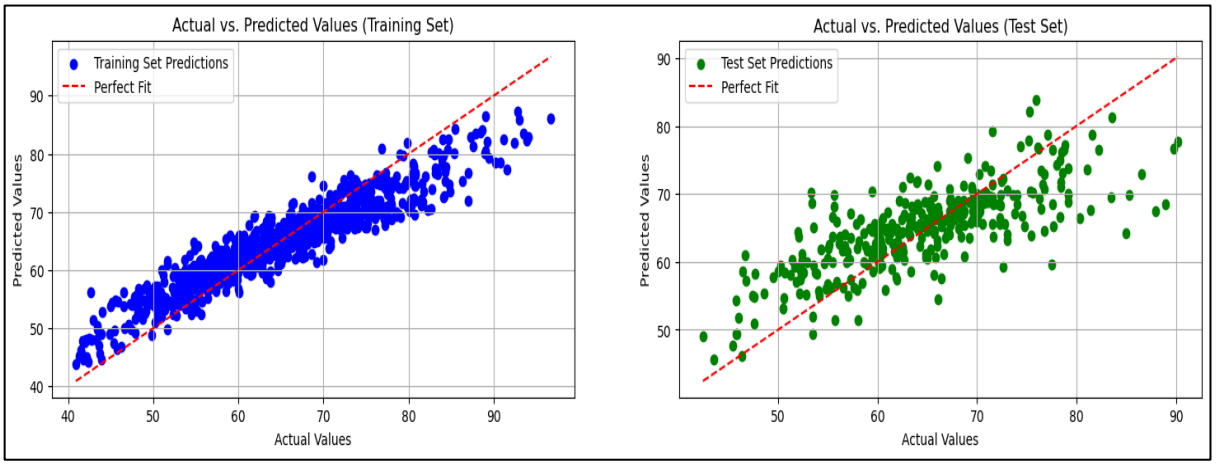


Figure D 6: *Random Forest parity plot for PGMs grade prediction*

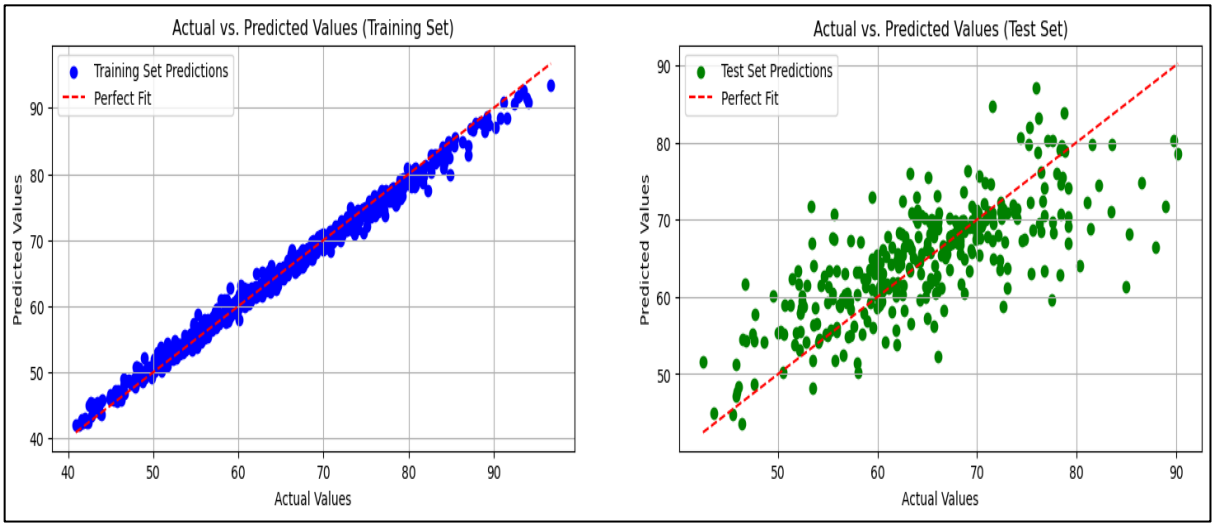


Figure D 7: *XGBoost parity plot for PGMs grade prediction*

Appendix D_1 Support Vector Machine Performance

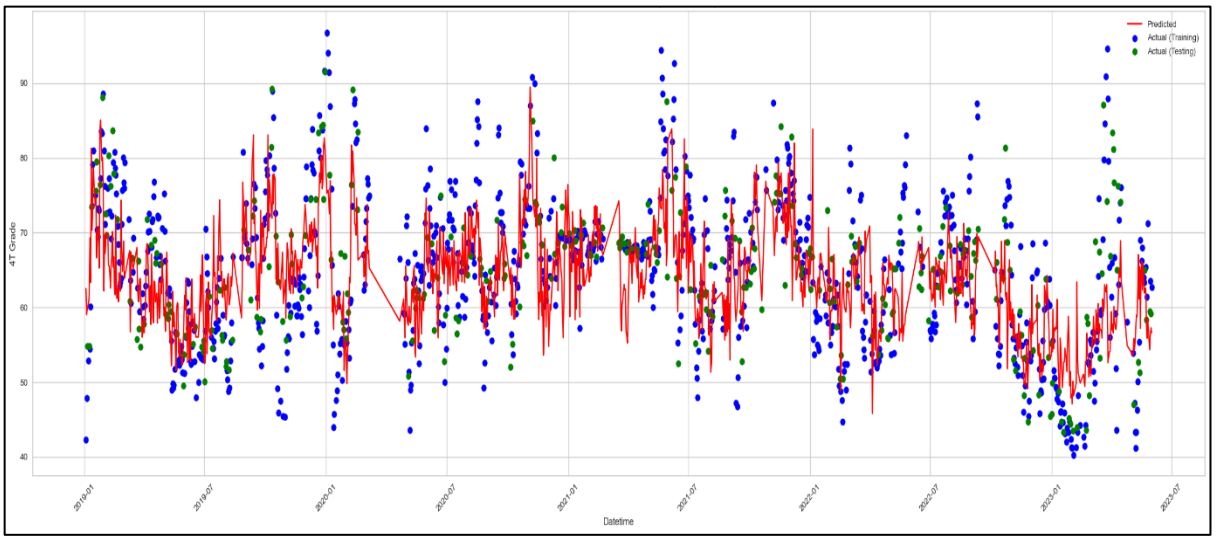


Figure D_1 1: *Support Vector Regression performance for grade prediction over time*

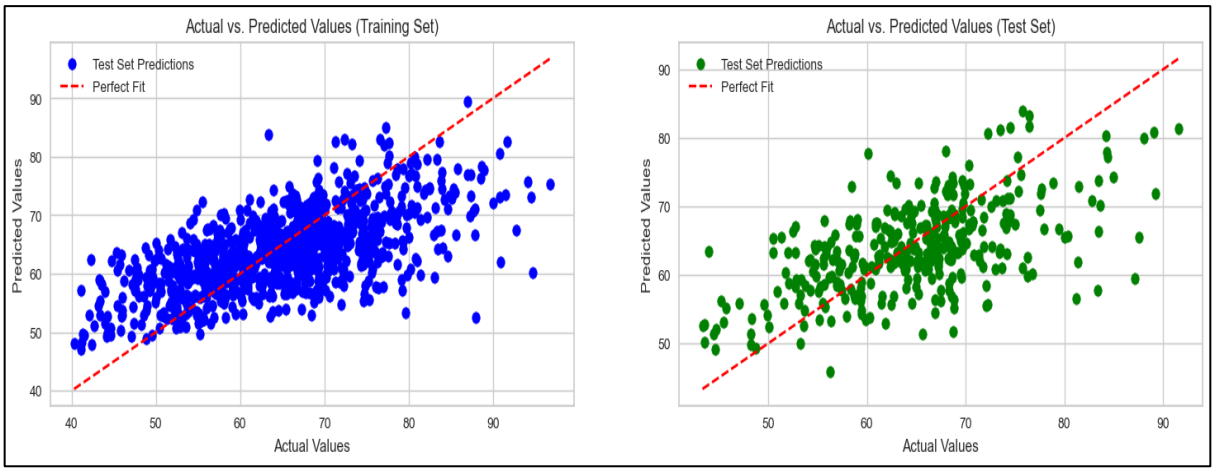


Figure D_1 2: *Support Vector Machine parity plots for both training and testing set.*

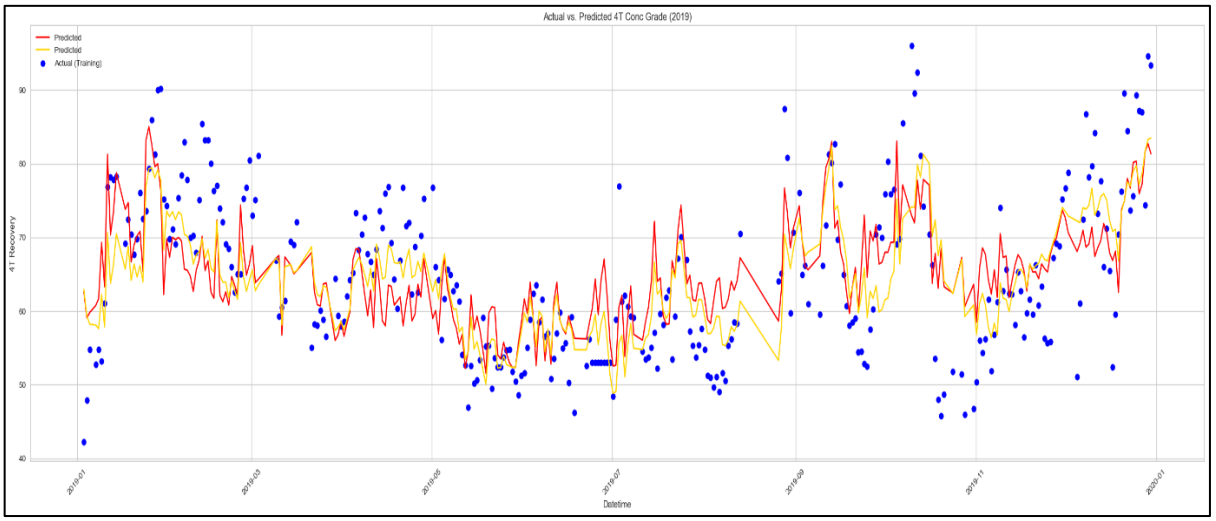


Figure D_1 3: *Support vector model performance vs the Leuenberger observer integration over time.*

Appendix D_2 Leuenberger Observer Performance Plots

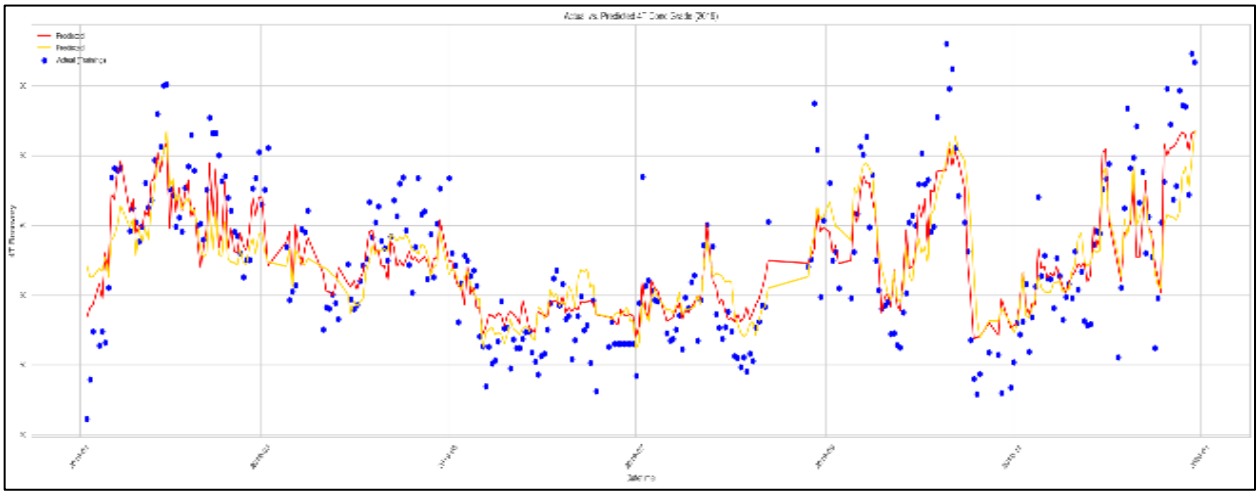


Figure D_2 1: *Neural Network performance with vs without the observer.*

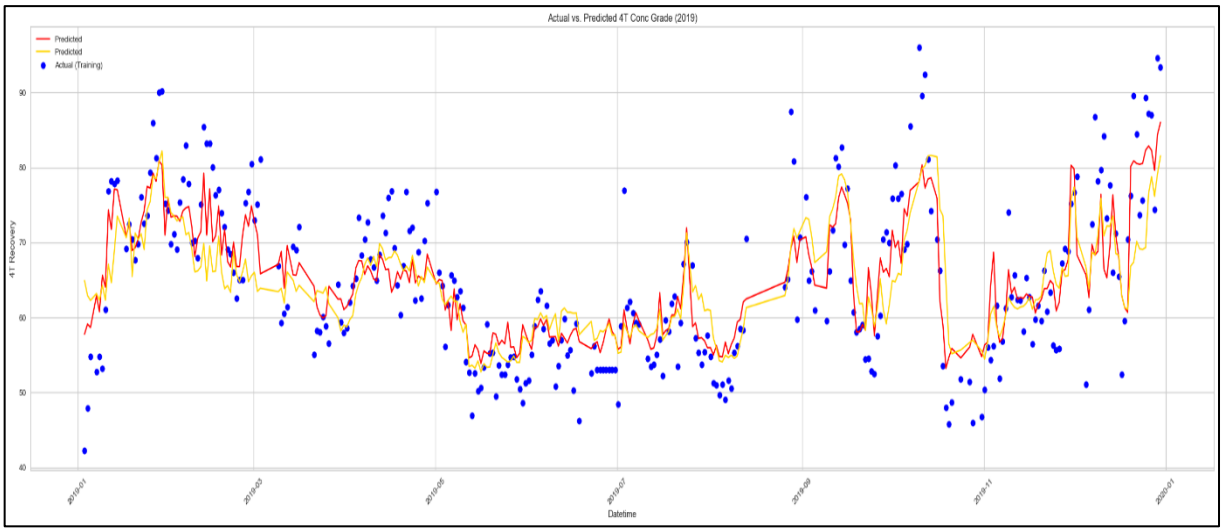


Figure D_2 2: *Random Forest performance with vs without the observer.*

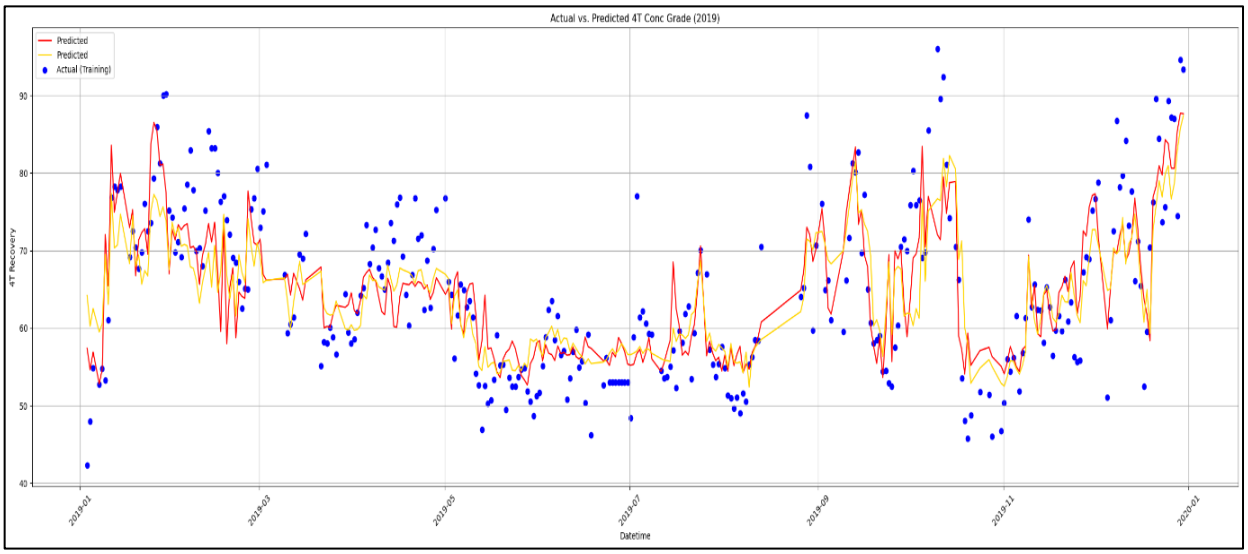


Figure D_2 3: *XGBoost performance with vs without the observer.*

The following code illustrates how the Luenberger Observer was implemented and integrated into the machine learning pipeline for soft sensor development. The observer was used to estimate unmeasured or partially observed process states. The observer-generated estimates were used to augment the feature set before training machine learning models. The code below demonstrates the full implementation, including observer construction, feature augmentation, and model training. This hybrid approach combines domain-based estimation with data-driven learning for more robust soft sensor performance.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

class LuenbergerObserver:
    def __init__(self, A, B, C, L):
        self.A = A
        self.B = B
        self.C = C
        self.L = L
        self.x_hat = np.zeros((A.shape[0], 1))

    def estimate(self, u, y):
        self.x_hat = self.A @ self.x_hat + self.B @ u + self.L @ (y - self.C @ self.x_hat)
        return self.x_hat

def augment_with_estimated_states(X, y, observer, u):
    estimated_states = []
    for i in range(X.shape[0]):
        y_obs = np.array([[y[i]]])
        state_estimate = observer.estimate(u[i], y_obs)
        estimated_states.append(state_estimate.flatten())
    estimated_states = np.array(estimated_states)
    return np.hstack([X, estimated_states])

def train_and_evaluate_model_lr(...):  # shortened for appendix display
    # Define system matrices and observer
    # Clean and prepare dataset
    # Apply observer and train ML model
    # Compare original vs corrected predictions
    pass
```

Figure D_2 4*: Python implementation of the Luenberger Observer and its integration into the machine learning workflow.*

The Luenberger observer in the provided code is used to estimate unmeasured or partially observed internal states of a dynamic system based on available input (u) and output (y) data. These estimated states are computed recursively using the observer equation:

$\hat{x}_{k+1} = A*\hat{x}_k + B*u_k + L*(y_k - C*\hat{x}_k)$,

where $\hat{x}$ is the estimated state vector, and L is the observer gain matrix that corrects estimation error by comparing actual and predicted outputs. This state estimation process is integrated with machine learning by augmenting the original dataset X with the observer-generated state estimates through the augment_with_estimated_states function. This results in a more informative feature set, which can improve the performance of downstream models, such as the Linear Regression model shown. By combining model-based estimates with data-driven learning, the approach enhances predictive accuracy and robustness, especially in systems where not all process variables are directly measurable.

Appendix E - Different data splitting techniques explored for the PGMs grade predictions

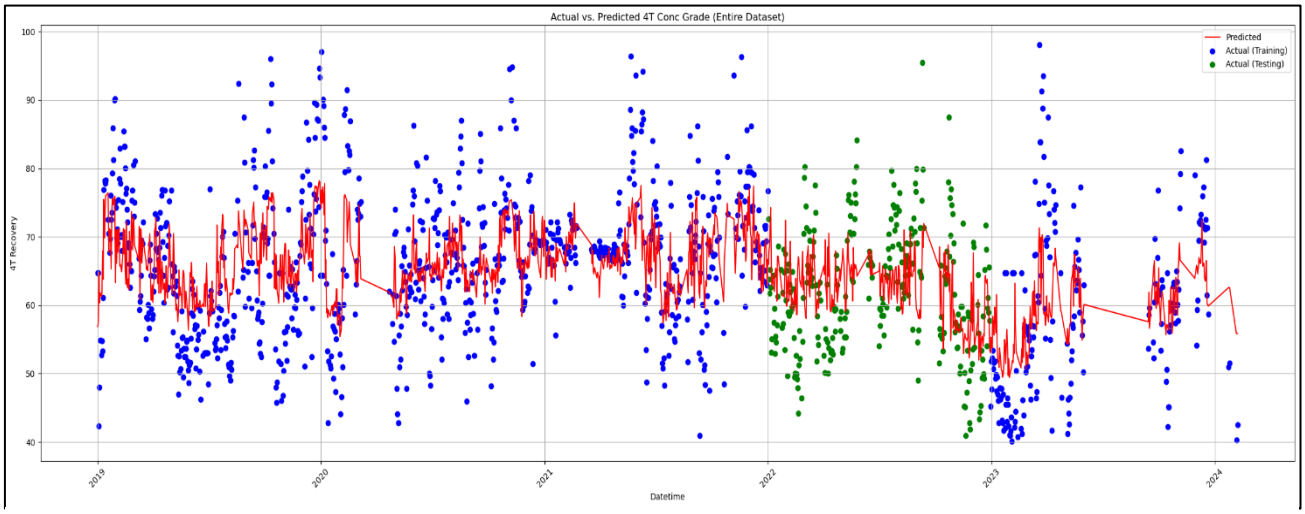


Figure E 1: *Random Forest plot with one year testing data*

The figure above shows the performance of the Random Forest model using sequential data splitting, where only data captured from the start to the end of 2022 was used as the testing set, and the remaining data was used for training. This was the initial approach attempted to assess the impact of data splitting on the model's performance. However, this method yielded lower performance compared to random splitting and the final approach adopted in the research.

The figure shows the performance of the Random Forest model using a sequential data splitting

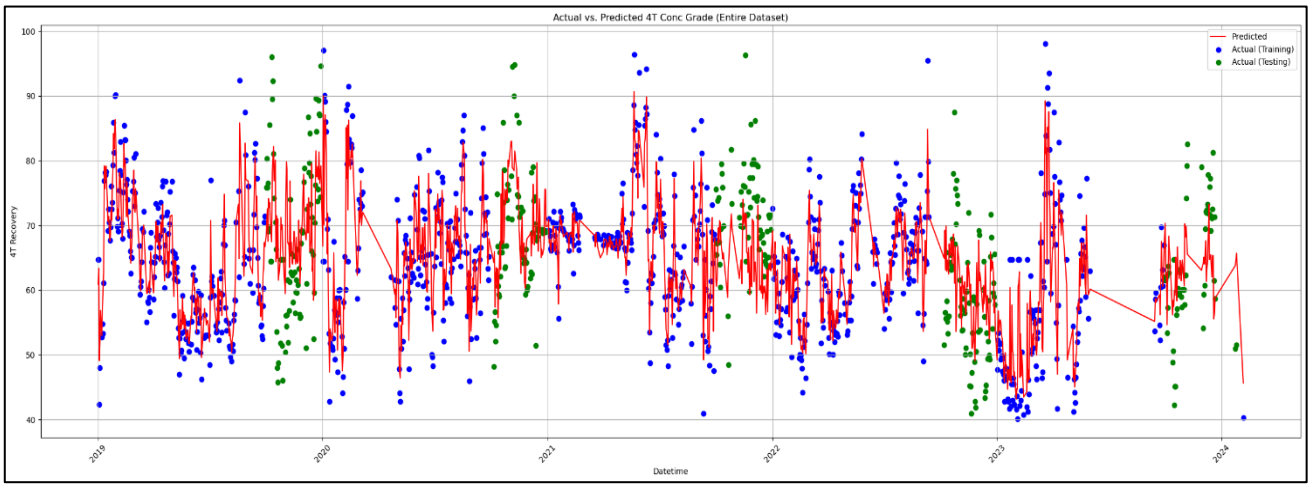


Figure E 2: Random Forest Model Performance with Sequential Data Splitting (Last 3 Months as Test Set)

strategy, where the last three months of the dataset (green dots) were reserved as the testing set, and all preceding data (blue dots) was used for training. The red line represents the model's predictions across the full timeline. This approach was used to simulate a realistic forecasting scenario and evaluate the model's ability to generalize to the most recent operational data. This approach performed better than the one in figure E 1 but still less compared to random splitting and the one adopted in the study.